

**Segunda Edição dos Anais do
VI Workshop-Escola de Sistemas de Agentes,
seus Ambientes e apliCações**

— WESAAC 2012 —

Organizado por

**Jomi Fred Hübner
Anarosa Alves Franco Brandão
Ricardo Silveira
Jerusa Marchi**

Florianopolis, 02-04 Maio de 2012

Anais do VI Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações — VI WESAAC / Hübner, J.F.; Brandão, A.A.F.; (Org) — Florianópolis, 2012.

329p. :il.

2a edição

ISSN 2177-2096

1. Agentes. 2. Sistemas de Agentes. 3. Ambientes para Agentes. 4. Aplicações de Agentes. I. Hübner, J.F. II. Brandão, A.A.F.

CDD

Prefácio

Este documento contempla a 2a. edição dos Anais do VI Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações - WESAAC 2012. Nele encontram-se os trabalhos apresentados na sexta edição do WESAAC, que foi realizado na cidade de Florianópolis-SC, nas dependências da Universidade Federal de Santa Catarina (UFSC), entre os dias 02 e 04 de maio de 2012, com o apoio dos Programas de Pós-Graduação em Engenharia de Automação e Sistemas e em Ciência da Computação da UFSC e da Sociedade Brasileira da Computação (SBC).

Continuando a tradição da série WESAAC, os objetivos do evento continuam relacionados à integração de pesquisadores e estudantes de todos os níveis na área de Agentes e Sistemas de Agentes e divulgação das atividades de pesquisa dos diversos grupos de pesquisa do Brasil, com o intuito de facilitar o intercâmbio de conhecimentos. Para isso, o evento é constituído de uma combinação de Oficinas e Palestras (a parte “escola”), proferidas por pesquisadores experientes, e apresentações de Trabalhos Completos e Resumos Estendidos (a parte “workshop”).

O histórico deste evento, que inicialmente foi denominado “Workshop - Escola de Sistemas de Agentes para Ambientes Colaborativos” e, a partir de sua quarta edição passou a ter a denominação atual, mostra o crescimento constante da comunidade de pesquisadores na área de agentes e sistemas baseados em agentes no Brasil. As três primeiras edições do evento tiveram uma abrangência regional, atingindo especialmente pesquisadores da região Sul do Brasil. A partir da quarta edição, realizada na cidade do Rio Grande - RS, aumentou-se o escopo do evento, ampliando sua abrangência de regional para nacional.

Nesta sexta edição do WESAAC, mantivemos a abrangência nacional, com a participação de pesquisadores destacados da área de sistemas de agentes, de diversas instituições do Brasil, tais como USP, UFF, UFPE, UFRGS, PUC-RS, UFPel e do exterior, notadamente da Universidade Nova de Lisboa e da Ecole des Mines de Saint Etienne, França.

Para esta edição, o evento recebeu uma variedade de contribuições. Foram submetidos 39 artigos, sendo 24 artigos completos e 15 artigos resumidos. Dentre os artigos completos, 15 foram aceitos para apresentação oral, divididas em três sessões técnicas, e 6 foram aceitos para apresentação na forma de poster. Dos artigos resumidos, 11 foram aceitos para apresentação como poster. Todos os artigos aceitos constam deste documento.

Com em anos anteriores, os artigos completos apresentados no evento foram submetidos a um comitê avaliador que selecionou quatro deles para submissão de versão estendida para a Revista de Informática Téorica e Aplicada (RITA), <http://seer.ufrgs.br/rita>. Os melhores artigos foram:

- *Extending the Robocup Rescue to Support Stigmergy: Experiments and Results*
Gabriel R. C. Jacobsen, Carlos A. Barth e Fernando Dos Santos.
- *Reinforcement learning for route choice in an abstract traffic scenario*
Anderson R. Tavares e Ana L. C. Bazzan.
- *Simulação do Espalhamento da Influenza na Cidade de Cascavel utilizando Agentes Computacionais*
Marcos Paulo Nicoletti, Claudia Brandelero Rizzi e Rogerio Luis Rizzi.
- *TITAN: um jogo de estratégia utilizando conceitos de sistemas multiagentes*
Maicon Rafael Zatelli, José Rodrigo Ferreira Neri, Daniela Maria Uez e Rafael Frizzo Callegaro.

Gostaríamos de agradecer aos palestrantes convidados, Jaime e Olivier, que abrilhantaram o evento com suas palestras. Também agradecemos a todos os pesquisadores que

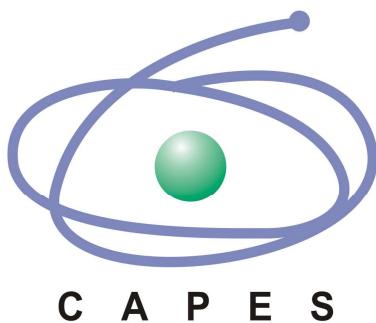
submeteram os seus artigos, assim como aos membros do comitê de programa, aos revisores adicionais pelo criterioso trabalho desenvolvido e às nossas instituições (UFSC e USP). Finalmente, agradecemos ao fomento recebido das agências CNPq, CAPES e FAPESC que tornaram possível o WESAAC 2012.

Florianópolis, Maio, 2012

Jomi F. Hübner

Anarosa Alves Franco Brandão

Patrocínio



Organização

Organização Geral

Jomi Fred Hübner Universidade Federal de Santa Catarina

Coordenação do Comitê de Programa

Anarosa Alves Franco Brandão Universidade de São Paulo

Organização Local

Ricardo Silveira Universidade Federal de Santa Catarina
Jerusa Marchi Universidade Federal de Santa Catarina

Comitê Consultivo

Antônio Carlos da Rocha Costa
Universidade Federal do Rio Grande
Rejane Frozza
Universidade de Santa Cruz do Sul
João Luis Tavares da Silva
Universidade de Caxias do Sul
Diana Francisca Adamatti
Universidade Federal do Rio Grande
Gustavo Gimenez-Lugo
Universidade Federal Tecnológica do Paraná

Comitê de Programa

Adamatti, Diana	FURG
Alencar, Fernanda	Universidade Federal de Pernambuco - UFPE
Alvares, Luis Otavio	Universidade Federal do Rio Grande do Sul
Bassani, Patricia B. Scherer	Universidade Feevale
Bagatini, Daniela	UNISC
Bazzan, Ana L. C.	Universidade Federal do Rio Grande do Sul
Behar, Patrícia Alejandra	Universidade Federal do Rio Grande do Sul
Bercht, Magda	UFRGS
Blois Ribeiro, Marcelo	Pontifical Catholic University of Rio Grande do Sul
Boff, Elisa	Universidade de Caxias do Sul
Bordini, Rafael H.	Federal University of Rio Grande do Sul
Rizzi, Claudia Brandelero	Unioeste
Campos, André	UFRN
Casa, Marcos Eduardo	Universidade de Caxias do Sul
Cazella, Silvio	Unisinos
Choren, Ricardo	Military Institute of Engineering (IME/RJ)
Rocha Costa, Antonio Carlos da	Universidade Federal do Rio Grande
Coutinho, Luciano Dos Reis	UFMA
Silva, Andréa Aparecida Konzen Da	UNISC

David, Nuno	ISCTE
Marchi, Ana Carolina Bertoletti De	UPF
Dimuro, Graçaliz	Universidade Federal do Rio Grande
Ferreira Jr., Paulo Roberto	UFPEL
Flores, Cecilia	UFCSPA
Freddo, Ademir Roberto	Universidade Tecnologica Federal do Parana
Frozza, Rejane	Universidade de Santa Cruz do Sul (UNISC)
Giménez-Lugo, Gustavo	Federal University of Technology-Paraná (UTFPR)
Leite, Joao	Universidade Nova de Lisboa
Lorenzi, Fabiana	Universidade Federal do Rio Grande do Sul and Universidade Luterana do Brasil
Machado, Aydano	UFAL
Marchi, Jerusa	Federal University of Santa Catarina
Moraes, Marcia	PUCRS
Nedel, Luciana	UFRGS
Okuyama, Fabio	IFRS - Campus Porto Alegre
Pereira, Adriana Soares	UFSM
Pimentel, Cesar	UTL - Portugal
Raabe, André	UNIVALI
Rabelo, Ricardo J.	UFSC - Federal University of Santa Catarina
Ribeiro, Alexandre	UCS
Rodrigues, Maira	UFMG
Santos, Elder Rizzon	UFSC
Sichman, Jaime	University of Sao Paulo
Silva, Joao Luis	Universidade de Caxias do Sul
Silveira, Ricardo Azambuja	Universidade Federal de Santa Catarina
Tedesco, Patricia	Center for Informatics / UFPE
Torres Da Silva, Viviane	Universidade Federal Fluminense
Webber, Carine	Universidade de Caxias do Sul

Revisores Adicionais

Aguiar, Marilton Sanchotene de
 Casare, Sara
 Lima, Allan
 Queiroz, Diego
 Rodrigues, Patricia Alves
 Schmitz, Tiago
 Simplicio Jr, Marcos

Sumário

I Palestras Convidadas

Interoperability in Multiagent Systems: Preliminary Results	3
<i>Jaime Simão Sichman</i>	
From Organization Oriented Programming to Multi-Agent Oriented Programming	5
<i>Olivier Boissier</i>	

II Oficinas

Programação Orientada a Agentes	9
<i>João Leite</i>	
Governando Sistemas Multiagentes	11
<i>Viviane Torres da Silva</i>	
Modelagem Orientada a Objetivos e Agentes com o Framework i*	13
<i>Fernanda Alencar</i>	
Aplicações de Simulação Baseada em Agentes	15
<i>Ana Bazzan</i>	
Programação Multiagente	17
<i>Rafael H. Bordini</i>	
RoboCup Rescue Agent Simulation League	19
<i>Paulo Roberto Ferreira Jr, Luis Gustavo Nardin</i>	

III Artigos Completos

Simulação do Espalhamento da Influenza na Cidade de Cascavel-PR Utilizando Agentes Computacionais	23
<i>Marcos Paulo Nicoletti, Claudia Brandelero Rizzi e Rogério Luis Rizzi</i>	
Simulação Multiagente de uma Abordagem Evolutiva e Espacial para o Jogo do Ultimato	35
<i>Luís Felipe Kiesow de Macedo, Murian Dos Reis Ribeiro, Stephanie Loi Brião, Marilton Sanchotene de Aguiar, Graçaliz Pereira Dimuro e Celso Nobre Da Fonseca</i>	
Extending the RoboCup Rescue to Support Stigmergy: Experiments and Results ..	47
<i>Gabriel R. C. Jacobsen, Carlos A. Barth e Fernando Dos Santos</i>	
Simulação Multiagente Interativa no Ambiente SIMULA	57
<i>Marcos Paulo Martins Da Silva, Daniela Duarte Da Silva Bagatini e Rejane Frozza</i>	
Segregação Sócio-espacial: um Estudo utilizando Sistemas Multiagentes	69
<i>Carlos Quadros, Josimara Silveira, Felipe Silva, Leonardo Rodrigues, Liliane Antigueira, Stephanie Brião, Suvania Oliveira e Tauã Cabreira</i>	
Simulando a Execução de Políticas Públicas através de Jason e CArtAgO	81
<i>Iverton A. S. Santos e Antonio C. Rocha Costa</i>	

Modelando a Organização Social de um SMA para Simulação dos Processos de Produção e Gestão Social de um Ecossistema Urbano: o caso da Horta San Jerónimo da cidade de Sevilla, Espanha	93
<i>Flávia Santos, Glenda Dimuro, Thiago Rodrigues, Diana Francisca Adamatti, Graçaliz Dimuro, Esteban de Manuel Jerez e Antônio Carlos Rocha Costa</i>	
TITAN: Um jogo de estratégia simulado utilizando conceitos de sistemas multiagentes	105
<i>Maicon Rafael Zatelli, José Rodrigo Ferreira Neri, Daniela Maria Uez e Rafael Frizzo Callegaro</i>	
A Markovian Multiagent Musical Composer	117
<i>Joel Luis Carbonera e João Luis Tavares Silva</i>	
Extending the Framework TAO with Norms for Multi-Agent Systems	129
<i>Emmanuel Sávio Silva Freire, Mariela Cortés, Enyo Gonçalves e Yrleyjander Salmito</i>	
Reinforcement learning for route choice in an abstract traffic scenario	141
<i>Anderson R. Tavares e Ana L. C. Bazzan</i>	
A Model for Opinion Ranking	155
<i>Allan Lima e Jaime Sichman</i>	
Um estudo sobre alinhamento de ontologias no domínio de reputação de agentes ..	167
<i>Marcos Tae e Anarosa Alves Franco Brandão</i>	
ADAM: An Autonomous Agent for High-Frequency Currency Trading in the Brazilian Market	179
<i>Vicente Matheus Moreira Zuffo e Paulo André Lima de Castro</i>	
Gerenciamento de Agentes Remotos no Projeto Subverse via Scripts embutidos em Linguagem LUA	191
<i>Saulo Popov Zambiasi</i>	

IV Artigos aceitos como pôsteres

Um sistema multiagente para a formação e avaliação de grupos sócio afetivos em ambientes CSCL	205
<i>Alfredo Costa Junior Oliveira, Cícero Costa Quarto, Rômulo Martins França, Luís Carlos Costa Fonseca e Sofiane Labidi</i>	
Cooperative UAVs using multi-agent coordination techniques for search operations	217
<i>Aquila Chaves e Paulo Sérgio Cugnasca</i>	
Ambiente de derivação de sistemas multiagentes industriais apoiados por metodologia e ontologia	229
<i>Vanderlei Weber</i>	
Evolução da Ferramenta MAS-ML tool para a Modelagem do Diagrama de Papéis	241
<i>Francisco R. O. De Lima, Allan R. Feijó, Igor B. Nogueira, Felipe J. A. Maia, Emmanuel Sávio Silva Freire, Mariela Cortés and Enyo Gonçalves</i>	
Planejamento de Rotas de Robôs Móveis: Estudo da Viabilidade de uma Abordagem baseada em Algoritmos Genéticos em um Ambiente Multiagente	253
<i>Tauã M. Cabreira, Marilton S. de Aguiar e Graçaliz P. Dimuro</i>	

Simulação do Trânsito no Centro da Cidade do Rio Grande/RS	265
<i>Josimara de Ávila Silveira, Felipe Neves da Silva e Leonardo Martins Rodrigues</i>	

V Artigos Resumidos

Sistemas Multiagente Plenamente Distribuídos	275
<i>Tiago Mazzutti e Ricardo Azambuja Silveira</i>	
Uso do PopOrg na modelagem de personagens autônomos em jogo com a técnica de interactive storytelling	279
<i>Gleifer Alves, Antonio Carlos Rocha Costa, Raquel Barbosa and Priscilla Gaertner</i>	
Inserindo o Deslocamento de Multidões no Simulador ITSUMO	283
<i>Roger Ferreira da Cruz e Paulo Roberto Ferreira Jr</i>	
Autonomy Development for Unmanned Aerial Vehicles with Jason Agents	287
<i>Marcelo Tomio Hama e Rafael Heitor Bordini</i>	
Sistema multiagentes para indexação e recuperação de objetos de aprendizagem ...	291
<i>Ronaldo Lima Rocha Campos e Ricardo Azambuja Silveira</i>	
Modelo Hospedeiro-Parasitóide Baseado em Sistema Multiagente	297
<i>Érica Lunardi, Aline Brocker, Igor Kimieckiki, Fabio Okuyama and Celson Canto Silva</i>	
An Agent-Based Enrichment System for Genetic Diversity Analyses	301
<i>Giordano Soares-Souza, Guilherme Kingma, Eduardo Tarazona e Maira Rodrigues</i>	
Integrando Agentes Inteligentes e Valor Agregado para o Monitoramento e Controle de Processos de Software	305
<i>Leandro Leocádio Coelho De Souza, Mariela Inés Cortés, Emmanuel Sávio Silva Freire e Gustavo Augusto Lima De Campos</i>	
Uma metodologia para modelagem de sistemas multiagentes	309
<i>Daniela Maria Uez e Jomi Fred Hubner</i>	
Uma nova abordagem para a integração da interação em um sistema multiagente .	313
<i>Maicon Rafael Zatelli e Jomi Fred Hubner</i>	

Parte I

Palestras Convidadas

Interoperability in Multiagent Systems: Preliminary Results

Jaime Simão Sichman

Escola Politécnica da Universidade de São Paulo

jaimie.sichman@poli.usp.br

Abstract. *Multiagent systems have as key concepts the notions of agents, environments, organizations and interactions. Some of these concepts have been inspired by other disciplines, such as social sciences, social psychology, economy and business, among others. As a consequence, none of these concepts have a unique, universally accepted single model, definition or implementation. This issue is more critical when we consider open MAS, when the active entities of the system may enter and leave autonomously whenever they want to. Moreover, these entities may have been designed and/or engineered by different teams, using different architectures. In this talk, we will show how this interoperability problem can be solved, by showing two different approaches based respectively on ontology and model driven engineering techniques.*

From Organization Oriented Programming to Multi-Agent Oriented Programming

Olivier Boissier

EMSE, Saint Etienne, France

olivier.Boissier@emse.fr

Abstract. Social and organizational aspects of agency are a major issue in the Multi-Agent Systems (MAS) domain. Recent applications of MAS on Web and Ambient Computing enforce the need of using these dimensions in the programming of MAS. The aim is to ensure the governance of such systems while preserving their decentralization and openness. In this talk, we present how multi-agent organizations provide first class abstractions, models and tools that contribute to this aim. We focus on the MOISE framework that has been developed these last years. This framework proposes an organization programming language for defining multi-agent organizations that are managed and supported by organizational artifacts at the system execution level and by organization-awareness constructs at the agent programming level. This framework is included in the JaCaMo platform, integration of Jason Agent Programming Language, CarTaGo environment platform and MOISE. We illustrate different features of Organization Oriented Programming of MAS using different examples of developed applications. We will highlight also how it integrates in a broader perspective of multi-agent oriented programming of decentralized and open systems.

Parte II

Oficinas

Programação Orientada a Agentes

João Leite

Universidade Nova de Lisboa - Portugal

jleite@di.fct.unl.pt

Resumo. Com os avanços significativos, nos últimos anos, na área de agentes autónomos e sistemas multi-agente, tecnologias promissoras têm surgido como uma alternativa sensata para o desenvolvimento e engenharia de sistemas multi-agente. O resultado é uma variedade de linguagens de programação, plataformas de execução e ferramentas que facilitam o desenvolvimento e engenharia de sistemas multi-agente. Este curso irá fornecer uma visão geral das linguagens de programação, técnicas e ferramentas que estão actualmente disponíveis para apoiar a implementação eficaz de agentes num sistema multi-agente, dando aos participantes conhecimento de algumas competências básicas no desenvolvimento de agentes para sistemas multi-agente. Será adoptada uma abordagem experimental, através de um conjunto de exercícios laboratoriais, que permita aos participantes a prática das competências adquiridas. Este tutorial está preparado para principiantes na área de programação orientada a agentes, mas requer algumas noções básicas de agentes BDI e programação em lógica.

Governando Sistemas Multiagentes

Viviane Torres da Silva

Universidade Federal Fluminense - Brasil

viviane.silva@ic.uff.br

Resumo. Dentro de sistemas multi-agentes, sistemas de governança são aplicações que possuem o intuito de governar o comportamento dos agentes em sistemas multi-agentes abertos. Os agentes destes sistemas são agentes heterogêneos e tipicamente implementados por diferentes desenvolvedores. Com objetivo de lidar com a diversidade de implementações dos agentes e os possíveis problemas que os diferentes comportamentos podem causar ao sistema devido a autonomia dos agentes, os sistemas de governança definem um conjunto de normas (ou leis) que devem ser seguidas pelas entidades do sistema multi-agentes. Sabendo que os agentes podem cumprir ou não com as normas do sistema, o parceiro de um determinado agente pode avaliar o seu comportamento e compartilhar esta informação com outros agentes. Estes agentes utilizarão a informação sobre a reputação do agente ao escolher os seus futuros parceiros. Os sistemas de Reputação são aplicações que coletam, distribuem e agregam informações sobre o comportamento dos participantes nas interações, i.e., sobre as reputações dos agentes. Esta oficina abordará tanto os sistemas de governança quanto os sistemas de reputação baseados em normas.

Modelagem Orientada a Objetivos e Agentes com o Framework i*

Fernanda Alencar

Universidade Federal de Pernambuco - Brasil

fernandaalenc@gmail.com

Resumo. Hoje, entender o contexto social e organizacional é fundamental para o sucesso de muitos sistemas. O framework i* oferece uma abordagem orientada a agente/metas para a engenharia de requisitos. Através da modelagem explícita e da análise dos relacionamentos estratégicas entre os múltiplos atores, a abordagem incorpora análise social rudimentar em um framework de análise e projeto de sistemas. Atores dependem uns dos outros para os objetivos serem alcançados, as tarefas serem executadas e os recursos serem fornecidos. A noção de "softgoal" é usada para tratar, sistematicamente, com atributos de qualidade ou requisitos não funcionais. Dependências entre os atores dão origem a novas oportunidades, bem como vulnerabilidades. As redes de dependências são analisadas através de um processo de raciocínio qualitativo. Durante a concepção de sistemas, atores exploram configurações alternativas de dependências a fim de avaliar o seu posicionamento estratégico em um contexto multi-agente e social. Este minicurso irá introduzir, explicar e apresentar o framework i* com exemplos, e descrever como utilizá-lo durante as fases iniciais do processo de requisitos.

Aplicações de Simulação Baseada em Agentes

Ana Bazzan

Universidade Federal do Rio Grande do Sul - Brasil

bazzan@inf.ufrgs.br

Resumo. Nesta oficina serão abordados conceitos e aplicações de simulação baseada em agentes. Será feita uma breve revisão bibliográfica sobre os marcos e desafios na área de simulação baseada em agentes. Posteriormente serão discutidas algumas ferramentas e ambientes de simulação pautados neste paradigma. Por fim serão apresentadas aplicações em domínios diversos, com foco em sistemas complexos. Os participantes terão a oportunidade de utilizar uma ferramenta para colocar em prática os conceitos vistos.

Programação Multiagente

Rafael H. Bordini

Pontifícia Universidade Católica do Rio Grande do Sul - Brasil

r.bordini@pucrs.br

Resumo. Neste curso, faremos uma revisão de programação orientada a agente e veremos como esse paradigma foi combinado com programação orientada a organizações multi-agentes e programação orientada a ambientes. O curso foca numa plataforma em particular chamada JaCaMo: uma integração das plataformas Jason (para programação de agentes), Moise (para programação de organizações) e CArtAgO (para programação de ambientes compartilhados). O curso inclui uma sessão de programação em laboratório utilizando a plataforma JaCaMo.

RoboCup Rescue Agent Simulation League

¹Paulo Roberto Ferreira Jr e ²Luis Gustavo Nardin

¹Universidade Federal de Pelotas – RS – Brasil

²Universidade de São Paulo – SP – Brasil

paulo@inf.ufpel.edu.br, gnardin@usp.br

Resumo. O RoboCup Rescue Agent Simulation é uma competição proposta no âmbito da iniciativa internacional RoboCup que tem como objetivos de promover (i) a pesquisa e desenvolvimento de políticas eficientes para a mitigação de danos causados por desastres de larga escala em ambientes urbanos, e (ii) o desenvolvimento de novos simuladores. A competição tem como desafio o desenvolvimento de uma equipe de agentes heterogêneos que interagem diretamente com um ambiente de desastre simulado e com outros agentes a fim de limitar e minimizar os danos causados por tal desastre. Nesse cenário, os agentes possuem conhecimento e comunicação limitados, sendo o principal objetivo a coordenação das ações dos agentes de resgate (bombeiros, ambulâncias e policiais), os quais são responsáveis, por apagar incêndios, resgatar civis soterrados ou feridos, e desbloquear ruas, a fim de reduzir os efeitos do desastre. No âmbito das áreas de Sistemas Multiagentes e Inteligência Artificial, o desenvolvimento de uma equipe de agentes pode envolver diferentes aspectos, tais como: a arquitetura de agentes e suas estratégias comportamentais e sua tomada de decisão, a coordenação e cooperação de agentes explorando os controles centralizado e distribuído, e a comunicação de agentes. Nessa oficina apresentaremos a arquitetura da plataforma de simulação na competição RoboCup Rescue Agent Simulation, bem como a hierarquia de classes que compõem sua arquitetura. Além disso, estudaremos pragmaticamente duas equipes de agentes de resgate que implementam estratégias distintas, as quais possibilitarão o entendimento de aspectos específicos da plataforma de simulação. A primeira equipe implementa uma estratégia gulosa na qual será explorado o uso das ações básicas dos agentes sem envolver a interação entre os mesmos. A segunda equipe implementa uma estratégia que envolve a interação entre os agentes por meio da troca de mensagens em canais de comunicação. Por fim, será apresentado os possíveis usos dessa plataforma na validação de técnicas de Sistemas Multiagentes.

Parte III

Artigos Completos

Simulação do Espalhamento da Influenza na Cidade de Cascavel-PR Utilizando Agentes Computacionais

**Marcos Paulo Nicoletti¹, Claudia Brandelero Rizzi¹,
Rogério Luis Rizzi¹**

¹Centro de Ciências Exatas e Tecnológicas – Universidade Estadual do Oeste do Paraná
Caixa Postal 711 - 85819-110 - Cascavel - PR - Brasil

marcos.nicoletti@unioeste.br, claudia.rizzi@unioeste.br,

rogerio.rizzi@unioeste.br

Abstract. This paper presents and discusses results of the computational experiment in which the spread of influenza was modeled as compartmental models of SIRS type, using the approach of model-based agents. It was simulated the process of spreading the influenza in a specific region of Cascavel-PR, covering part of three districts are geographically close. In this environment, interacted simplified 4653 agents distributed in six distinct types, agents infants, children, teens, adults and seniors. Each agent type has different behaviors according to displacement in the environment for study, work, play or stay home. Disease transmission occurs when the interaction in the same place, among susceptible individuals and infected. The analysis of the tests was made from data obtained in technical literature and has already started work to better characterize and map the city and to validate the model and the results using real data collected by the Secretaria Municipal de Saúde.

Resumo. Este trabalho apresenta e discute resultados decorrentes de um experimento computacional em que o espalhamento da Influenza foi modelado a partir da concepção de modelos compartimentais tipo SIRS, empregando a abordagem de agentes baseados em modelos. Simulou-se o processo de espalhamento da gripe em uma região específica da cidade de Cascavel-PR, abrangendo parte de três bairros. Neste ambiente, interagiram simplificadamente 4.653 agentes distribuídos em 6 tipos distintos, os agentes bebês, crianças, adolescentes, jovens, adultos e idosos. Cada tipo de agente apresenta comportamentos específicos quanto ao deslocamento no ambiente, para fins de estudo, trabalho, lazer ou a estada residencial. A transmissão da doença ocorre quando da interação entre indivíduos suscetíveis e infectados. Análises foram realizadas a partir de resultados obtidos e já foram iniciados trabalhos visando mapear e caracterizar melhor o município bem como validar o modelo através de dados fornecidos pela Secretaria Municipal de Saúde.

1. Introdução

A epidemiologia estuda o processo de saúde e doença em coletividades humanas objetivando analisar a distribuição e os fatores determinantes das enfermidades, danos e eventos associados à saúde visando propor medidas preventivas, de erradicação ou controle, empregando indicadores que dão sustentação ao planejamento, gestão e avaliação de ações

em saúde coletiva [Rouquayrol and Goldbaum 1999]. A epidemiologia se ocupa também em identificar e entender o agente causal e fatores relacionados aos agravos da saúde bem como identificar e explicar os padrões de distribuição geográfica das doenças.

Pesquisas realizadas sobre a transmissão de doenças infectocontagiosas envolvem a compreensão de características complexas incluindo padrões biológicos, ecológicos, geográficos, sociais e epidemiológicos. Alguns aspectos desses estudos podem ser realizados com o emprego da modelagem computacional, de modo a analisar variações e interrelações desses padrões, dentre outras caracterizações, na identificação de cenários que possam subsidiar ações e políticas públicas epidemiológicas.

A especificação de um modelo computacional requer a identificação das principais variáveis envolvidas, bem como a determinação das hipóteses que fundamentam a dinâmica do evento. Nessa especificação devem estar inclusas formulações (físicas, matemáticas, biológicas) apropriadas ao problema que são, então, representadas computacionalmente. Deste processo são obtidas simulações das quais são realizadas análises, previsões que são testadas e comparadas com dados experimentais ou com aqueles disponíveis na literatura técnica. Se o resultado é suficientemente bom para os propósitos determinados, a modelagem é aceita. Caso contrário, ela é modificada e o ciclo se repete.

Uma metodologia empregada para representar a propagação de doenças é utilizar modelos compartimentais. Este tipo de abordagem baseia-se na divisão da população de hospedeiros em categorias, entre as quais os indivíduos fluem com taxas que dependem das características próprias da doença e das formas de transmissão, entre outros fatores. A população hospedeira é subdividida em classes, como a dos indivíduos suscetíveis (S), latentes (E), infectados (I) e removidos (R). Para populações fechadas considera-se que a quantidade indivíduos, N , é tal que $N = S + E + I + R$ [Vynnycky and White 2010].

Um modelo frequentemente empregado para simular a dinâmica do Influenza é o SIRS (Suscetível-Infectado-Recuperado-Suscetível), que é aquele resultante da modelagem de doenças em que os indivíduos infectados podem recuperar-se, porém não adquirem imunidade permanente, passando a ser suscetível após certo período. Exemplos de doenças sob essa categorização são as gripes, que adquiridas num período, conferem imunidade apenas parcial, visto que a população de indivíduos é suscetível (parcial ou totalmente) a uma nova cepa do vírus. Um fluxograma do modelo SIRS com taxas vitais iguais considera as taxas μ , β , α e δ , onde μ indica a taxa de mortalidade e de natalidade constantes em todos estados, β é a taxa de contato entre os indivíduos no estado S e no estado I , α é a taxa de recuperação, quando indivíduos no estado I passam ao estado R e δ é a taxa de reinfecção, quando indivíduos passam do estado R para o estado S .

Para observar o comportamento da dinâmica populacional de indivíduos nessas categorias considerando-se aspectos e comportamentos inerentes às suas classes, deve-se tomar uma modelagem baseada em indivíduos, onde cada um dos elementos componentes da população é considerado. Essa abordagem pode ser fundamentada usando agentes computacionais, cujas ações individuais e autônomas que realizam no ambiente que compartilham, de uma perspectiva sistêmica, interferem no contexto global da dinâmica da transmissão e espalhamento da doença. Através dos agentes é possível especificar e validar, no modelo computacional proposto, as diferentes hipóteses relacionadas aos diversos tipos de agentes, seus atributos, regras de comportamento, interações e seus efeitos sobre

os fatos observáveis no nível macro do sistema [Janssen 2012].

Considerando que a dinâmica da população é influenciada pela rede de contato, é relevante em um modelo baseado em indivíduos, a especificação de uma rede que leve em consideração os relacionamentos e contatos locais e globais que ocorrem em um ambiente. Neste caso, deve-se utilizar uma rede do tipo mundo pequeno (*small world*) [Newman 2000], visto que ela pode ser configurada para estabelecer as conexões entre os vizinhos e aqueles indivíduos que não estão próximos a ele geograficamente mas estão do ponto de vista do contato.

O presente trabalho apresenta resultados decorrentes da modelagem computacional do espalhamento da Influenza em uma região específica da cidade de Cascavel, Paraná, Brasil, empregando modelagem compartmental, agentes computacionais baseados em modelos e redes de contato de mundo pequeno. Os experimentos computacionais aqui apresentados foram realizados considerando dados obtidos na literatura ou aproximados visto que os dados fornecidos pela Prefeitura Municipal de Cascavel ainda estão em processo de organização e tratamento.

2. Agentes baseados em modelos

O modelo de agentes desenvolvido para representar a dinâmica da Influenza é aquele baseado em modelo que deve, a partir de seu estado interno, interagir com o ambiente, de modo que a percepção atual é combinada com esse estado interno para gerar uma descrição atualizada do estado atual. Cada agente implementa as operações de atividades por faixa etária, de transição de estados e de conecto-mobilidade.

O operador de faixas etárias realiza operações lógicas buscando avaliar os possíveis estados que um agente pode assumir, tomando-se as distintas possibilidades de realização de atividades em determinadas localidades considerando-se as diferentes faixas etárias do agente, sendo que assume uma única faixa etária ao longo da evolução. O operador de transição de estados realiza os contatos entre agentes, que podem estar em distintos estados, realizando ou não a transição de estado quando as taxas de infecção ou de recuperação forem adequadas. O operador de conecto-mobilidade distribui o agente para os nodos pelas arestas, considerando-se aspectos de conectividade e mobilidade.

A operação de composição entre esses três operadores produz o operador de evolução global do estado no tempo do agente, quando interagindo com o ambiente. Tal operador em geral não é bijetivo, pois é resultado de relações entre distintos atributos e, portanto, geralmente não é reversível. Para realizar uma simulação, é necessário configurar certos parâmetros que estabelecem quais as condições iniciais do ambiente e do modelo interno do agente. Os principais são:

- Dados topológicos: que consistem na identificação de nodos, arestas, conectividade e mobilidade que são informações necessárias à determinação de quaisquer tipos de localidades e para estabelecer a conecto-mobilidade aos agentes para transportá-los às vizinhanças geográficas e lógicas.
- Dados de atividades por faixa etária: que consistem na especificação do comportamento do agente, estruturado por faixa etária, que atua diferentemente de acordo com seu posicionamento em determinadas localidades e período de tempo.
- Dados demográficos: que consistem na população estruturada por faixa etária. A

- distribuição considera o tipo de localidade onde os agentes atuarão, respeitando-se os parâmetros da densidade média da população bem como o período de tempo.
- Dados epidemiológicos: que consistem na especificação das taxas de contato efetivo, de recuperação e de perda de imunidade, definindo a ocorrência ou não da transição entre os estados suscetível, infectante e removido.
 - População inicial: que consiste na informação inicial das quantidade ou proporções de agentes nos estados suscetível, infectante e removido.

Considerando que um determinado agente encontra-se numa faixa etária numa posição geográfica em um subpasso de tempo, o operador de evolução global realiza as transições para cada passo discreto de tempo. Um cenário na simulação é o estado final de uma evolução.

3. Parâmetros geográficos e dinâmicos

Cascavel é uma cidade situada no oeste do estado do Paraná. Possui 2.091 km² de área territorial e dista 490 km da capital. Seus 286.205 habitantes residem em 100.931 domicílios. Entre creches, escolas de ensino pré-escolar, fundamental e médio, existem 194 estabelecimentos oficialmente registrados. São 8 as instituições de ensino superior. A população economicamente ativa se ocupa de atividades ligadas ao comércio, indústria de transformação, agricultura, pecuária, construção civil e prestação de serviços. São aproximadamente, 115.000 indivíduos que trabalham na zona urbana. Há 9.000 estabelecimentos que realizam atividades econômicas. A densidade demográfica é de 138,35 hab/km², o grau de urbanização é de 94,36% e o índice de desenvolvimento humano (IDH-M) é de 2.000 [Ipardes 2012], [Ibge 2011].

Em 2009, ano em que se iniciou o registro da Influenza como doença com notificação compulsória, Cascavel teve 5.089 casos notificados e 4.081 casos confirmados da doença. Tratou-se de uma pandemia que preocupou a população, exigiu intervenção específica por parte da vigilância epidemiológica do município e motivou o presente trabalho de modelagem e simulação. Embora a doença tenha acometido indivíduos espalhados em todo o município, optou-se por realizar um primeiro experimento envolvendo uma parte geográfica da cidade, o centro dela, onde residem e circulam grande quantidade de pessoas por dia. Esta área, que abrange parte de 3 bairros, é ilustrada pela figura 1, onde se destacam as 60 quadras consideradas no experimento e seu arruamento.

Nessa região foram mapeados 1095 pontos, sendo 817 considerados residências e 277 locais de interesse envolvendo estabelecimentos de estudo, trabalho e lazer. A figura 2 quantifica e qualifica os tipos dos 277 locais de interesse situados geograficamente na região ilustrada na figura 1, e define a quantidade máxima de agentes que podem ali estar no mesmo período de tempo (a capacidade). Em cada uma das 817 residências, no máximo 6 agentes podem ali permanecer ao mesmo tempo. A quantidade de residências é determinada pela diferença entre os pontos de interesse e o total de nodos.

Para a caracterização dos agentes envolvidos na simulação especificou-se 6 tipos distintos: agentes bebês, crianças, adolescentes, jovens, adultos e idosos. Considerando que o quantidade total de habitantes dos três bairros envolvidos na simulação, segundo a sinopse decorrente do censo 2010 fornecida pelo Instituto Brasileiro de Geografia e Estatística (IBGE), era de 46.533 habitantes [Ibge 2011], optou-se por quantificar, para cada tipo distinto de agente, o percentual de 1/10 do total de indivíduos categorizados

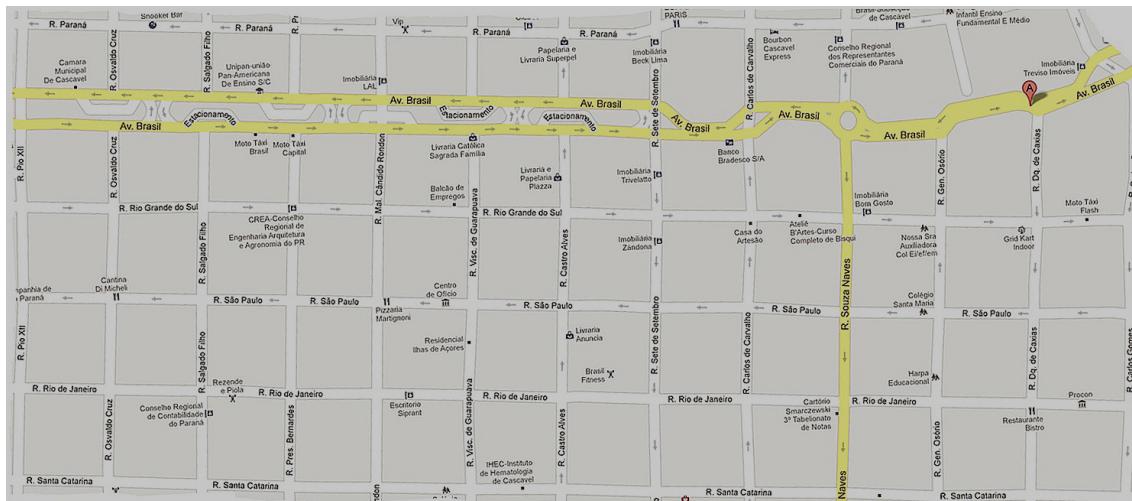


Figura 1. Recorte geográfico de Cascavel selecionado para a simulação.

Locais de Estudo			Locais de Trabalho			Locais de Lazer		
Qtd	Tipo	Cap.	Qtd	Tipo	Cap.	Qtd	Tipo	Cap.
1	Universidades	800	6	Instituições públicas	10	3	restaurantes	30
2	Colégios	350	5	Imobiliárias	5	1	Academia	28
2	Creches	40	4	Livrarias	8	2	Igrejas	60
			7	Escritórios	12	2	Praças	40
			1	Laboratório	78	4	Bares	20
			3	Bancos	40			
			2	Hotéis	8			
			3	Farmácias	6			
			69	Comércio médio	20			
			160	Comércio pequeno	6			

Figura 2. Os 277 locais de interesse mapeados: estudo, trabalho e lazer.

entre os 6 tipos possíveis. Assim, para a simulação apresentada neste trabalho, foram utilizados 4.653 agentes distribuídos entre os 6 tipos distintos.

Agentes reativos baseados em modelos empregados neste trabalho mantêm um estado interno para aspectos não percebidos no ambiente a partir do período do dia (manhã, tarde, noite ou madrugada), de suas necessidades de movimentação decorrentes de sua faixa etária (residência, estudo, trabalho e lazer) e de sua condição de saúde (suscetível, infectado, removido). A decisão para a movimentação do agente leva em conta o período do dia (manhã, tarde, noite, madrugada) e se o local para o qual foi previamente escolhido comporta sua estada, ou seja, se não atingiu a quantidade máxima de agentes que comporta. A figura 3 mostra o relacionamento entre faixa etária × período × locais de permanência, de cada tipo de agente em cada período do dia.

Visando estimar a expectativa da quantidade aproximada de agentes a se deslocarem por período de tempo, foram feitas as seguintes considerações, das quais decorrem os valores apresentados na figura 3.

- A taxa de freqüência às creches municipais do Município de Cascavel em 2011 é

Agente (tipo)	Idade (anos)	Manhã	Tarde	Noite	Madrugada
Bebês Total = 268	Menores de 6	Residência: 188 ou Creche: 80	Residência: 188 ou Creche: 80	Residência: 268	Residência: 268
Crianças Total = 477	De 6 até menores que 14	Residência: 24 ou Estudo: 453	Estudo: 453 ou Lazer: 24	Residência: 477	Residência: 477
Adolescentes Total = 222	De 14 até menores de 18	Estudo: 211 ou Residência: 11	Estudo: 211 ou Lazer: 11	Residência: 222	Residência: 222
Jovens Total = 1129	De 18 até menores de 30	Estudo: 226 ou Trabalho: 903	Estudo: 226 ou Trabalho: 903	Estudo: 226 ou Residência: 903	Residência: 1073 ou Lazer: 56
Adultos Total = 1990	De 30 até menores de 60	Trabalho: 1871 ou Residência: 119	Trabalho: 1871 ou Residência: 119	Residência: 1592 ou Lazer: 398	Residência: 1891 ou Lazer: 99
Idosos Total = 567	Maiores de 60	Residência: 567	Residência: 482 ou Lazer: 85	Residência: 567	Residência: 567

Figura 3. Relacionamentos faixa etária × período × locais de permanência.

de 30% das crianças menores de 6 anos, calculadas a partir dos dados do Censo de 2010. Neste caso, no máximo 80 agentes bebês poderiam estar nas creches nos períodos da manhã e tarde.

- Pelas informações obtidas na documentação ENAD 2010, 95% das crianças e adolescentes na faixa etária correta frequentam às escolas. Assim, 453 agentes crianças e 211 agentes adolescentes devem estar presentes nas Instituições de Ensino, nos períodos adequados. Para as atividades de Lazer, no período da tarde, foram considerados 5% do total de cada um dos tipos destes agentes.
- Segundo o IBGE [Ibge 2010], no Paraná, 19,9% dos jovens estudam nos períodos da manhã, tarde ou noite. Isso fornece 226 agentes nesta faixa etária. Além disso, considerou-se que 80% dos agentes jovens trabalham nestes períodos. No período de madrugada, para as atividades de lazer, apenas 5% deles estão em tal atividade.
- Pelas informações divulgadas [MDIC 2012], 6% dos adultos em 2011 estavam desempregados. Tomando essa proporcionalidade, considerou-se que 1871 agentes adultos trabalham no período da manhã ou tarde. À noite considerou-se que 80% deles estão estudando e 20% estão em atividades de lazer. Na madrugada, 95% estão em suas residências e 5% em atividades de lazer.
- Quanto aos agentes idosos, considerou-se que na parte da tarde, 5% estão em atividades de lazer.

4. Ambiente computacional

A implementação computacional do ambiente de simulação utilizado neste trabalho foi desenvolvida na linguagem Java e utiliza o framework Java Agente DEvelopment Framework (JADE) [Bellifemine et al. 2012] para a manipulação de agentes. A biblioteca JFreeChart [Gilbert 2007] foi utilizada para geração de gráficos.

A simulação desenvolvida emprega modelos compartimentais para a modelagem da dinâmica da Influenza, agentes reativos baseados em modelos para a modelagem dos agentes computacionais e redes de mundo pequeno (*small world*) para representar redes de contato entre indivíduos. A dinâmica espaço-temporal das interações entre os

indivíduos ocorre em locais residenciais, de estudo, trabalho e lazer e são estruturadas por faixas etárias compreendendo bebês, crianças, adolescentes, jovens, adultos e idosos, por estado do tempo em um dia, podendo estar no período da manhã, tarde, noite e madrugada (sumarizado na figura 3) e por estado, suscetível, infectado ou removido.

Para a modelagem computacional foram definidas três classes de agentes: o agente Timer, os agentes Compartimentais e os agentes Humanos, sendo que apenas esses últimos são modelados como sendo do tipo reativos baseados em modelos. Há um único agente Timer, três agentes Compartimentais e 4.653 agentes Humanos. Tanto o agente Timer quanto os três agentes Compartimentais atuam em uma estrutura de contêiner. O agente Timer faz contagem dos dias. Ele também contém informação temporal que permite que todos os agentes Humanos saibam sobre a mudança dos períodos de tempo no ambiente (manhã, tarde, noite e madrugada).

Os agentes Compartimentais, mais especificamente o Agente Compartimental Suscetível, o Agente Compartimental Infectado e o Agente Compartimental Removido efetuam a contagem da quantidade de agentes Humanos que estão em cada um dos três compartimentos. Também são responsáveis por informar ao agente Timer quando cada um dos agentes Humanos concluiu as atividades de seu dia.

Os 4.653 agentes Humanos assumem três estados em um dado momento. Um estado indica sua faixa etária: bebê, criança, adolescente, jovem, adulto e idoso. Outro estado indica seu local de atuação, ou seja, em sua residência ou em atividades de trabalho, estudo ou lazer. O terceiro estado indica o estado compartmental em que se encontra: suscetível, infectado ou removido. Cada um dos agentes mantém informações sobre esses três estados internos que modelam o mundo da simulação e que contribuem para a tomada de decisão a partir de suas percepções sobre o ambiente e sobre eles mesmos.

Assim, os agentes Humanos, cientes de sua condição em termos de faixa etária (bebê, criança, adolescente, jovem, adulto e idoso), compromissos (em casa, no trabalho, estudando ou se divertindo) assumidos durante os quatro estados possíveis de um dia (manhã, tarde, noite ou madrugada) e condição de saúde (suscetível, infectado ou removido), se deslocam para um dos possíveis locais onde podem ou devem estar. Um caso típico é aquele em que um agente jovem saudável, pela manhã ou tarde está estudando ou trabalhando. De noite está estudando ou em sua residência. De madrugada, está em sua residência ou em atividades de lazer (figura 3). Embora essa seja sua rotina, ela pode ser modificada na dinâmica do dia e esse agente por ir/estar em outro local como, por exemplo, ter se deslocado para uma creche. Como sua presença ali é eventual, sua contribuição para o espalhamento da Influenza é nula. Se esse agente estiver em seu local de estudo ou trabalho ou lazer, sua contribuição para o espalhamento da Influenza é máxima.

A topologia do ambiente é definida por uma estrutura contendo 60 clusters. Cada cluster representa uma das 60 quadras componentes de partes de três bairros do município de Cascavel, que foram tomadas para comporem o ambiente da simulação e são ilustradas na figura 1. Cada cluster é formado por uma estrutura contendo um grafo não direcionado constituído por n nodos, sendo que essa quantidade varia em cada nodo já que representa a quantidade de locais ali existentes e sua densidade demográfica, tomando como parâmetros as informações das unidades censitárias do IBGE, do censo de 2010.

Nessa estrutura estão representados os 277 locais de estudo, trabalho e lazer e

817 locais residenciais. Cabe destacar que os nodos que representam os locais de estudo, trabalho e lazer foram posicionados nos clusters que representam as quadras em que estão situados fisicamente em Cascavel.

Os agentes se movimentam pelos nodos apenas através das arestas que os ligam. Se as arestas ligam os diversos nodos que compõem o cluster elas são denominadas internas. As arestas que fazem as ligações entre clusters são denominadas externas. A estrutura de conexões (clusters e nodos) pode ser modificada permitindo a inclusão de rede de mundo pequeno (*small world*) através da inserção de arestas na lista de conectividade. As arestas podem estar geograficamente distantes, mas estarão logicamente conectadas, como pertencendo à estrutura de vizinhança local. Quando utilizada, a rede de mundo pequeno pode ser do tipo interna, quando arestas são inseridas internamente em um cluster, ou externa, quando arestas são inseridas de maneira que ligam clusters diferentes.

Para isso, seleciona-se no ambiente de simulação, um valor de até 5% [Newman 2000] que informa o percentual de arestas que serão incluídas na estrutura original. Este é o caso em que, por exemplo, um agente infectado que reside no lado leste da região mapeada, tem contato direto com outro agente, que mora na região oeste e está suscetível, visto que atua naquela região.

Ainda sobre a movimentação dos agentes, se não há ligação entre um nodo e outro, o agente não pode movimentar-se entre eles. Também não pode movimentar-se para um nodo quando o mesmo estiver com sua capacidade esgotada. A capacidade do nodo é determinada no início da simulação quando é definida a quantidade máxima de agentes que cada nodo comporta. Quando a capacidade do nodo se esgota, os agentes não podem ali entrar e selecionam outro nodo para visitar. Caso não encontrem na sua adjacência um nodo com capacidade disponível, se manterão no local onde estavam.

4.1. O simulador

A figura 4 ilustra o ambiente de simulação. Sua interface se divide em duas partes: a Tela principal e a Barra de ajustes de parâmetros.

Na Tela principal é apresentado o mapa do local representado com seu respectivo grafo. Os vértices são representados com cores diferenciadas para cada estado do local. A escolha pela representação ocorre da seguinte maneira. Se no local existem agentes infectados o vértice será vermelho. Se não existem agentes infectados e a quantidade de agentes suscetíveis é maior que o quantidade de agentes recuperados, o vértice será azul. Se não existem agentes infectados e o quantidade de agentes recuperados é maior que o quantidade de agentes suscetíveis, então o vértice será verde. Se não existir nenhum agente no vértice ele será preto. Na Barra de ajustes de parâmetros há os seguintes conjuntos de parâmetros: Tamanho da população, Parâmetros do patógeno, dias da simulação, Grau de conectividade da rede de mundo pequeno, Evolução dos dias, e o botão de Confirmação de parâmetros.

Na seção Tamanho da população é definida a quantidade total de agentes em cada uma das faixas etárias consideradas: bebês, crianças, adolescentes, jovens, adultos e idosos, com a respectiva quantidade total de agentes infectados para cada uma dessas faixas no início da simulação. Na seção Parâmetros do patógeno, neste caso a Influenza, são definidos três parâmetros: a taxa de transmissão do vírus; a taxa de recuperação do agente, que depende da característica da doença, e que é calculada em termos de dias de simulação;

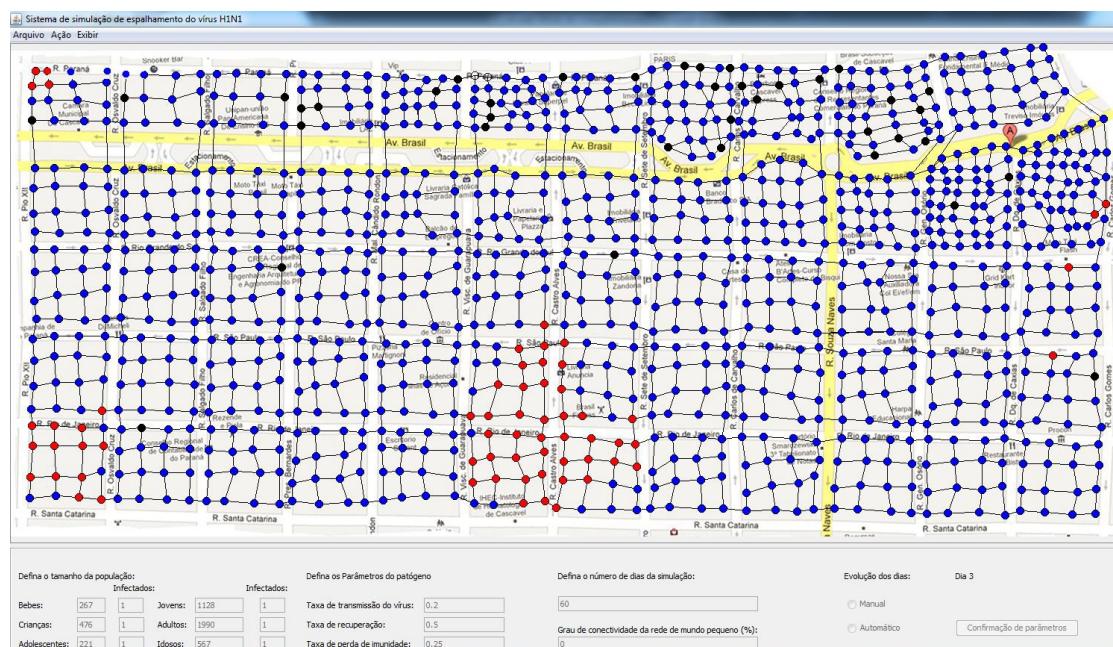


Figura 4. Tela principal do ambiente de simulação.

a taxa da perda de imunidade, que também depende das característica da doença, e que também é calculada em termos de ciclos de tempo da simulaçao. Esses parâmetros foram selecionados para este experimento tendo orientação fornecida pela Vigilância Epidemiológica do município de Cascavel. Na seção Número de dias da simulação é definida a quantidade de dias que a simulação deverá considerar. Na seção Grau de conectividade da rede de mundo pequeno, é definida a porcentagem de nodos que terão arestas incluídas. Para este experimento tomou-se 3% com inclusão de arestas internas e externas. Na sessão Evolução dos dias é apresentada, sequencialmente, a quantidade de dias da simulação. O botão de Confirmação de parâmetros confirma todos os parâmetros informados.

5. Resultados e discussões

No modelo de simulação utilizado não há nascimentos nem mortes, isto é, a população é constante, sendo conservativo o modelo. A simulação ocorre da seguinte maneira. É definido o grafo não direcionado sobre o qual será realizada a simulação e que representa os clusters e seus nodos. Os agentes são iniciados. É feita a distribuição da população estruturada por faixa etária. A distribuição inicia alocando cada agente em uma residência. Cada agente sabe sua faixa etária e em decorrência dela, onde é seu local de residência, seu local de trabalho, estudo e de lazer. No início do primeiro dia, os agentes aguardam pela informação de que o período da manhã já se iniciou. A partir deste ponto e até o final da simulação, a execução ocorre ciclicamente, da seguinte maneira:

- O agente sorteia um vértice para visitar. Caso o vértice já esteja com sua capacidade esgotada, procurará outro vértice. Caso não encontre em sua adjacência, nenhum vértice que possa visitar ficará no local onde estava.
- O agente atualiza seu estado compartmental de acordo com a regra de transição do modelo SIRS, procedendo da seguinte maneira. Depois que ele se desloca ou não para outro ponto do grafo, levando em conta seu estado atual (suscetível,

infectado ou removido) é feita a atualização para seu novo estado. Por exemplo, se o agente está em uma posição válida (a exemplo de um agente jovem em um local de trabalho no período da manhã) e se está no estado suscetível, é feita uma operação considerando o fator de probabilidade de transmissão informado no início da simulação (por exemplo 0,98). Se o resultado desta operação for um valor maior que o fator de transmissão, é modificado o estado do agente, de suscetível para infectado.

- Após atualizar seu estado compartmental, o agente aguarda pela informação de que houve mudança no período de tempo no ambiente (por exemplo, de manhã para tarde) para deslocar-se novamente.
- Ao final do dia, com a passagem dos quatro períodos, o agente informa ao agente Compartimental o seu estado. O agente Compartimental incrementa seu contador da quantidade de agentes que estão no estado que representa e informa ao agente Timer que um agente terminou sua execução do dia.

Foram realizados vários experimentos com diversas variações de parâmetros utilizando o ambiente de simulação implementado. Dois são apresentados a seguir e objetivam mostrar que o modelo proposto submetido ao recorte geoespacial da cidade de Cascavel, apresenta desempenho interessante do ponto de vista da compreensão dos diversos elementos necessários à realização de simulações do espalhamento da doença e das adequações que precisam ser implementadas visando retratar um cenário mais fidedigno. A figura 5 apresenta os parâmetros utilizados em ambos os experimentos.

Parâmetros	Teste I	Teste II
Bebês infectados	0	0
Crianças infectadas	0	0
Adolescentes infectados	10	0
Jovens infectados	5	10
Adultos infectados	5	5
Idosos infectados	0	0
Taxa de infecção	0,4	0,2
Taxa de recuperação	0,02	0,02
Taxa de perda de imunidade	0,25	0,25

Figura 5. Parâmetros utilizados nos testes I e II apresentados.

A figura 6 mostra o gráfico do espalhamento da Influenza entre a população de 4653 agentes no período de 365 dias, com os parâmetros apresentados na figura 5, Teste I. Nota-se que tendo início com 20 indivíduos infectados entre adolescentes, jovens e adultos e uma taxa de infecção de 0,4, nos primeiros 30 dias, a doença já havia atingido uma população de 3.000 agentes. Em 60 dias, o quantidade de infectados havia caído para próximo de zero, visto que a infecção não foi sustentada pelas condições intrínseca aos dados topológicos e dinâmicos empregados. Passados 250 dias da primeira infecção, a população de indivíduos já estava suscetível a uma nova infecção. Pode-se inferir que, supondo-se que a infecção ocorreu no inverno no primeiro ano, no inverno do ano seguinte uma nova infecção poderia ser esperada.

A figura 7 mostra o gráfico do espalhamento da Influenza entre a população de 4653 agentes no período de 365 dias, com os parâmetros apresentados na figura 5, Teste

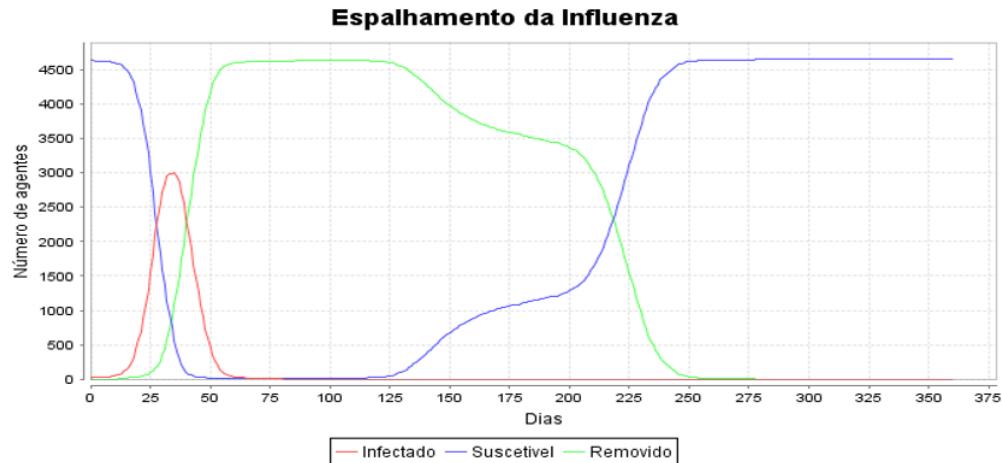


Figura 6. Espalhamento da Influenza entre a população: Teste I.

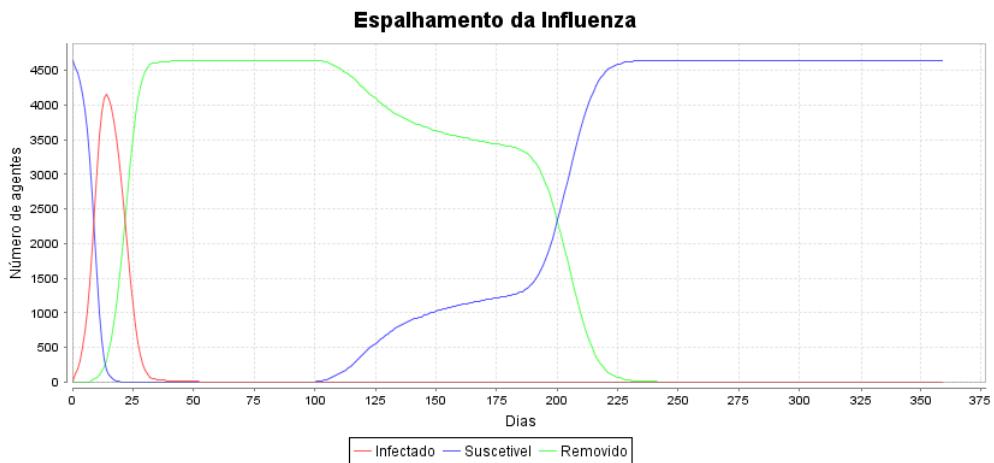


Figura 7. Espalhamento da Influenza entre a população: Teste II.

II. A figura 7 mostra que, tendo início com 15 indivíduos infectados entre adolescentes e adultos e uma taxa de infecção de 0,2, metade da força de infecção do teste I, nos primeiros 15 dias, a Influenza já havia atingido quase que a totalidade da população de agentes. Pode-se inferir que a conectividade dos agentes, tal como o que se observa na realidade, tem uma influência importante no espalhamento da doença. Assim as relações de contato influenciam de modo importante a propagação de uma doença de contato direto, como a simulada.

6. Considerações finais

Neste trabalho foram apresentados resultados obtidos a partir de um experimento computacional em que o espalhamento da Influenza em parte de três bairros da cidade de Cascavel-PR, utilizando parâmetros fictícios, foi modelado utilizando modelos compartimentais tipo SIRS, agentes reativos baseados em modelos e redes de mundo pequeno. Na modelagem, procurou-se contemplar características importantes do ambiente e da doença a fim de obter resultados condizentes com o contexto real e a literatura técnica da área. Os resultados obtidos permitiram não apenas entender melhor a dinâmica a ser modelada

como outros requisitos que precisam ser implementados visando dispor de uma ferramenta de simulação mais eficiente e robusta.

O trabalho apresentado, constitui a primeira aproximação no sentido do desenvolvimento de um projeto maior que prevê uma modelagem mais adequada considerando a ambientação de toda a cidade de Cascavel. Para isso, alguns dos principais aspectos que estão sendo trabalhados tendo como elemento motivador os resultados do experimento apresentado são: a especificação de um ambiente de simulação a partir da manipulação de mapas do tipo *shape*, fornecidos pelo IBGE, decorrentes do censo 2010, que inclui unidades censitárias e separação da população por faixas; a melhoria do desempenho computacional considerando possibilidades de paralelizar o modelo, incluindo melhor exploração do potencial do uso do Jade; a exploração mais adequada da utilização técnica das redes de mundo pequeno; a melhoria de recursos de visualização de resultados e a organização dos dados da Influenza de Cascavel referentes aos anos de 2009/2011 visando a calibração e validação do modelo.

O emprego de agentes baseado em modelos se mostrou apropriado para os propósitos deste projeto e sua utilização para a próxima fase desta pesquisa tem requerido da equipe refinamentos em termos da especificação teórica da modelagem dos agentes. Um trabalho específico sobre este tema está em fase de preparação para publicação.

Referências

- Bellifemine, F. L., Caire, G., and Greenwood, D. (2012). Developing multi-agent system with jade. <http://jade.tilab.com/>.
- Gilbert, D. (2007). The jfreechart class library, developer guide. <http://www.jfree.org/jfreechart/>.
- Ibge (2010). Instituto brasileiro de geografia e estatística. síntese de indicadores sociais, uma análise das condições de vida da população brasileira, 2010. <http://www.ibge.gov.br/home/estatistica/>.
- Ibge (2011). Instituto brasileiro de geografia e estatística. sinopse do censo demográfico 2010 paraná. <http://www.censo2010.ibge.gov.br/sinopse/>.
- Ipardes (2012). Instituto paranaense de desenvolvimento econômico e social. caderno estatístico município de cascavel. <http://www.ipardes.gov.br/>.
- Janssen, M. A. (2012). Agent-based modelling. International Society for Ecological Economics. <http://www.ecoeco.org/>.
- MDIC (2012). <http://www.brasilmaior.mdic.gov.br/noticias/>.
- Newman, M. (2000). Models of the small-world. *Journal of Statistical Physics*, vol.101, pp. 819-841.
- Rouquayrol, M. Z. and Goldbaum, M. (1999). Epidemiologia, história natural e prevenção de doenças. In Rouquayrol, M. and Filho, A., editors, *Epidemiologia e Saúde*. MEDSI, Rio de Janeiro.
- Vynnycky, E. and White, R. G. (2010). *An Introduction to Infectious Disease Modelling*. Oxford University Press.

Simulação Multiagente de uma Abordagem Evolutiva e Espacial para o Jogo do Ultimato

**Luís Felipe K. Macedo¹, Murian dos R. Ribeiro³, Stephanie L. Brião¹
Celso N. da Fonseca¹, Marilton S. de Aguiar^{1,3}, Graçaliz P. Dimuro^{1,2}**

¹ Programa de Pós-Graduação em Modelagem Computacional (PPGMC)

²Programa de Pós-Graduação em Computação (PPGComp)

Universidade Federal do Rio Grande (FURG)

³Programa de Pós-Graduação em Computação (PPGC)

Universidade Federal de Pelotas (UFPEL)

{mdrribeiro, marilton}@inf.ufpel.edu.br

{felipe.lemad, tephylois88, celsonf, gracaliz}@gmail.com

Abstract. This work presents a multiagent simulation for a spatial and evolutionary approach of the Ultimatum Game, implemented using NetLogo. We consider a heterogenous agent population with incomplete information related to the other agents' strategies. For the agent strategy evolution, aiming at the self-regulation of the exchanges allowed by the game, balancing individual and collective goals, we use a genetic algorithm. Results of the simulations performed are presented, with a comparison of the results obtained using two different scenarios.

Resumo. Este trabalho apresenta uma simulação multiagente para uma abordagem evolucionária espacial do Jogo do Ultimato, implementada no software NetLogo. Considera-se uma população heterogênea de agentes com informação incompleta sobre as estratégias de outros agentes. Para a evolução das estratégias dos agentes, com o objetivo de auto regular as trocas proporcionadas pelo jogo, balanceando objetivos individuais e coletivos, é utilizado um algoritmo genético. Apresentam-se os resultados das simulações realizadas, com a comparação dos resultados em dois cenários diferentes.

1. Introdução

A teoria dos jogos [von Neumann and Morgenstern 1944] ajuda a entender teoricamente o processo de decisão de agentes que interagem entre si em situações estratégicas, a partir da compreensão da lógica da situação em que estão envolvidos. Considera-se que os agentes fazem suas escolhas de forma racional. [Fiani 2006]

No entanto, a teoria dos jogos clássica não tem sido suficiente para explicar o comportamento humano observado em diversas situações, como nos sistemas sociais. Para resolver esse problema, uma série de pesquisadores se desviaram do paradigma de auto-estima e escolha racional, adotando a teoria da preferência social. [Camerer et al. 2001]

Este artigo, inspirado no trabalho de Xianyu [Xianyu 2010], apresenta uma abordagem evolucionária e espacial para o Jogo do Ultimato, para estudar o comportamento de trocas econômicas entre uma população heterogênea de agentes. Os jogadores, que

podem ter diferentes preferências sociais, não possuem informações das recompensas dos outros jogadores, não conhecem o perfil de seus oponentes, o que caracteriza um jogo de informação incompleta.

O objetivo principal é alcançar a auto-regulação das trocas entre os agentes, maximizando o número de trocas, com o balanceamento dos objetivos individuais e coletivos. Os agentes, embora possuam autonomia para alcançar seus objetivos individuais, são motivados a realizar trocas com outros agentes. Utilizando um algoritmo genético, os agentes evoluem suas estratégias de trocas, no sentido de aumentar consideravelmente o número de trocas durante as várias etapas do jogo e, consequentemente, aumentar suas utilidades.

Com a intenção de obter simulações compatíveis com o comportamento de sistemas sociais, é introduzido nos agentes um conjunto pequeno de preferências sociais: (i) agentes que resistem a resultados desiguais, *avessos à desigualdade*, dispostos a abrir mão de alguma recompensa material para que as trocas realizadas sejam mais justas; (ii) agentes que possuem um valor *mínimo aceitável* durante as trocas; (iii) agentes que desejam o *bem estar social* da comunidade.

O artigo está organizado como descrito a seguir. Na Seção 2, apresentam-se conceitos relacionados ao Jogo do Ultimato. O modelo evolucionário e espacial do Jogo do Ultimato para regulação de trocas econômicas é discutido na Seção 3. Os resultados das simulações realizadas, em dois diferentes cenários, são apresentados na Seção 4. A Seção 5 é a Conclusão.

2. O Jogo do Ultimato: no sentido de uma abordagem espacial e evolucionária

Na forma mais simples do jogo do ultimato, dois jogadores determinam como dividir entre eles um dado valor econômico. O primeiro jogador, o **proponente**, propõe a forma como deve ser dividido o valor, e o segundo jogador, o **respondente**, avalia a proposta. Se o respondente aceitar a oferta, o dinheiro é dividido de acordo com a proposta feita pelo proponente, caso contrário, ambos jogadores não recebem nada. Os jogadores são esclarecidos que só participarão do jogo uma única vez e que não é possível barganhar, ou seja, uma vez feita a oferta pelo proponente, cabe ao respondente dizer se aceita ou não.

Baseado na teoria dos jogos clássica, se o segundo jogador for racional, ele irá preferir receber qualquer valor, mesmo que este valor seja muito baixo, pois receber pouco é melhor do que não receber nada. Como o proponente sabe que qualquer valor proposto ao segundo jogador será aceito, mesmo que este valor seja muito baixo, reservará para si o maior ganho possível e, consequentemente, deixará o menor valor possível para o segundo jogador. Esta é uma solução racional dada pelo equilíbrio de Nash¹. [Xianyu 2010, Kellermann 2008, Fiani 2006]

No entanto, os resultados de experiências realizadas com grupos de pessoas jogando o ultimato contrariam a solução racional. Experimentos do jogo do ultimato realizados em 25 países diferentes mostram que a maioria dos proponentes fazem uma oferta justa (até 80% deles oferecem de 40% a 50% do total) e mais da metade dos que respondem rejeitam ofertas menores que 30% do valor total [Oosterbeek et al. 2004,

¹Uma combinação de estratégias é equilíbrio de Nash se a estratégia de um jogador é a melhor resposta para as estratégias escolhidas pelos outros jogadores, e isto é válido para todos os jogadores.

Nowak et al. 2000, Page et al. 2000]. Este comportamento dos humanos é considerado irracional comparado a racionalidade teoricamente proposta pela teoria dos jogos.

Observa-se ainda que se o mesmo jogo for realizado várias vezes entre as mesmas pessoas o resultado tenderá para ofertas mais justas, já que o jogador que responde poderá rejeitar ofertas menores com o intuito de obter melhores ofertas em rodadas posteriores [Page et al. 2000].

Assim, outros fatores são importantes na análise do jogo do ultimato, tais como suas preferências sociais. [Camerer et al. 2001, Xianyu 2010]

Salienta-se também a estrutura espacial e/ou de rede para o modelo de interação de agentes é outro fator importante que influencia o resultado de jogos espaciais (i.e., quando consideram-se $n > 2$ jogadores). [Xianyu 2010, Lieberman et al. 2005, Szaba and Fáth 2007]

Para ciência social, a teoria dos jogos evolucionários pode descrever e prever com maior sucesso as escolhas dos seres humanos, uma vez que está melhor equipada para lidar com as apropriadas suposições mais fracas de racionalidade.

A teoria dos jogos evolucionários tem raízes biológicas que derivam de três fatores. Primeiro, a evolução tratada por esta teoria não precisa ser a evolução biológica. Evolução, neste contexto, pode muitas vezes ser entendida como evolução cultural, onde este se refere a mudanças nas crenças e normas ao longo do tempo. Em segundo lugar, a teoria dos jogos evolucionários utiliza um conceito de adaptabilidade ou sucesso reprodutivo diferentemente do conceito de racionalidade da teoria dos jogos. Este conceito mais fraco de racionalidade, em muitos casos, é mais apropriado para a modelagem dos sistemas sociais do que a suposição de racionalidade da teoria dos jogos tradicional. Em terceiro lugar, a teoria dos jogos evolutivos, como uma teoria explicitamente dinâmica, permite a evolução das estratégias dos agentes ao longo do tempo. A teoria dos jogos evolucionários tem sido usada para explicar muitos aspectos do comportamento humano, como: altruísmo, empatia, comportamento moral, aprendizagem social e normas sociais. [Alexander 2009]

Para a evolução das estratégias dos jogadores no modelo evolucionário e espacial do jogo do ultimato apresentado neste artigo, utiliza-se algoritmos genéticos (AGs).²

O objetivo de AGs é encontrar soluções aproximadas para problemas de grande complexidade computacional [Goldberg 1989, Laguna and Moscato 1996] mediante o processo de *evolução simulada* que possui uma manipulação *cega* dos cromossomos, ou seja, o processamento não possui nenhuma informação a respeito do problema que está tratando de resolver, exceto o valor da função objetivo.

No conceito original, a função objetivo, também conhecida por função de avaliação, é a única informação que avalia o cromossomo. Um indivíduo da população é representado por um único cromossomo, contendo a codificação (genótipo) de um candidato à solução do problema (fenótipo).

Um AG canônico funciona da seguinte forma [Aguiar 1998]: i) uma população

²AGs foram introduzidos por John Holland nos anos 60, com base na teoria do naturalista Darwin (1859), o qual afirma que os indivíduos mais adaptados ao seu ambiente possuem maior chance de sobreviver e gerar descendentes [Linden 2008].

de cromossomos se mantém ao longo de todo o processo; ii) a cada um dos cromossomos associa-se um valor de adaptação que está diretamente relacionado com o valor da função objetivo a otimizar; iii) cada cromossomo codifica um ponto no espaço de busca do problema; iv) dois cromossomos são selecionados de acordo com seus valores de adaptação para serem os geradores de duas novas configurações mediante um processo de reprodução; v) estas novas configurações ocupam, reservam, seu espaço na nova geração. Este processo é repetido tantas vezes quantas forem necessárias.

3. O Modelo Proposto

Nesta seção, apresenta-se o modelo proposto para uma abordagem evolucionária e espacial para o jogo do ultimato, com um conjunto de n agentes com preferências sociais, conectados por uma rede complexa *small-world-network*³, que define a vizinhança para cada um dos n agentes do sistema multiagente (SMA). O modelo proposto está baseado no trabalho de Xianyu [Xianyu 2010].

Cada jogo do ultimato entre dois jogadores é jogado em duas etapas, alternando os papéis de proponente e respondente entre os jogadores.

Em cada ciclo da simulação, os agentes interagem com todos agentes de sua vizinhança, somando suas recompensas através da função de *payoff* e analisando sua utilidade estimada por funções de utilidade específicas definidas pelas preferências dos indivíduos da população.

Em cada interação entre dois agentes, o valor total a ser dividido entre eles é igual a 1 (um). Cada agente possui diferentes estratégias de jogo, respeitando, se existirem, as restrições de suas preferências sociais. A estratégia é dada por um par de números reais o_i e r_i , com $o_i, r_i \in [0, 1]$, aqui denotada por est (o_i, r_i) , onde: o_i é a oferta do agente i , quando atua como proponente, e r_i é um valor de reserva ou mínimo aceitável do agente i , quando este é o jogador que responde.

Em cada ciclo da simulação, os valores de oferta o_i e reserva r_i são ajustados procurando maximizar a função de utilidade adotada pelo agente, de acordo com sua preferência social. A partir do cálculo das utilidades e da análise das mesmas em diferentes estados, os indivíduos selecionam uma melhor estratégia para a condição atual.

A Função de Recompensa

Em cada jogo, se o agente i com a estratégia est (o_i, r_i) interage com o agente j com a estratégia est (o_j, r_j) , a recompensa $p_{ij}(o_i, o_j)$ que o agente i obterá está dada pela função $p_{ij} : [0, 1] \times [0, 1] \rightarrow [0, 1]$, definida da seguinte forma:

$$p_{ij}(o_i, o_j) = \begin{cases} 1 - o_i + o_j & \text{se } o_i \geq r_j \text{ e } o_j \geq r_i \\ 1 - o_i & \text{se } o_i \geq r_j \text{ e } o_j < r_i \\ o_j & \text{se } o_i < r_j \text{ e } o_j \geq r_i \\ 0 & \text{se } o_i < r_j \text{ e } o_j < r_i \end{cases} \quad (1)$$

Na Eq. (1), se o agente i , quando proponente, faz uma oferta o_i que é maior ou igual do que o mínimo que o agente j está disposto a aceitar, e, por sua vez, o agente

³Uma rede Mundo Pequeno (do inglês, *small-world network*) é uma rede que possui alto grau de agrupamento e baixa distância média entre os vértices [Watts and Strogatz 1998].

j , quando proponente, faz uma oferta o_j maior ou igual ao mínimo que o agente i está disposto a aceitar, então o ganho do agente i , dado por $1 - o_i + o_j$, é o maior ganho que este agente pode obter nesta interação. Ou seja, o maior ganho para um agente é quando ocorrem as duas trocas. Similarmente, interpretam-se os outros casos da função de recompensa. Observa-se que, o último caso se refere a quando as propostas de cada um dos agentes são ambas recusadas nas duas fases do jogo. Este é o caso em que não há ganho para nenhum dos jogadores, caso em que não ocorre nenhuma troca.

A *recompensa total* (somatório das recompensas) dos agentes é obtida após cada agente ter jogado com todos os seus vizinhos.

As Preferências Sociais

Cada agente é codificado com uma das três diferentes formas de preferência social apresentadas em [Fehr and Schmidt 1999, Andreoni and Miller 2002, Xianyu 2010]:

- (i) *Nível mínimo aceitável*: agentes que possuem um valor mínimo aceitável durante as trocas;
- (ii) *Aversão à desigualdade*: agentes evitam jogadas com resultados desiguais;
- (iii) *Bem-estar social*: agentes que desejam o bem estar da comunidade.

As Funções de Utilidade

Para tomada de decisão sobre como modificar suas estratégias, cada um dos indivíduos da população se baseia na avaliação de funções de utilidades (tal como em [Fehr and Schmidt 1999, Xianyu 2010]).

Seja um agente i com preferências sociais “nível mínimo aceitável” ou “aversão à desigualdade”, número de vizinhos $m - 1$, grau de sofrimento a_i (ou inveja, quando seu retorno é menor do que os seus agentes vizinhos), grau de sofrimento b_i (ou culpa, quando o retorno do agente é maior do que os seus agentes vizinhos), e vetor de alocação de recompensas (*payoffs*) $X = \{x_1, x_2, \dots, x_n\}$, onde n é o número de jogadores, então a utilidade do agente i é dada por:

$$U_i(X) = x_i - \frac{a_i}{(m-1)} \sum_{j \neq i} \max(x_j - x_i, 0) - \frac{b_i}{(m-1)} \sum_{j \neq i} \max(x_i - x_j, 0)$$

Considera-se agora o terceiro tipo de preferência social, quando os agentes se preocupam com o bem estar social dos seus vizinhos. Sejam $m - 1$ vizinhos, grau de inveja a_i , grau de culpa b_i , peso w_i do agente i sobre sua preocupação com o bem estar social dos seus vizinhos, e o vetor de alocação de recompensas $X = \{x_1, x_2, \dots, x_n\}$, então a utilidade do agente i é dada por:

$$U_i(X) = x_i + w_i \sum_{j \neq i} x_j - \frac{a_i}{(m-1)} \sum_{j \neq i} \max(x_j - x_i, 0) - \frac{b_i}{(m-1)} \sum_{j \neq i} \max(x_i - x_j, 0)$$

Utilizando um Algoritmo Genético para Evolução das Estratégias dos Agentes

Neste modelo, tanto a oferta o_i como a reserva r_i do agente i podem ser ajustados em cada ciclo da simulação (após cada jogador ter efetuado um jogo completo com

todos os seus vizinhos) com intuito de maximizar a utilidade. No entanto, o agente i , quando proponente, não possui informações precisas sobre o menor valor que o outro jogador está disposto a aceitar, ou seja, não conhece o perfil do outro jogador, tratando-se assim de um jogo de informação incompleta. Para modelar o processo de evolução de estratégias dos agentes neste contexto, foi desenvolvido um algoritmo genético, baseado em [Xianyu 2010].

Cada agente é constituído por um cromossomo⁴ codificado com 14 genes

$$[g_i^0, \dots, g_i^{13}],$$

refletindo suas preferências sociais e a forma como o agente evolui suas estratégias, onde:

- g_i^0 consiste no valor de oferta o_i realizada pelo agente i , quando proponente;
- g_i^1 é o nível de reserva ou mínimo aceitável do agente i quando respondente;
- g_i^2 é o grau de sofrimento quando o retorno do agente é menor do que os seus agentes vizinhos (inveja), representando a variável a_i das funções de utilidade;
- g_i^3 é o grau de sofrimento quando o retorno do agente é maior do que os seus agentes vizinhos (culpa), representando a variável b_i das funções de utilidade;
- g_i^4 é o peso de quanto o agente i se preocupa com o bem-estar dos outros agentes, representando a variável w_i que aparece na utilidade dos indivíduos com preferência social “bem-estar social”;
- g_i^5, \dots, g_i^{13} são os elementos do vetor de probabilidades que ajustam as estratégias após cada ciclo da simulação.

Assim, cada agente tem um vetor de probabilidades

$$[g_i^5, \dots, g_i^{13}]$$

que, a cada ciclo, determina como as estratégias do agente serão modificadas para utilização no próximo ciclo. Esse vetor de probabilidades tem nove elementos correspondentes às alternativas possíveis para ajuste das estratégias o_i e r_i , que podem aumentar, diminuir ou não alterar seus valores. As modificações no vetor serão realizadas de acordo com a análise das funções de utilidade adotadas pelo agente.

Sejam, $g_i^5 = p_i^0, \dots, g_i^{13} = p_i^8$. As nove alternativas para modificar as estratégias do agente i em cada ciclo são codificadas no seu vetor de probabilidades como:

- p_i^0 é probabilidade de aumentar ambos os valores o_i e r_i ;
- p_i^1 é probabilidade de aumentar o_i e diminuir r_i ;
- p_i^2 é probabilidade de diminuir ambos os valores o_i e r_i ;
- p_i^3 é probabilidade de diminuir o_i e aumentar r_i ;
- p_i^4 é probabilidade de aumentar o_i e não alterar o valor de r_i ;
- p_i^5 é probabilidade de diminuir o_i e não alterar o valor de r_i ;
- p_i^6 é probabilidade de não alterar o valor de o_i e aumentar r_i ;
- p_i^7 é probabilidade de não alterar o valor de o_i e diminuir r_i ;
- p_i^8 é probabilidade de aumentar não alterar nem o valor de o_i nem o valor de r_i .

⁴Em AGs um cromossomo é uma estrutura de dados que representa uma das possíveis soluções do espaço de busca do problema.

A alternativa para modificar as estratégias do agente i é escolhida de acordo com o número aleatório gerado no intervalo $[0, 1]$.

Assim, o agente i é representado por:

$$[o_i \ r_i \ a_i \ b_i \ w_i \ p_i^0 \ p_i^1 \ p_i^2 \ p_i^3 \ p_i^4 \ p_i^5 \ p_i^6 \ p_i^7 \ p_i^8]$$

O vetor de probabilidades é ajustado a cada ciclo, para refletir o desempenho da escolha das alternativas de evolução das estratégias, mantendo sempre o somatório das probabilidades igual a 1.

Por exemplo, se o cromossomo que representa o agente i se adapta, evolui, de forma que o agente aumenta sua utilidade, então a probabilidade p_i^k , com $0 \leq k \leq 8$, da alternativa escolhida na rodada anterior, é aumentada e proporcionalmente as outras probabilidades são diminuídas, mantendo a soma de todas elas igual a um.

Caso contrário, quando o cromossomo não evolui, isto é, o agente não aumenta sua utilidade, então a probabilidade da estratégia p_i^k , com $0 \leq k \leq 8$, escolhida no ciclo anterior, é diminuída e proporcionalmente as outras probabilidades são aumentadas.

O modelo possui um fator denotado por f_p para determinar em quanto se aumenta ou diminui percentualmente as probabilidades do vetor de probabilidades, durante o processo de simulação. Também existe um fator f_e para determinar em quanto se aumenta ou diminui percentualmente os valores o_i e r_i que definem a estratégia do agente i .

Submetida a um processo evolucionário a população deverá conter indivíduos mais aptos. O algoritmo deste trabalho pode ser considerado como um tipo de algoritmo de aprendizado por reforço. Através do processo de evolução, as estratégias que obtém retornos maiores tem maior probabilidade de sobreviver e alcançar mais oportunidades para uso futuro.

4. Simulações e Análise de Resultados

A implementação do modelo de Jogo do Ultimato Espacial e Evolucionário proposto neste trabalho foi realizada no NetLogo. Como explicado na Seção 3, para a distribuição das conexões entre os agentes é utilizada uma topologia de rede *Small World Network*. Foi definida uma população com indivíduos que podem assumir uma das três diferentes formas de preferência social, discutidas também na Seção 3. Os grupos (sub-redes) formados por estes indivíduos são heterogêneos quanto à preferência social e de tamanho não fixo, representado por um vetor.

Os parâmetros (culpa a_i , inveja b_i , peso w_i , fator f_p , fator f_e) definidos no modelo podem ser modificados. Para cada configuração destes parâmetros são realizadas 25 simulações. Cada simulação realizada tem 2000 ciclos. Em todas as simulações, o valor inicial da oferta o_i e da reserva r_i são aleatórios.

A população utilizada em todas as simulações compreende 900 agentes repartidos em sub-redes. Os 900 indivíduos desta população são divididos igualmente para cada uma das preferências (300 agentes apresentam restrições quanto ao “mínimo aceitável”, 300 agentes “avessos à desigualdade” e 300 agentes que desejam o “bem-estar social”).

Além das interações entre os indivíduos vizinhos dentro de uma mesma sub-rede,

em cada uma destas é escolhido um líder⁵ e o jogo do ultimato é realizado entre todos os líderes, caracterizando assim trocas entre todas as sub-redes.

Dois cenários de simulação foram utilizados.

Cenário 1:

O modelo foi parametrizado de modo que cada agente i da população:

- Ajusta a probabilidade predominante no vetor de probabilidades (aumentando ou diminuindo) em 25%;
- Ajusta (aumentando ou diminuindo) a oferta e a reserva em 50%.

Agentes i com a preferência social “mínimo aceitável” têm $a_i = 0.9$, $b_i = 0.2$ e o mínimo valor para r_i é 0.2; agentes “avessos à desigualdade” têm $a_i = 0.5$ e $b_i = 0.5$; e agentes que desejam o “bem-estar social” têm $a_i = 0.4$, $b_i = 0.3$ e $w_i = 0.2$.

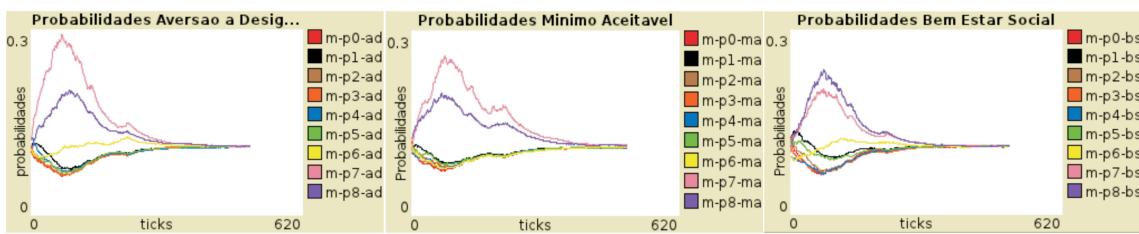


Figura 1. Aspecto geral dos vetores de probabilidades de acordo com a estratégia no Cenário 1

Como pode ser observado no exemplo de uma simulação com 2000 ciclos apresentado na Figura 1, as probabilidades que regem a tomada de decisão sobre o ajuste de oferta e reserva para cada agente estabilizam após um dado número de ciclos. O mesmo acontece, em média, com a oferta e a reserva dos agentes, de acordo com a sua preferência social, como pode ser observado na Figura 2.

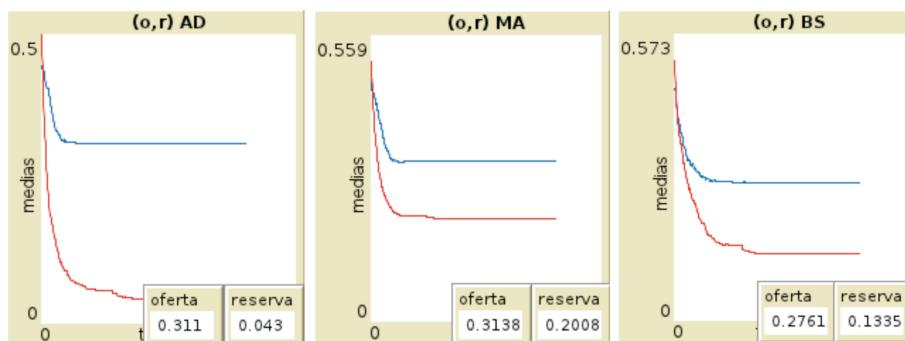


Figura 2. Aspecto geral das médias de oferta e reserva de acordo com a estratégia Cenário 1

Devido a este quadro de estabilização do sistema, criou-se um segundo cenário para a simulação e verificação das políticas de oferta e reserva.

⁵O agente com a maior utilidade entre todos os vizinhos

Cenário 2:

Neste cenário, adota-se inicialmente os mesmos parâmetros utilizados no Cenário 1. Entretanto, a cada 500 ciclos de simulação, a oferta e a reserva médias dos indivíduos em cada preferência social são replicadas em todos os agentes com a mesma preferência. A este processo chamou-se de *política de influência*. Além disso, neste momento, a topologia da rede também é alterada para verificar se a otimalidade das ofertas e reservas dependem da vizinhança.

Realizaram-se 25 simulações com este cenário. A Figura 3 mostra um exemplo de evolução do número de trocas que ocorrem durante os 2000 ciclos da simulação. É possível observar que no começo da simulação o número de vezes em que ocorrem duas trocas entre os agentes é pequeno e o número de uma e nenhuma troca é alto, consequentemente a média da utilidade destes é baixa.

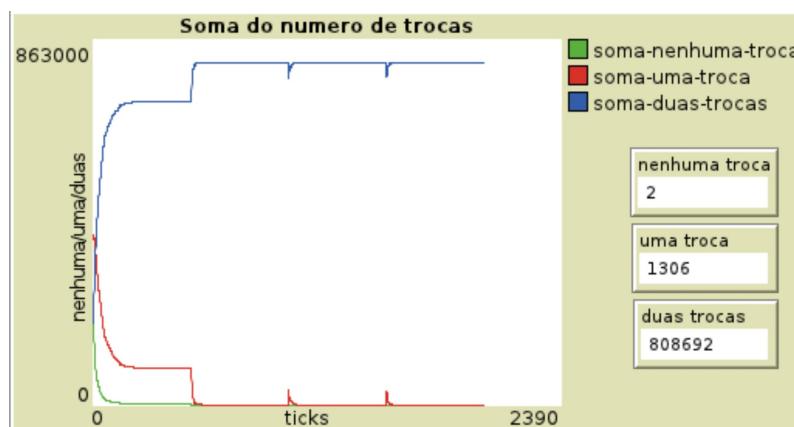


Figura 3. Número de trocas realizadas sucessivamente em diferentes redes após inclusão da política de influência no Cenário 2

Ao observar a Figura 3, é possível concluir que o número de duas trocas do sistema evolui, convergindo no final para um alto número de realizações de duas trocas implicando em uma utilidade mais alta para os agentes do SMA. Isto ocorreu porque os agentes balancearam objetivos individuais e objetivos coletivos motivando trocas com outros agentes, sejam quais forem suas preferências, sem deixar de ter estratégias direcionadas ao seu equilíbrio. Consequentemente, ocorre a otimização das utilidades da população, auto-regulando os valores econômicos da população.

A Figura 4 ilustra as utilidades alcançadas por cada uma das preferências com os parâmetros comentados anteriormente. São apresentados os valores absolutos da utilidade dos agentes em cada preferência social após 25 simulações. No gráfico com linha azul, esta representando a utilidade dos agentes “avessos à desigualdade”, o gráfico com linha vermelha representa a utilidade dos agentes com preferência social “mínimo aceitável”, e o gráfico com linha verde representa a utilidade dos agentes que preferem o “bem-estar social”.

A Figura 5 ilustra os resultados da quantidade de duas trocas, uma troca e nenhuma troca de 25 simulações realizadas no Cenário 2. A linha em azul representa a quantidade de vezes que não ocorreu nenhuma troca, a linha em vermelho representa a quantidade de vezes que ocorreu uma troca e a linha em verde a quantidade de vezes em

que ocorreram duas trocas. Portanto, no final de cada uma das 25 simulações realizadas no cenário 2, observa-se que ocorre a maximização do número de trocas bem sucedidas.

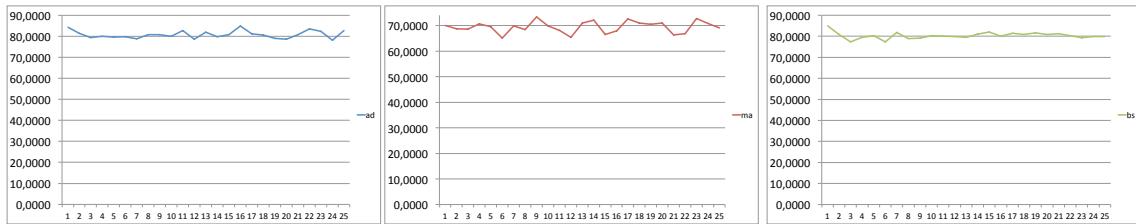


Figura 4. Níveis absolutos de utilidade para cada preferência social em 25 simulações no Cenário 2. Da esquerda para a direita: aversão à desigualdade, mínimo aceitável e bem estar social.

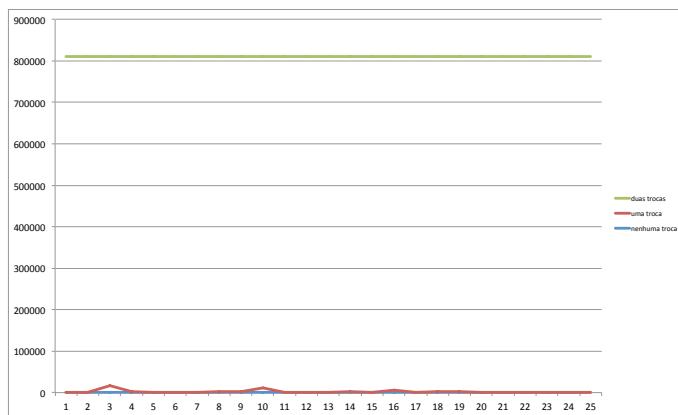


Figura 5. Resultado da quantidade de duas trocas, uma troca e nenhuma troca em 25 simulações no Cenário 2

Os resultados das simulações revelaram que a distribuição de estratégias do nível de oferta ficam concentradas no intervalo [0.2, 0.4], não existindo indivíduos com outra distribuição de estratégias referentes a oferta. Isto é, os agentes convergem para níveis de oferta aproximados.⁶

Ainda neste mesmo cenário, foram realizadas outras simulações diminuindo-se 0.1 dos valores a_i e b_i referentes a cada um dos três tipos de preferência social. Notou-se que agentes com menor culpa e inveja, seja qual for sua preferência, tem um aumento significativo na utilidade final, tal como no exemplo de simulação mostrado na Figura 6.

Comparação dos resultados entre os dois cenários

Para avaliar os testes de desempenho realizados, utilizou-se o *Teste t de Student*, também conhecido como “teste t”. Analisando os resultados das simulações realizadas, definidas com mesmos parâmetros de culpa a_i , inveja b_i , peso w_i , fator f_p e fator f_e para os dois cenários, o Cenário 2 apresentou uma diferença extremamente significante estatisticamente em relação ao Cenário 1.

⁶Observou-se que isto não ocorreu nos resultados do trabalho em [Xianyu 2010], que, embora mostraram uma concentração grande de agentes com nível de oferta no intervalo [0.2, 0.4], o menor intervalo ocorrido para todas as preferências sociais referentes a estratégia do nível de oferta é [0, 0.5].

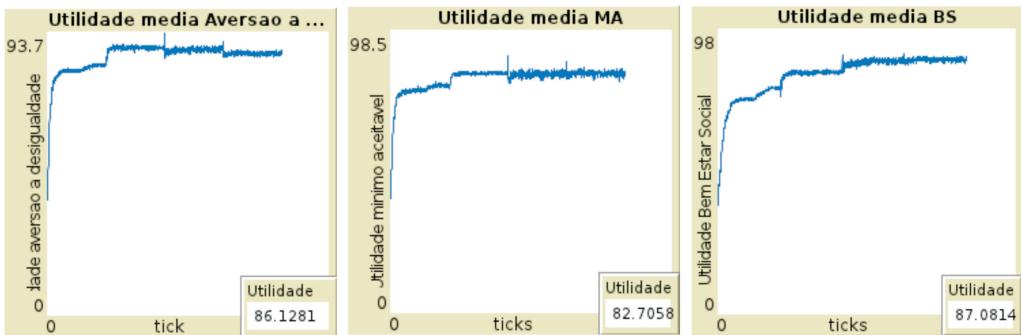


Figura 6. Aspecto geral das utilidades de acordo com novos parâmetros de a_i e b_i no Cenário 2

Foram comparados os resultados da quantidade de duas trocas realizadas e o valor final da utilidade média para cada uma das três preferências sociais. Para todas comparações avaliadas, com “teste t”, a diferença entre os dois cenários é estatisticamente significante, pois para todos os casos há 95% de certeza de que o Cenário 2 é superior ao Cenário 1, no sentido do número de trocas efetivamente concretizadas e o valor médio das utilidades.

5. Conclusão

Este artigo apresentou uma abordagem evolucionária e espacial para o Jogo do Ultimato, para estudar o comportamento de trocas econômicas entre uma população heterogênea de agentes, visando alcançar a auto-regulação das trocas, maximizando a quantidade de trocas bem sucedidas, com o balanceamento dos objetivos individuais e coletivos.

Os agentes, organizados segundo uma rede complexa de mundo pequeno, apresentam diferentes preferências sociais, e não conhecem o perfil de seus oponentes. Embora com autonomia para alcançar seus objetivos individuais, os agentes são motivados a realizar trocas com outros agentes pela sua função de recompensa. Utilizando um algoritmo genético, os agentes evoluem suas estratégias de trocas, no sentido de aumentar consideravelmente a quantidade de trocas durante as várias etapas do jogo e, consequentemente, aumentar suas utilidades.

Foram utilizados dois tipos diferentes de cenários para a simulação. No primeiro cenário, foram realizadas simulações com uma combinação de parâmetros que determinam as preferências sociais dos agentes e as taxas de variação do vetor de probabilidades e das estratégias. No segundo cenário introduziu-se a política de influência, onde os valores médios das ofertas e reservas dos agentes de uma mesma preferência social, a cada 500 ciclos, são imitadas por todos os outros agentes da mesma preferência social. Além disso, as redes são reconfiguradas aleatoriamente.

Considerando os resultados dos testes de desempenho realizados com os dois cenários, conclui-se que o Cenário 2 chega a resultados melhores quanto ao número de duas trocas e, consequentemente, obtém melhor utilidade para os agentes do SMA do que o Cenário 1.

Os resultados das simulações realizadas com o modelo proposto neste trabalho mostram que os agentes evoluem em suas estratégias, no sentido de maximizar o número

de trocas bem sucedidas, e consequentemente, maximizar as suas utilidades, respeitando as preferências sociais dos indivíduos.

Como trabalho futuro, pretende-se desenvolver o Jogo de Auto-Regulação de Processos de Trocas Sociais, onde consideram-se trocas não econômicas no sentido de Piaget. [Piaget 1995, Dimuro et al. 2011]

Agradecimentos

Este artigo foi financiado pela CAPES, CNPq (Proc. 476234/2011-5, 560118/10-4,305131/2010-9) e FAPERGS (Proc. 11/0872-3).

Referências

- Aguiar, M. S. (1998). Análise formal da complexidade de algoritmos genéticos. Master's thesis, PPGC da UFRGS, Porto Alegre/RS.
- Alexander, J. M. (2009). Evolutionary game theory. In *Stanford Encyclopedia of Philosophy*, Stanford. Stanford University.
- Andreoni, J. and Miller, J. (2002). Giving according to garp: An experimental test of the consistency of preferences for altruism. *Econometrica*, 70(2):737–753.
- Camerer, C. F., Ho, T.-H., and Chong, J. K. (2001). Behavioral game theory: Thinking, learning and teaching. *JOURNAL OF RISK AND UNCERTAINTY*, 19:7–42.
- Dimuro, G. P., Costa, A. C. R., Gonçalves, L. V., and Pereira, D. R. (2011). Recognizing and learning models of social exchange strategies for the regulation of social interactions in open agent societies. *Journal of the Brazilian Computer Society*, 17(3):143–161.
- Fehr, E. and Schmidt, K. M. (1999). A theory of fairness, competition, and cooperation. *The Quarterly Journal of Economics*, 114(3):817–868.
- Fiani, R. (2006). *Teoria dos Jogos*. CAMPUS.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts.
- Kellermann, G. A. (2008). Aspectos estatísticos e dinâmicos do jogo do ultimato espacial e não espacial. Master's thesis, PPGC da UFRGS, Porto Alegre/RS.
- Laguna, M. and Moscato, P. (1996). Capítulo 3: Algoritmos genéticos. In Diaz, B. A., editor, *Las Nuevas Técnicas Heurísticas y las Redes Neuronales*, pages 67–103. Ed. Paraninfo, Madrid.
- Lieberman, E., Hauert, C., and Nowak, M. A. (2005). Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316.
- Linden, R. (2008). *Algoritmos Genéticos*. Brasport, Rio de Janeiro/RJ, 2 edition.
- Nowak, M. A., Page, K. M., and Sigmund, K. (2000). Fairness versus reason in the ultimatum game. *Science*, 289(5485):1773–1775.
- Oosterbeek, H., Sloof, R., and van de Kuilen, G. (2004). Cultural differences in ultimatum game experiments: Evidence from a meta-analysis. Experimental 0401003, EconWPA.
- Page, K. M., Nowak, M. A., and Sigmund, K. (2000). The spatial ultimatum game. *Proceedings of the Royal Society of London B Biological Sciences*, 267(1458):2177–2182.
- Piaget, J. (1995). *Sociological Studies*. Routlege, London.
- Szaba, G. and Fáth, G. (2007). Evolutionary games on graphs. *Physics Reports*, 446:97–216.
- von Neumann, J. and Morgenstern, O. (1944). *Theory of games and economic behaviour*. Princeton UP.
- Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.
- Xianyu, B. (2010). Social preference, incomplete information, and the evolution of ultimatum game in the small world networks: An agent-based approach. *Journal of Artificial Societies and Social Simulation*, 13:2.

Extending the RoboCup Rescue to Support Stigmergy: Experiments and Results

Gabriel R. C. Jacobsen, Carlos A. Barth, Fernando dos Santos

¹Departamento de Sistemas de Informação
Universidade do Estado de Santa Catarina (UDESC) – Ibirama, SC – Brazil

gabrielrigoj@gmail.com, {carlos.barth, fernando.santos}@udesc.br

Abstract. Social insects have inspired researches in computer sciences as well as engineers to develop models for coordination and cooperation in multiagent systems. One example of these models is the model of stigmergy. In this model agents use indirect communication (communication through the environment) in order to coordinate actions. The RoboCup Rescue simulator is used as a testbed to evaluate this model in a real world considering a highly constrained scenario of an earthquake. This paper investigates the feasibility of using stigmergy in the RoboCup Rescue and the improvements of performance that the agents can be led to. We extended the RoboCup Rescue environment to enable the use of stigmergy by the agents in it. Experimental results shown that the use of stigmergy leads to an improvement on agents' performance by 11.5% to 26%, depending on the scenario.

1. Introduction

An agent is a computational system situated in an environment, being capable of acting in an autonomous fashion to accomplish its own goals [16]. A multiagent system is composed by agents that can interact through coordination, cooperation or negotiation, in order to reach global goals [7].

A lot of approaches for coordination, cooperation and negotiation are available in the literature. Ideally, those approaches shall be evaluated on real world scenarios to check their effectiveness. The RoboCup Rescue[8] simulator was created with this purpose.

In the RoboCup Rescue the agents face a catastrophic environment (earthquake), and should mitigate the situation in order to minimize the material and human damages. The simulator replicates a highly constrained environment regarding traffic and inter-agent communication. An example of these constraints is the radio channel, that is limited by a maximum number of bytes an agent can send through a channel at once. Those restrictions impose the need of an effective use of the available resources. Despite the ability of agents to communicate among themselves via radio channels, if the communication system was broken (due to the catastrophe) there would be no way to interact.

Social insects have been inspired computer scientists and engineers to develop approaches for coordination and cooperation. The book of Bonabeau et al. [1] presents a review of some computational models created from observations on social insects. One of those models, which is particularly interesting to this paper, is the model of stigmergy [2]. Through stigmergy, the insects colony reaches self-organization with no direct interactions among their individuals. All the interactions are done by indirect communication, through the environment, using pheromones.

We have not found any study regarding the use of stigmergy in the RoboCup Rescue environment. The current version of the RoboCup Rescue does not allow indirect communication between agents through stigmergy. So, this paper investigates two hypothesis: a) it is possible to use stigmergy in the RoboCup Rescue environment; and b) the performance of agents is improved using stigmergy.

We extended the RoboCup Rescue simulator to enable the use of stigmergy by the agents. We performed a set of experiments to verify the feasibility of stigmergy and to compare the performance against agents that do not use stigmergy. The results showed that the use of stigmergy leads to an improvement on agents' performance by 11.5% to 26%, depending on the scenario.

This paper is organized as follows. Section 2 presents the required background on swarm intelligence. Section 3 presents the RoboCup Rescue simulator. Section 4 presents related works. Section 5 describes the proposed RoboCup Rescue extension to use stigmergy. Section 6 presents the empirical evaluation via a set of experiments. Finally, section 7 shows the conclusions and future work.

2. Swarm Intelligence

Social insects, like ants and termites, organize themselves in colonies. Despite the simplicity of each individual, the colony as a whole is able to deal with complex problems, like the construction of nests and the cooperative transport of prey. According to [1], the collective activities of social insects are self-organized. The complex collective behavior may emerge from interactions among individuals that exhibit simple behavior, in a flexible and robust way. These abilities inspired engineers and computer scientists to develop models that mimic the self-organized behavior of social insects to solve problems. These models are then used to build swarm intelligent agents and systems.

Deneubourg et al. [2] developed a model of stigmergy, a phenomenon observed in some species of social insects. In such phenomenon, the colony reaches self-organization with no direct interactions among the individuals. All the interactions are done by indirect communication, through the environment, using a pheromone trail. The pheromone trail stimulates the individuals, which take certain actions in response to the stimulus.

Stigmergy is observed in the process of ant foraging, in which ants search for food. Initially there is no pheromone in the environment, which means that the ants take random paths to search for food. If an ant finds a source of food, this ant moves back to the nest, laying a pheromone trail while walking. When the nestmates sense the pheromone trail, they are stimulated to follow it to the food source. After a while, a lot of ants are engaged in the transportation of the food by following the shortest pheromone path between the nest and the source of food. Dorigo et al. [3] used the model of stigmergy to solve the classical traveling salesman problem.

The model of [2] says that the decision of an ant to follow a path is probabilistic, and take into account the number of ants that already followed the path. In other words, given two paths A and B , after i ants followed some of these paths, there will be A_i pheromone units on path A and B_i units on path B . The next ant $i + 1$ chooses path A or B with probabilities $prob_A$ and $prob_B$, depending on A_i and B_i , as shown in equation 1.

$$prob_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n} \quad prob_B = (1 - prob_A) \quad (1)$$

The higher the value of A_i is, the higher is the probability of choosing the path A . The parameter n is used to specify the degree of nonlinearity: when n is high, a path with slightly more pheromone than the other will have a higher probability of being chosen. The parameter k quantifies the degree of attraction of an unmarked path: when k is high, a higher amount of pheromone is necessary to make the choice nonrandom.

An ant that passes on a path already marked with pheromone also drops a certain amount of pheromone to reinforce the stimulus on its nestmates. Thus, the amount of pheromone present on the path depends on the number of ants which already passed on that path. Given that $\tau_A(t)$ represents the amount of pheromone on a path A at a time instant t , the amount of pheromone in the next time instant $t + 1$ can be expressed by equation 2 [3]

$$\tau_A(t + 1) = \rho * \tau_A(t) + \Delta\tau_A(t, t + 1) \quad (2)$$

where $\Delta\tau_A(t, t + 1)$ is the amount of pheromone dropped between the time t and $t + 1$ by every ant k that passed on the path A , as show in equation 3.

$$\Delta\tau_A(t, t + 1) = \sum_{k=1}^m \Delta\tau_A^k(t, t + 1) \quad (3)$$

The value of ρ represents the coefficient of pheromone evaporation. As the ants stop using a certain path (i.e. the food at the source is over), the pheromone evaporates as the time evolves. The lower is the amount of pheromone on some path, the lower is the stimulus to the ants. In that sense, the evaporation avoids the existence of paths that lead to nowhere, or suboptimal paths. According to [3], $0 \leq \rho < 1$, and its value can be experimentally set in a way that gives the best results to the system.

3. RoboCup Rescue Simulator

The goal of the RoboCup Rescue Simulation League [8, 15] is to provide a testbed for simulated rescue teams acting in situations of urban disasters. Currently the RoboCup Rescue simulator tries to reproduce conditions that arise after the occurrence of an earthquake in an urban area, such as the collapsing of buildings, road blockages, fire spreading, buried and/or injured civilians. The simulator incorporates some collaborative agents acting to mitigate the situation. Some issues such as heterogeneity, limited information, limited communication channel, planning, and real time characterize this as a complex multiagent domain [8]. RoboCup Rescue aims at being an environment where multiagent techniques that are related to these issues can be developed and benchmarked.

In the RoboCup Rescue, the main agents are fire brigades, police force, and ambulance teams. Agents have limited perception of their surroundings; can communicate only by radio channels, but are limited in the number and size of messages they can exchange. Regarding information and perception, agents have knowledge about the map. This allows agents to compute the paths from their current locations to given places. However

this does not mean these paths are free since an agent has only limited information about the actual status of some path, e.g. whether or not it is blocked by debris.

At the implementation level, the simulator is composed of a kernel and a set of modules. The kernel controls the simulation, invoking every module as needed. Each module is responsible for simulate an aspect of the disaster scenario. For instance, the fire module simulates fire ignition and propagation on the buildings, and the traffic module is responsible for moving the agents in the map.

Each simulation follows a sequence of time steps. On each time step, every agent must decide what action will be taken, given its perception. The perception is composed of objects representing the environment (e.g. a building object, a road object, etc). For each object, the agent can access its properties (e.g. whether a building is on fire, amount of blockages over a road, etc). When a decision is taken, the agent sends a command to the kernel with the chosen action, e.g.: the agent sends a *move* command containing the path he wants to move on whenever the action is “to move on the environment”.

To measure the performance of the agents, the rescue simulator defines a score. The score takes into account a relation between the building area left undamaged and the initial building area. When there are civilians to be rescued, the score also considers a relation between the health condition of all civilians at the beginning and end of the simulation.

4. Related Work

Regarding the use of swarm intelligence and stigmergy in robotics and multiagent systems, the work of [9] applies swarm intelligence to a multiagent system in a real constrained world. The use of stigmergy (through virtual pheromones) proved to be a useful strategy for reducing the communication overhead between robots. Hoff et al. [6] use stigmergy as a message protocol, measuring the performance of the swarm in environments with and without obstacles. The work of [14] focuses on the capability of the robots in perceiving the environment as a way of making decisions in a collaborative swarm of robots through stigmergy. In [12] the authors aim to implement the necrophoric behavior of the bees as a way to give the robots in the swarm the capability of recognizing and rescuing a disabled robot. Payton et al. [10] presents a swarm of robots acting in a rescuing scenario, where virtual pheromones are used as a strategy of communication and coordination. In [11], the authors conclude that information about damages in essential infrastructure is crucial to make decision in a critical scenario. They propose that a decision support system can receive feedback of a swarm of robots specialized in inspecting infrastructure in a disaster scenario using stigmergy, collaborating to life maintenance efforts.

Regarding RoboCup Rescue, an efficient coordination amongst agents is a critical factor given the characteristics of the scenario and limited capabilities of the agents. Swarm techniques have been applied to get coordination. Ferreira Jr. et al. [4, 5] presented Swarm-GAP, a multiagent task allocation algorithm based on the model of division of labor in insect colonies. Santos and Bazzan [13] proposed *eXtreme-Ants*, also a multiagent task allocation algorithm which uses both the model of division of labor and the model of recruitment for cooperative transport. The recruitment model is used in *eXtreme-Ants* to deal with tasks which need a number of agents engaged simultaneously

to be accomplished. A drawback of both Swarm-GAP and *eXtreme-Ants* is the use of the communication channel to establish the agent's coordination.

The current version of the RoboCup Rescue simulator¹ does not allow an agent to use indirect communication, via stigmergy (dropping and sensing pheromones). Thus, we have not found any work that report results about the performance of a team of agents which uses stigmergy in this environment. This paper investigates this issue. We improve the simulator to allow the stigmergy, and run a set of experiments to verify the performance.

5. Stigmergy in the RoboCup Rescue

In order to use stigmergy in the RoboCup Rescue, we need to extend the simulator to accomplish the following requirements:

- (a) every road A must have a way to store an amount of pheromone τ_A ;
- (b) it must be possible to an agent k to perceive the pheromone τ_A stored on a road;
- (c) it must be possible to an agent k to drop certain amount of pheromone $\Delta\tau_A^k$ on a road;
- (d) the existing pheromone on a road must evaporate over a time, given an evaporation coefficient ρ .

To satisfy the requirement (a), we extended the implementation of the simulator to incorporate property τ_A on every road object A . The property stores the amount of pheromone virtually laid on each road. Once agents can access the properties set to a road object, these agents will detect the property value τ_A at every time step, satisfying the requirement (b).

In order to satisfy the requirement (c), we extended the implementation to provide to the agent a way it can informs the amount of pheromone $\Delta\tau_A^k$ it want to drop on every road of a given path. The traffic simulator, while processing the moving command, is responsible for increasing the amount of pheromone following the equation 2.

The last requirement (d) is accomplished by a change on the traffic simulator. At every time step, the traffic simulator applies the user-defined value of the evaporation coefficient ρ over the amount of pheromone τ_A present on every road, as given by equation 2.

6. Experiments and Results

To investigate our first hypothesis, which claims that is possible to use stigmergy in the RoboCup Rescue, we performed a series of experiments using the *Kobe4* map². We enabled only fire brigade agents to accomplish the task of fire-fighting. We also disabled the simulation of blockages, ensuring all roads are free. Fig. 1 presents the map used in the experiments.

The decision of disabling all agents but the fire brigades was based on our perception of similarity between the two tasks of fire-fighting and food-searching (ants). Regarding the movement of the group of agents in the environment, both tasks are based on the same main lines: first the agents explore the environment looking for some source; since some source is found a sign is sent to the other agents expecting some cooperation.

¹available at www.robocuprescue.org

²The *Kobe4* map is used in competitions of the *RoboCup Rescue Simulation League*. It is available for download at the site of the RoboCup Rescuesimulator

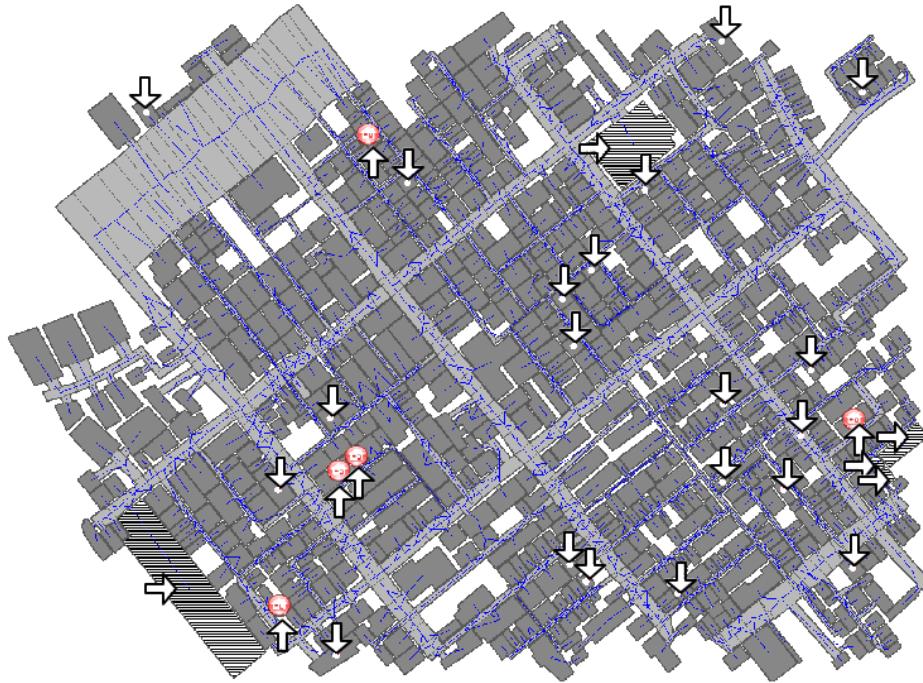


Figure 1. The *Kobe4* map used in the experiments with 5 refuges. The small circles (pointed out by up-to-down arrows) represent the initial position of the fire brigades. The big circles (pointed out by down-to-up arrows) represent the refuges, in which fire brigades can refill their water tanks. Striped rectangles represent the five initial fire spots (pointed out by left-to-right arrows).

Similar to ants searching for a food source, the fire brigade agents should explore the environment looking for buildings on fire. In the same way the ants transport a piece of food to the nest when they found it, fire brigades go back to some refuge to refill when they are running out of water. Since each agent has knowledge about the map and its roads, it can calculate the shortest path to the refuge. When traveling back to a refuge, each agent drops an amount of pheromone on the roads it passes. When the water tank is full, each agent that leaves the refuge is stimulated by the pheromone on the roads, according to equation 1. This fact increases the tendency of the agent to move to a fire and continue fighting it, similarly to the ants that leave the nest and are stimulated to move to a food source.

The experiments were performed on two scenarios: 20 agents and 1 refuge (20_1 for short), and 20 agents and 5 refuges (20_5). On each scenario, the following values were used for $\Delta\tau_A^k$ (amount of pheromone dropped by each agent on a piece of road): 0 (no pheromone), 1, 5, and 10 pheromone units. The evaporation coefficients used were 0 (no evaporation), 0.25, 0.50, 0.75, and 1 (total evaporation). To measure the performance of the agents, we use the building area left (e. g. after an earthquake followed by a fire and the intervention of the fire brigades). Figure 2 presents the results obtained. All data is averaged over 10 runs of the simulator. For the sake of completeness we show averages (and standard deviations) for all scenarios in appendix A (Table 1), where grey cells indicate the best score in each scenario.

As we can see, when we consider each scenario individually, there is no differ-

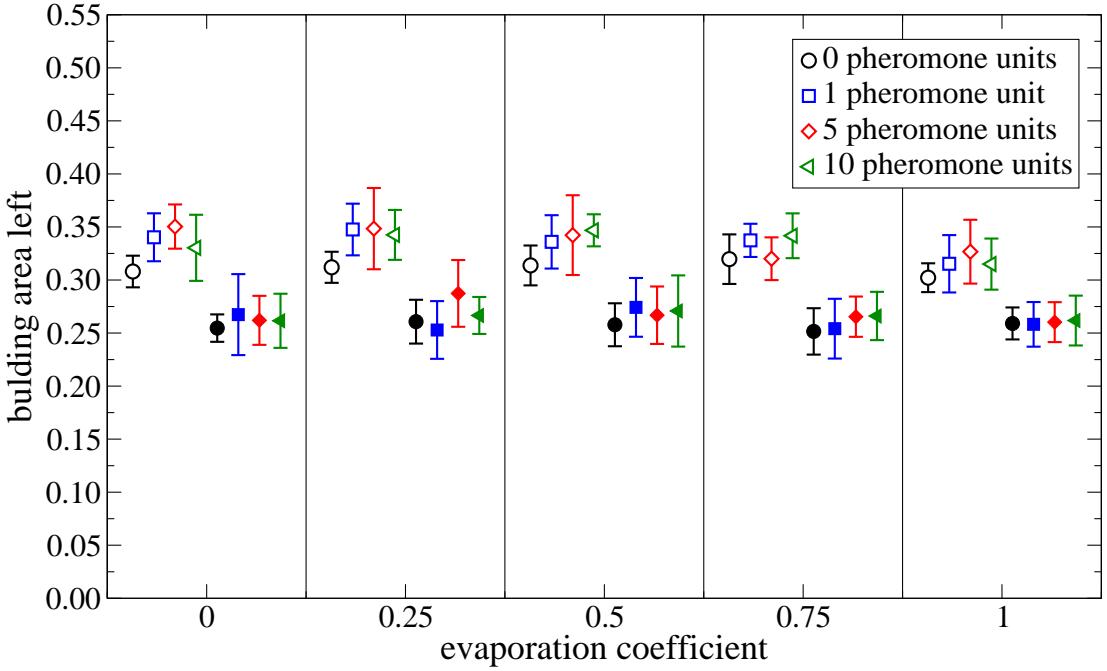


Figure 2. Results for each *Kobe4* scenario, showing the average building area left for each pheromone unit and evaporation coefficient. Unfilled symbols represent the 20_1 scenario, while filled symbols represent the 20_5 scenario.

ence that can be statistically significant between the performances, independently of the combination of evaporation coefficient and pheromone units. We believe this is due to the reduced set of values tested for those parameters. Given the complexity and dynamics of the environment, it is possible that a better performance is obtained with an unevaluated combination of evaporation coefficient and pheromone units.

From the experiments we can observe a relation between the use of stigmergy and the number of refuges where the fire brigades refill its tanks. Given 1 refuge enabled, the performance is higher (*t* test, 95% confidence) than enabling 5 refuges in all the scenarios. With one refuge enabled, all the agents move to it to refill its tanks, increasing the amount of pheromone in the paths. So, the stimulus on the agents which are leaving the refuge is increased. As a result, many agents move to the same fire spot, fighting the fire cooperatively and reducing the damages, leading to a improved performance.

Our second hypothesis, which conjectures that the performance of agents can be improved using stigmergy, is investigated comparing to agents that do not use stigmergy. For this comparison, we use the reference implementation of the sample fire brigade agent which comes in the RoboCup Rescue. While our fire brigades uses stigmergy to decide which fires to extinguish, the sample fire brigade adopts a greedy strategy, choosing to extinguish the closest fire in its perception, based on the euclidean distance between the agent and the fire spots. Except for this decision strategy, the other aspects of both sample agent and ours remains the same, i.e. when the water tank is empty, both types of agents go back to some refuge to refill. Thus, despite the simplicity of both types of agents, we have a fair comparison between them, taking into account only the aspect of interest to our hypothesis: the effect of stigmergy on the performance of agents.

Fig. 3 presents the results obtained for the sample fire brigade. The results showed for the stigmergy fire brigade are the best ones for each scenario, as highlighted on Table 1 (appendix A). As we can see, the average performance is improved in 26% in the scenario with 1 refuge, and by 11.5% with 5 refuges (*t* test, 95% confidence).

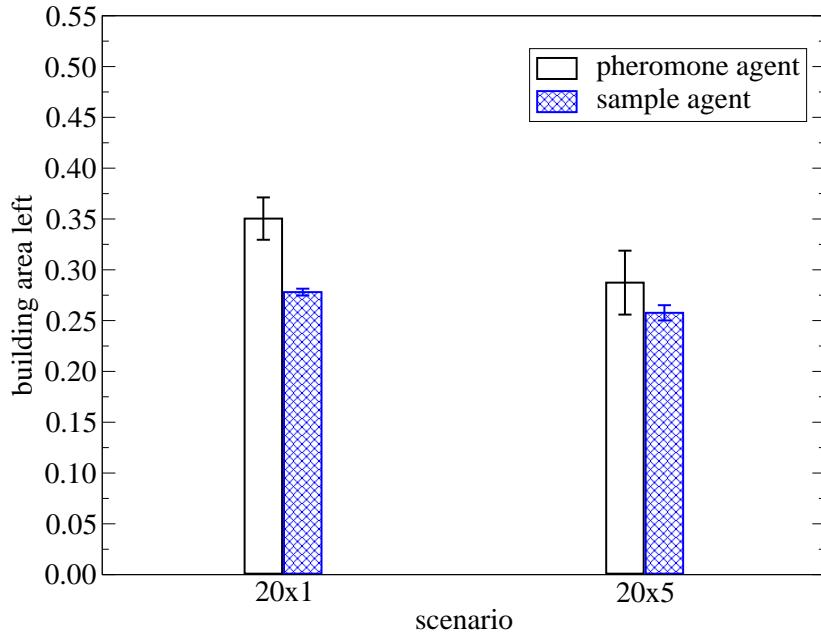


Figure 3. Results comparing fire brigades using stigmergy against sample fire brigades, showing how the performance is improved using stigmergy.

7. Conclusions and Future Work

In this paper we have addressed two hypothesis: the feasibility of using stigmergy in the RoboCup Rescue; and the improvement of the agent's performance as a result of using stigmergy. Previous works in robotics and multiagent systems [9, 6, 14, 12, 10, 11, 13, 5] do not consider stigmergy in an environment with the RoboCup Rescue characteristics.

We have extended the RoboCup Rescue to incorporate stigmergy. The agents can now drop and sense pheromones in the environment, enabling the formation of pheromone trails and thus, the use of stigmergy. The pheromone evaporation was also incorporated in order to minimize the existence of obsolete paths.

We have presented a set of experiments to demonstrate the use of stigmergy by fire brigade agents. The results obtained from the experiments show that the use of stigmergy is feasible. Moreover, the use of stigmergy leads to an improvement in the performance of agents from 11.5% to 26%, depending on the number of available refuges.

As future work we consider to investigate the literature about swarm intelligence to find out if there is a way of incorporating to the pheromone trails a notation of direction. Currently, there is no indication of the direction the pheromone trails points to. We believe this feature will improve the quality of the agent movement in the environment by avoiding the agents of getting stuck at some point (sometimes an agent keeps going back and forward on some part of the path).

We also consider to perform experiments with all three types of agents using stigmergy (fire brigades, police forces and ambulance teams). However, we need to extend the RoboCup Rescue in a way that the agents can be able to distinguish types of pheromones on the roads (e.g. fire brigades must be stimulated by pheromones dropped by other fire brigades rather than ambulance teams).

References

- [1] E. Bonabeau, G. Theraulaz, and M. Dorigo. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, USA, 1999.
- [2] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.
- [3] Marco Dorigo, Vittorio Maniezzo, and Alberto Colomi. Positive feedback as a search strategy. Technical Report 91–016, Politecnico di Milano, 1991.
- [4] Paulo R. Ferreira, Jr., Felipe Boffo, and Ana L. C. Bazzan. A swarm based approximated algorithm to the extended generalized assignment problem (E-GAP). In *Proceedings of the 6th International Joint Conference on Autonomous Agents And Multiagent Systems (AAMAS)*, pages 1231–1233, May 2007.
- [5] Paulo Roberto Ferreira, Jr., Fernando dos Santos, Ana L. C. Bazzan, Daniel Epstein, and Samuel J. Waskow. Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies. *Autonomous Agents and Multi-Agent Systems*, 20:421–443, May 2010.
- [6] N.R. Hoff, A. Sagoff, R.J. Wood, and R. Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pages 123 –130, 2010.
- [7] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, (1):7–38, 1998.
- [8] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, volume 6, pages 739–743, Tokyo, Japan, October 1999. IEEE.
- [9] Yan Meng, O. Kazeem, and J.C. Muller. A hybrid aco/pso control algorithm for distributed swarm robots. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 273–280, 2007.
- [10] David Payton, Regina Estkowski, and Mike Howard. Compound behaviors in pheromone robotics. *Robotics and Autonomous Systems*, 44(3–4):229–240, 2003.
- [11] Vagelis Plevris, Matthew G. Karlaftis, and Nikos D. Lagaros. A swarm intelligence approach for emergency infrastructure inspection scheduling. In Kasthurirangan Gopalakrishnan and Srinivas Peeta, editors, *Sustainable and Resilient Critical Infrastructure Systems*, volume 1, pages 201–230. Springer Berlin, Berlin, 2010.
- [12] A.H. Purnamadjaja and R.A. Russell. Pheromone communication: implementation of necrophoric bee behaviour in a robot swarm. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 2, pages 638–643, 2004.

- [13] Fernando dos Santos and Ana L. C. Bazzan. Towards efficient multiagent task allocation in the robocup rescue: a biologically-inspired approach. *Autonomous Agents and Multi-Agent Systems*, 22:465–486, May 2011.
- [14] Olivier Simonin and François Charpillet. Indirect cooperation between mobile robots through an active environment. In *5th National Conference on "Control Architecture of Robots"*, pages 71–80, 2010.
- [15] C. Skinner and M. Barley. Robocup rescue simulation competition: Status report. In Ansgar Bredenfeld, Adam Jacoff, Itsuki Noda, and Yasutake Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 632–639. Springer-Verlag, Berlin, 2006.
- [16] M. J. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester, 2002.

Appendix A

scenario	pherom. units	evaporation coefficient				
		0	0.25	0.5	0.75	1
20_1	0	0.308 ± 0.015	0.312 ± 0.015	0.314 ± 0.019	0.320 ± 0.023	0.302 ± 0.014
	1	0.340 ± 0.023	0.348 ± 0.024	0.336 ± 0.025	0.337 ± 0.016	0.315 ± 0.027
	5	0.350 ± 0.021	0.348 ± 0.038	0.342 ± 0.038	0.320 ± 0.020	0.327 ± 0.030
	10	0.330 ± 0.031	0.343 ± 0.024	0.347 ± 0.015	0.342 ± 0.021	0.315 ± 0.024
20_5	0	0.255 ± 0.013	0.261 ± 0.021	0.258 ± 0.020	0.252 ± 0.022	0.259 ± 0.015
	1	0.267 ± 0.038	0.253 ± 0.027	0.274 ± 0.028	0.254 ± 0.028	0.258 ± 0.021
	5	0.262 ± 0.023	0.287 ± 0.031	0.267 ± 0.027	0.265 ± 0.019	0.260 ± 0.019
	10	0.262 ± 0.026	0.267 ± 0.017	0.271 ± 0.034	0.266 ± 0.023	0.262 ± 0.023

Table 1. Results for each *Kobe4* scenario, showing the average building area left for each pheromone unit and evaporation coefficient (grey cells indicate the best score)

Simulação Multiagente Interativa no Ambiente SIMULA

Marcos Paulo Martins da Silva¹, Daniela Bagatini¹, Rejane Frozza¹

¹UNISC - Universidade de Santa Cruz do Sul

Departamento de Informática

Av. Independência, 2293 – Santa Cruz do Sul - RS

marcospaulomartinsdasilva@yahoo.com.br; {bagatini, frozza}@unisc.br

Abstract. To facilitate the development of multiagent systems, was implemented an environment to develop reactive agents, called SIMULA. This environment aims reduce the programming effort in creating user applications using a graphical interface that allows specify the agents and how they will act in resolution of a problem. This paper presents aspects of interactivity in the SIMULA environment, allowing users to analyze different situations in the simulation process.

Resumo. Para facilitar o desenvolvimento de sistemas multiagentes, foi concebido um ambiente de desenvolvimento de agentes reativos com propósito didático chamado SIMULA. Este ambiente tem como objetivo principal diminuir o esforço de programação do usuário na criação de suas aplicações, por meio de uma interface gráfica que permite especificar os agentes e como eles agirão na resolução de um determinado problema. O objetivo principal deste trabalho foi desenvolver aspectos de interatividade no ambiente SIMULA, permitindo aos usuários analisarem diferentes situações no processo de simulação da modelagem realizada.

1. Introdução

O avanço da tecnologia de construção de agentes inteligentes tornou ainda mais atraente a área de simulação. É possível simular situações como: um jogo de futebol, um robô coletor de materiais em Marte, tráfego aéreo e urbano, catástrofes naturais, situações ambientais diversas, entre outras, ou seja, qualquer problema real, simples ou complexo, com o intuito de melhorar o processo decisório em relação aos resultados atingidos (Rezende, 2003).

Como existem problemas complexos que estão além das capacidades individuais dos agentes, foram concebidos os Sistemas Multiagentes. Segundo (Rabelo, 2009), estes sistemas consistem de uma aplicação distribuída, composta por um conjunto de agentes que cooperam entre si para a solução de um problema. Geralmente, são utilizados os agentes reativos em sistemas de simulação, por serem capazes de perceber seu ambiente e responder rapidamente às mudanças que ocorrem nele, tendo como princípio satisfazer seus objetivos (Wooldridge, 2002).

Os ambientes de desenvolvimento de sistemas multiagentes reativos permitem a simulação de eventos reais. Entre os ambientes disponíveis, destaca-se o Sistema SIMULA (Frozza, 1997), desenvolvido para fins didáticos, sendo utilizado para simulação de problemas reais, onde o usuário define a situação inicial do problema e os

possíveis comportamentos dos agentes. O ambiente executa a simulação e apresenta uma situação final atingida pela atuação dos agentes.

O objetivo principal deste trabalho foi tornar o SIMULA um ambiente de simulação multiagente que possibilita ao usuário interagir nos resultados das simulações, ou seja, permitir alterações/manipulações diretamente no cenário apresentado pelo SIMULA, o que não era possível em versão anterior do sistema. Desta forma, evitam-se problemas como o aglomerado de agentes, por exemplo, modificando o resultado da simulação. Outras alterações foram: retirar agentes, impedir agentes de passar por um determinado caminho, deixar pista em determinadas posições, entre outras ações. A intervenção do usuário durante a simulação permite uma análise mais realista da atuação dos agentes no domínio modelado no ambiente.

O SIMULA é um ambiente utilizado pelos pesquisadores da área de Sistemas Multiagentes em suas disciplinas de Inteligência Artificial e projetos, visto que é um ambiente desenvolvido para fins educacionais e que trabalha muito bem com a questão de simulação multiagente para agentes reativos. Existem outros ambientes de simulação, como o NetLogo, que é um ambiente de simulação social. Mas que exigem dos usuários a programação em alguma linguagem. No SIMULA, o usuário fica livre da programação em linguagens, ele faz todas as especificações e o próprio ambiente gera código fonte (importante contribuição do SIMULA em relação a outros ambientes). Desta forma, o SIMULA como um ambiente de simulação interativo é uma contribuição para a área, que irá propiciar novas pesquisas.

Este artigo está organizado da seguinte forma: na seção 2, apresenta-se um breve panorama sobre ambientes de simulação multiagente; na seção 3, aborda-se sobre as novas funcionalidades que possibilitam simulação multiagente interativa no ambiente SIMULA; na seção 4 relata os resultados dos testes e detalhes de implementação e, por fim, são apresentadas as considerações finais.

2. Simulação Multiagente

Agentes e sistemas multiagentes estão sendo cada vez mais utilizados nas aplicações do dia a dia. Diretamente ou indiretamente, possibilitam a resolução de problemas complexos, como exemplo, no campo de simulação, com sistemas multiagentes reativos e cognitivos, na qual é possível uma simulação muito próxima da realidade, desde uma colônia de formigas até catástrofes ambientais.

Os modelos baseados em agentes consistem em interações entre os agentes e o ambiente. Mesmo em um modelo simples percebe-se um comportamento complexo que resulta destas interações com importantes informações sobre o funcionamento do sistema. Isto torna possível a exploração das conexões entre os comportamentos de indivíduos em seu micro nível e o macro nível de modelos que emergem da interação de uma sociedade destes indivíduos (Azevedo & Menezes, 2006). Entende-se por micro nível o sistema ou indivíduo isolado que possui um problema a solucionar de acordo com seu conhecimento e capacidade para resolvê-lo; e no macro nível, o sistema é considerado como uma sociedade de agentes que possuem conhecimento, capacidade de resolução limitada e interação (Werner & Demazeau, 1991).

2.1. Requisitos para um Ambiente de Simulação Multiagente

Em ambientes de simulação multiagente deve ser possível descrever o sistema a ser

modelado, a simulação computacional do sistema e os recursos para a análise e monitoramento do comportamento dos agentes. E ao definir-se a sociedade de agentes, devem ser utilizados os seguintes elementos (Azevedo & Menezes, 2006):

- A definição dos agentes: as características do agente, como sua aparência e forma, tamanho, quantidade, posição (pré-fixada ou aleatória) e cor para melhorar a visualização.
- A definição do comportamento: os elementos que definem o comportamento dos agentes são: movimento (como o agente se desloca no ambiente); percepção (como o agente percebe outros agentes e objetos no ambiente); colisão (o que o agente fará quando mais de um agente quiser ocupar a mesma posição do ambiente).
- A definição do ambiente: é o local onde o agente será posicionado e onde realizará suas ações e interações.
- A identificação dos agentes e definição das variáveis: as regras de comportamento são estabelecidas na identificação do agente, sendo possível a formação de conjuntos e classes de agentes. Ao descrever o modelo são utilizadas variáveis que estabelecem parâmetros entre os agentes e o modelo, sendo classificadas em tipos como: lógicas, globais, locais e vinculadas ao agente.

Com a definição destas características dos agentes e do ambiente a serem modelados, são necessárias ferramentas (ambientes de simulação) para analisar o modelo e seus dados comportamentais, facilitando a compreensão do processo.

2.2. Ambientes de Simulação Multiagente

Quase todos os ambientes de simulação multiagente são compostos de um conjunto de conceitos para o projeto e descrição de uma modelagem multiagente. Alguns ambientes de simulação existentes oferecem uma padronização dos projetos de *software* e ferramentas para simulação sem limitar a complexidade ou o tipo de modelo, porém são limitados pela sua dificuldade de uso, ou seja, o usuário deve ter conhecimento de programação; e também possuem ferramentas insuficientes para a construção, execução, observação e documentação dos modelos.

Foram escolhidos três ambientes multiagentes para serem apresentados neste trabalho: SWARM (Minar et al, 1996), NetLogo (Netlogo, 2010) e SIMULA (Frozza, 1997), a fim de contribuírem para análise de características a serem abordadas na proposta de melhorias do Ambiente SIMULA, para torná-lo um ambiente interativo para desenvolvimento de aplicações com agentes reativos.

Pelos estudos realizados sobre o ambiente SWARM, foi possível verificar que apesar de ser uma adequada ferramenta para simulações multiagente, destoa dos outros ambientes por não ter uma *interface* muito amigável e, principalmente, os usuários da ferramenta devem ter um bom conhecimento em linguagem de programação para criar seus modelos de simulação. Já o *NetLogo* e o SIMULA têm uma *interface* amigável e fazem uso de recursos que simplificam o desenvolvimento de modelos e visualização da simulação. O *NetLogo* possui como destaque uma grande biblioteca de modelos de simulação prontos para testes nas mais diversas áreas, porém o desenvolvimento de modelos é bastante trabalhoso, pois o usuário além de desenvolver o modelo de simulação, deve montar toda a interface de visualização da simulação; e o SIMULA por ser voltado mais para uma conceitualização didática, não possui tantos recursos como o *NetLogo*, mas possui uma interface muito simples e prática, bem como recursos de

geração de código que possibilitam ao usuário rapidez e praticidade no desenvolvimento de modelos de simulação.

2.3. Trabalhos Relacionados

Atualmente o campo de modelagem baseada em agentes tem sido amplamente utilizado em simulações das mais diversas áreas de domínio. A seguir serão descritos alguns trabalhos relacionados à simulação para mostrar as diferentes áreas nas quais a simulação pode ser utilizada.

O trabalho de Sistema de Simulação e Controle de Tráfego (Cancian & França, 2007) é composto de um sistema para simulação e controle de tráfego urbano, que utiliza sistemas multiagentes e sistemas especialistas organizando uma estrutura hierárquica de agentes, onde um agente principal comanda os outros agentes que são os veículos. Entre os veículos há ambulâncias que atendem a chamados de emergência. Os agentes utilizados nesta simulação basicamente são agentes reativos. Somente o agente gerente poderia ser considerado híbrido, devido a sua atuação. Os sistemas especialistas são exclusivamente utilizados na definição das regras para os agentes.

O trabalho de Sistema de Simulação da Criminalidade (Furtado, 2008) tem o objetivo de fortalecer a hipótese de que o estudo e o desenvolvimento dos sistemas multiagentes inspirados em sistemas biológicos possui uma estratégia adequada para modelar o comportamento criminal. A modelagem computacional de crimes tem o objetivo de produzir simulações para se obter uma melhor compreensão da criminalidade.

O objetivo do trabalho de Sistema de Simulação de Futebol (Bagatini, 2001) foi o desenvolvimento de um time de futebol para o simulador *Soccerserver*, ou seja, desenvolver agentes jogadores que demonstrem um nível considerável de competência na realização de tarefas, como: percepção, ação, cooperação, estratégias pré-definidas, decisão e previsão.

Através destes trabalhos foi possível analisar como foram construídas as simulações, quais foram seus objetivos, como se comportam os agentes, que benefícios trouxeram para os pesquisadores, a fim de aplicar simulações em domínios reais.

Os trabalhos relacionados foram escolhidos por utilizarem agentes reativos, que foram o foco da pesquisa.

3. Simulação Multiagente Interativa no Ambiente SIMULA

Como abordado na Introdução, a área de simulação multiagente está em expansão em diversos domínios, como, por exemplo, simulação de catástrofes naturais e de situações do cotidiano de grandes cidades. E para estas simulações serem desenvolvidas, existem ambientes de simulação (apresentados na seção 2).

O SIMULA é um ambiente utilizado como plataforma na área de sistemas multiagentes em disciplinas como Inteligência Artificial, ou seja, é um ambiente desenvolvido para fins educacionais que trabalha muito bem com a questão de simulação multiagente para agentes reativos. Desta forma, um dos propósitos deste trabalho foi torná-lo um ambiente de simulação interativo, como uma nova contribuição para a área, propiciando novas pesquisas.

Devido a estes motivos e através deste trabalho, possibilitou-se a interação do usuário nos resultados das simulações realizadas pelo SIMULA, permitindo alterações/manipulações diretamente na simulação sem alterar as definições prévias do modelo como, por exemplo: retirar agentes, impedir agentes de passar por um determinado caminho, não deixar pista em determinadas posições, entre outras ações.

3.1. Novas Funcionalidades

A seguir, descrevem-se as novas funcionalidades que foram desenvolvidas para o SIMULA.

3.1.1. Log da Simulação

Todas as simulações executadas no SIMULA são visualizadas na tela de “Execução” do ambiente, onde o usuário pode acompanhar e analisar o comportamento dos agentes no ambiente. Para que este usuário possa analisar a simulação com diferentes parâmetros e com diferentes enfoques das atuações dos agentes, foi criado um “log” que armazena dados da simulação. Desta forma, o usuário tem a possibilidade de analisar as simulações geradas e chegar a uma simulação próxima do modelo pretendido.

O arquivo de “log” do SIMULA é um arquivo de texto criado em tempo de execução, onde são gravadas informações da simulação em andamento como (Figura 1):

- Horário de início da simulação: quando o usuário clicar no botão “Iniciar” pela primeira vez, o horário do sistema será gravado no início do arquivo de “log”.
- Horário de término da simulação: quando o usuário clicar nos botões “Log” e “Sair”, a simulação será terminada e o horário do sistema será gravado no final do arquivo de “log”.
- Tempo total da simulação em ciclos: como a simulação é baseada em ciclos, é feita uma contagem do total de ciclos percorridos pela simulação e gravado no final do arquivo de “log”.
- Horário de cada ciclo: para ter-se uma ideia de quanto tempo cada ciclo leva, é gravado no arquivo de “log” o horário do sistema em que cada ciclo inicia.
- Ações e reações de cada agente em cada ciclo: são gravadas, no arquivo de “log”, todas as ações e reações que cada agente inserido na simulação executa em cada ciclo.

Com base nestas informações que constituem o arquivo de “log”, após o término de cada simulação, o usuário pode ter um histórico da simulação e arquivá-la para futuras comparações e estudos, pois através deste arquivo o usuário pode visualizar em modo texto a simulação sem ter visto a sua execução. Outra contribuição do arquivo de “log” do SIMULA é de possibilitar ao usuário medir o desempenho dos agentes em suas tarefas, sendo útil para alocar determinadas tarefas a determinados agentes e, desta forma, trabalhar este aspecto de tempo nas ações dos agentes, tornando possível mensurar a capacidade dos agentes executarem tarefas em determinado tempo.

```
Início da simulação: 15:40:13.229.  
  
azul percebe vermelho  
azul executa movimento randômico  
vermelho percebe azul  
vermelho executa movimento randômico  
Ciclo 1: 15:40:13.432.  
  
azul percebe vermelho  
azul executa movimento randômico  
vermelho percebe azul  
vermelho executa movimento randômico  
Ciclo 2: 15:40:13.634.  
  
azul percebe vermelho  
azul executa movimento randômico  
vermelho percebe azul  
vermelho executa movimento randômico  
Ciclo 3: 15:40:13.837.  
  
azul percebe vermelho  
azul executa movimento randômico  
vermelho percebe azul  
vermelho executa movimento randômico  
Ciclo 4: 15:40:14.40.  
  
Fim da simulação: 15:40:37.768.  
  
Tempo da simulação: 4 ciclos.
```

Figura 1. Arquivo de Log.

Foi criado um novo botão na tela de execução do SIMULA chamado de “Log”, onde, ao clicar, a simulação é terminada e é aberta uma caixa de “Salvar como” para que o usuário possa salvar o arquivo de “log” em qualquer pasta de seu computador e com o nome do arquivo que desejar.

3.1.2. Evitando o Embolo

No ambiente SIMULA, após o usuário definir os agentes, ele parte para a etapa de definição das regras de comportamento destes agentes, onde o ambiente fornece uma biblioteca de funções para a montagem e definição das ações dos agentes. Dentre estas funções, há uma denominada “movimento_randomico”, onde o agente realiza um movimento para uma direção (verticalmente, horizontalmente ou diagonalmente), definida por uma função randômica interna. Esta função foi verificada, pois parecia não funcionar muito bem, já que muitas vezes os agentes ficavam “embolados” ou intercalando entre duas posições, por exemplo.

O que ocorria era que estava sendo utilizada uma função randômica do Java chamada “*Math.random()*” que randomizava as direções possíveis do agente se movimentar pelo ambiente. Esta função tem uma falha que faz com que ela vicie, ou seja, gere a mesma sequência de direção para os agentes por várias vezes. Para corrigir este problema foi utilizada a mesma função, porém com a condição que armazena a direção anterior e compara com a nova direção gerada. Caso esta direção seja a mesma, ela gera outra até ser diferente, fazendo com que o agente não repita os mesmos movimentos, permitindo uma melhor movimentação dos agentes no ambiente, favorecendo as suas ações na simulação.

3.1.3. Movendo Agentes

Quando a simulação é inicializada, é possível visualizá-la e, agora com este trabalho, interferir durante o processo da simulação. O processo de interatividade se dá na fase de

execução da simulação, onde o usuário pode mover os agentes de e para qualquer parte do ambiente, alterando a simulação. Para isto, é necessário que o usuário pare a simulação, clique com o botão direito do *mouse* sobre o agente escolhido, onde aparecerá um menu *popup* com algumas opções. Ao selecionar a opção “Mover” (Figura 2), o cursor do *mouse* mudará para o formato de uma mão e aguardará que o usuário selecione o novo local para o agente, movimentando o cursor para o local desejado no ambiente e clicando com o botão esquerdo do mouse, fazendo com que o agente vá para este local e o cursor do mouse volte para seu estado normal.

3.1.4. Excluindo Agentes

Além da movimentação dos agentes, também é possível a sua exclusão. Para isto, deve-se fazer o mesmo procedimento anterior, para abrir o menu *popup*, e em seguida selecionar a opção “Excluir” para excluir o agente da simulação, conforme Figura 2.

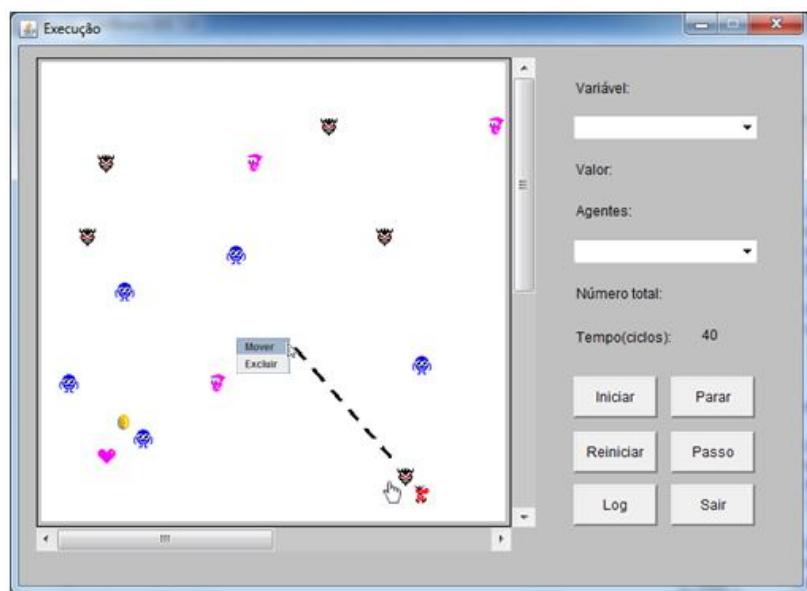


Figura 2. Interação no SIMULA.

3.1.5. Deixando Pistas

Outro processo de interatividade no SIMULA, baseia-se nas pistas, onde o usuário pode colocar e retirar pistas no ambiente de simulação, pistas estas necessárias em algumas simulações para que agentes as utilizem como meio de localização no ambiente. Para isto, é necessário que o usuário pare a simulação, clique com o botão direito do *mouse* sobre algum ponto vago do ambiente, ou seja, um lugar que não esteja ocupado por um agente ou obstáculo, onde aparecerá um menu *popup* com a opção “Pista”. Após esta opção ser selecionada e clicada, deixará uma pista naquele local apontado. Para a exclusão de uma pista é necessário fazer o mesmo procedimento anterior, porém deve-se clicar sobre uma pista e ao aparecer o menu *popup* selecionar e clicar na opção “Exclui”, opção esta que apagará a pista do ambiente.

3.1.6. Evitando o Bolo

Em algumas circunstâncias (na execução da simulação), determinados grupos de agentes aglomeram-se em volta de algo (por exemplo: um obstáculo), trancando o

sistema e causando o chamado “bolo”, “anomalia” esta, corrigida, com a intervenção do usuário, não necessitando da reinicialização da simulação para tentar contornar o problema.

Agora que o ambiente permite estas alterações/manipulações diretamente na simulação sem alterações no modelo de simulação como: “tirar um agente”; “não deixar um agente passar por um caminho”; “aqui não deixar pista”; entre outras, possibilita ao usuário fazer descobertas que não eram possíveis anteriormente, como por exemplo: retirar um agente de um determinado local e colocá-lo em outro, qual será a reação deste agente e dos outros? Com esta interatividade, o usuário terá novas possibilidades, deixando sua simulação cada vez próxima do modelo pretendido. Para a área de simulação multiagente, a característica de interatividade é o ponto principal para gerar diferentes situações e configurações dos agentes e sua atuação, a fim de aproximar a simulação da realidade modelada.

A cada início da execução da simulação, se os agentes não tiverem sua posição inicial pré-definida pelo usuário, o SIMULA randomicamente os posicionará no ambiente. Em alguns casos, alguns agentes não aparecem no ambiente, porque talvez tenham sido colocados na mesma posição de outros agentes. Esta situação ainda será verificada, a fim de não ocasionar prejuízos pela falta de agentes no processo de simulação.

4. Implementação e Testes

Como o ambiente SIMULA foi desenvolvido em Java e o trabalho em si teve foco na parte de execução do ambiente, a linguagem de programação permaneceu em Java e o *IDE* utilizado foi o *NetBeans 7.0*. O SIMULA é composto de vários arquivos “.java”. Os arquivos que sofreram alteração foram:

- **SIMULA.java:** Que foi alterado somente na saída do programa, ou seja, agora quando o usuário decidir sair do ambiente, abrirá uma caixa de diálogo perguntando se ele deseja salvar a simulação.
- **Metodos.java:** Possui todos os tipos de ações que os agentes podem executar na simulação. Em cada função, que agrupa cada tipo de ação dos agentes, foi inserida uma chamada à função “GravaS”, que encontra-se no arquivo “Executar.java”, que extrai da função o agente, o alvo (se houver) e a ação executada pelo agente, e grava no arquivo de *log*.
- **Executar.java:** Arquivo principal da tela de execução do ambiente. Neste arquivo foram desenvolvidas todas as funções para transformar o simula em um ambiente interativo.

Para o arquivo de *log* foi utilizado um arquivo de texto, que é criado em tempo de execução, sofre várias alterações (escritas) e é finalizado. O arquivo é criado assim que o botão “Iniciar” é pressionado na tela de execução do ambiente e finalizado ao clicar nos botões “Log” ou “Sair”. Foi criada uma função “Hora” para buscar a hora do sistema, sendo utilizada para gravar no arquivo de *log* os horários de início e fim da simulação, bem como o tempo de cada ciclo de simulação. Já existe uma variável “tempo” que contém o número de ciclos da simulação, a qual é utilizada também para mensurar o tempo da execução em ciclos da simulação e o início de cada ciclo.

Para o usuário interagir com o ambiente, seja selecionando um agente para

movê-lo, ou uma pista a ser inserida em determinado local, foram utilizados os eventos de cliques do mouse para registrar sua posição na tela e uma função que retornava o que havia na posição clicada, para que um menupopup exibisse as opções possíveis ao que a função havia retornado. Para fazer a exclusão de um agente, foi utilizada a função “morteDeAgente”, que já existia no código original, porém todo o código “novo” foi desenvolvido com um prévio e minucioso estudo do código original em conjunto com as variáveis e funções já existentes no código.

Como exemplo para testes foi utilizada uma simulação constituída pelos agentes azul, vermelho, preto e filhote (Figura 3). Os agentes azul e vermelho iniciam com movimento randômico e quando percebem um ao outro, movem-se para encontrarem-se, e ao encontrarem-se formam um coração, que por sua vez transforma-se em um ovo, que em certo tempo dá origem a um filhote. Caso o filhote perceba o agente preto, foge. O agente preto por ter sua área de percepção reduzida em relação a dos outros agentes, tem o auxílio de pistas, que podem ser inseridas pelo usuário durante a simulação, pois os agentes pretos ao perceberem os filhotes, os perseguem e os matam.

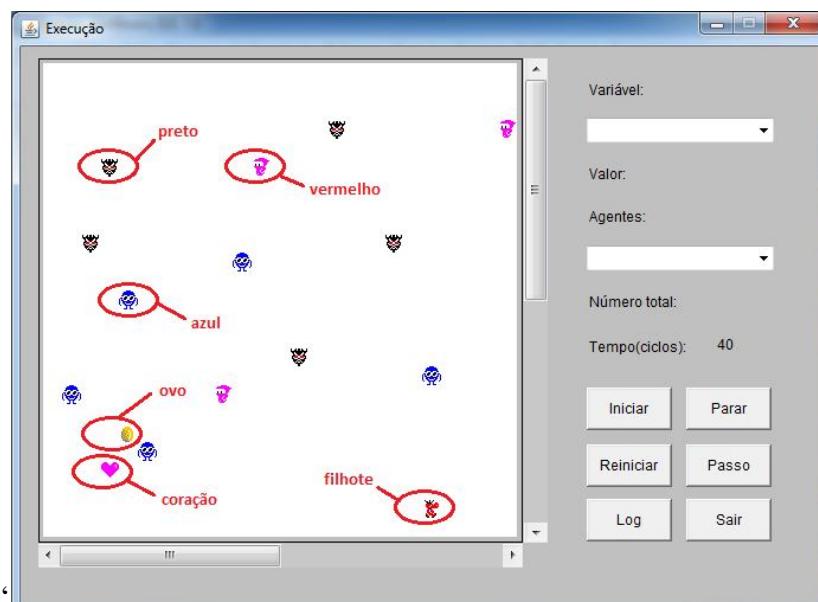


Figura 3. Apresentação dos agentes.

Na versão anterior do SIMULA para que o usuário pudesse assistir a morte de um filhote, ele teria que aguardar até que um filhote nascesse perto de um agente preto, o que em alguns casos poderia demorar pouco ou muito tempo, dependendo do encontro dos agentes azuis e vermelhos. Porém, na nova versão interativa do SIMULA, o usuário tem opções como: colocar uma pista entre um agente preto e o filhote, mover o filhote próximo a um agente preto ou mover um agente preto próximo ao filhote. Assim, o usuário pode intervir na simulação acelerando ou retardando algum acontecimento, movendo (Figura 4) o filhote ou o agente preto, ou ainda, colocando uma pista (Figura 5) na direção do filhote, para que os agentes pretos a percebam.

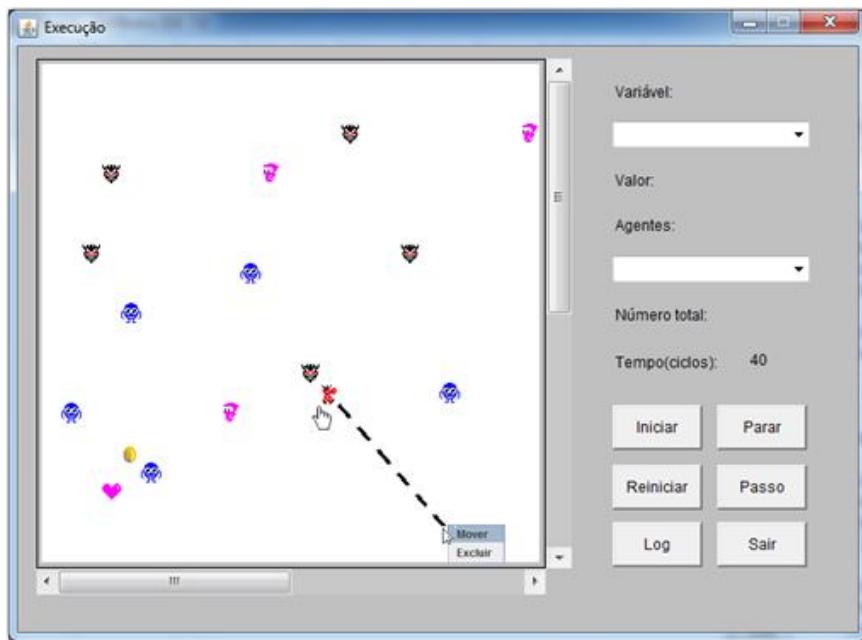


Figura 4. Movendo agente filhote.

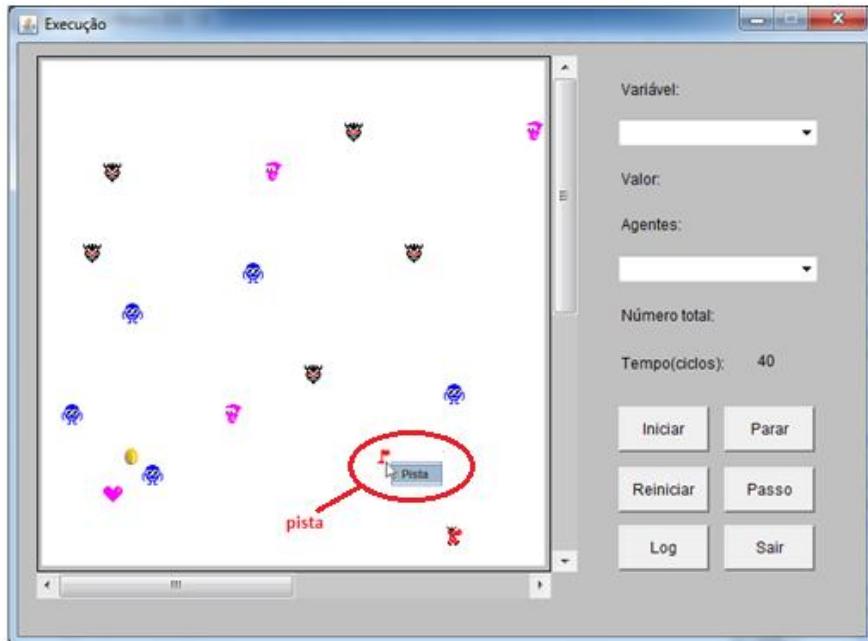


Figura 5. Inserção de pista.

A partir dos testes realizados com a simulação utilizada no exemplo apresentado, pode-se ter uma ideia de uma série de possibilidades que os usuários terão em relação à versão anterior do SIMULA, com a utilização da interatividade durante o processo de simulação. Como visualizado na simulação, além de acelerar processos da execução natural da simulação, como a morte do agente filhote, pode-se evitar vários problemas ocorridos anteriormente como o bolo e embolo de agentes e novas funcionalidades

como deixar pistas, excluir e mover agentes.

5. Conclusões

Com base nos estudos realizados com agentes e sistemas multiagentes, verificou-se que a área de simulação destina-se à solução de diversos problemas e, principalmente, a imitar a realidade de uma forma computacional buscando analisar situações diversas.

No trabalho, atingiram-se os objetivos da proposta destinados à interatividade, contribuindo para a área de inteligência artificial com o aprimoramento do ambiente SIMULA e principalmente na inserção da interatividade do usuário no ambiente, permitindo-lhe chegar a novas descobertas através desta ferramenta de fácil manuseio.

Em relação aos ambientes citados no artigo, o SIMULA tornou-se ainda mais distante do ambiente SWARM em termos de praticidade no desenvolvimento e execução da simulação, pois o SWARM além de não ter uma *interface* muito amigável, os usuários da ferramenta devem ter um bom conhecimento em linguagem de programação. Já em comparação com o *NetLogo*, o SIMULA por ter uma adequada *interface* e fazer uso de recursos que simplificam o desenvolvimento de modelos, bem como a visualização e interatividade do usuário no decorrer da simulação, aproxima-se um pouco do *NetLogo* em recursos, mas sempre mantendo a praticidade no desenvolvimento de modelos de simulação, por ser voltado mais para a área acadêmica.

Como trabalhos futuros, sugere-se o desenvolvimento de um “*parser*”, pois ao realizar todos os passos de configuração da simulação, o usuário deve gerar o código da aplicação antes de executá-la e há casos em que em um dos passos anteriores, pode ter havido a montagem de uma regra ou a definição de um agente de forma incorreta, causando erros na geração do código, que não são detalhados ao usuário. Portanto, é interessante o desenvolvimento de um “*parser*”, ou seja, um indicador de erros, que identifique ao usuário onde se encontra o erro, facilitando a correção de erros no processo de modelagem da simulação. Também, considerar a inserção de regras de comportamento para os agentes no momento da execução da simulação, diferentes das já definidas na modelagem.

REFERÊNCIAS

- Azevedo, L. L.; Menezes, C. S. (2006) AProSiMA: Ambiente para Resolução Cooperativa de Problemas Baseado em Simulação Multiagente, http://www.rc.unesp.br/serp/trabalhos_completos/completo2.pdf.
- Bagatini, Daniela D. S. (2001). Um Sistema Multiagente para o Simulador Soccerserver, <http://www.lume.ufrgs.br/handle/10183/1650>.
- Cancian, M. H.; França, R. B.. (2007) Uma Abordagem de Simulação e Controle de Tráfego Utilizando Sistemas Multiagente. Florianópolis – SC, http://www.das.ufsc.br/~gb/pg-ia/Jess07/relatorio_sma_maiara_ricardo.pdf.
- Frozza, R. (1997) SIMULA: Ambiente para Desenvolvimento de Sistemas Multiagentes Reativos, www.lume.ufrgs.br/bitstream/handle/10183/17923/000100248.pdf.
- Furtado, V. (2008) Modelagem e Simulação Multiagente da Criminalidade, <http://funcapciencia.funcap.ce.gov.br/artigos/artigos/anexos/Sistemas%20Complexos%20e%20Simulacao.pdf>.

- Minar, N.; Burkhart, R.; Langton, C.; Askenazi, M. (1996) The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations, <http://www.swarm.org/images/b/bb/MinarEtAl96.pdf>.
- Netlogo. (2010) NetLogo User Manual, <http://ccl.northwestern.edu/netlogo/docs/>.
- Rabelo, R. J. (2009) Projeto de Sistemas Multiagentes, <http://www.das.ufsc.br/~rabelo/Ensino/DAS6607/>
- Rezende, S. O. (Org.). (2003) Sistemas inteligentes: fundamentos e aplicações. Barueri: Manole.
- Werner, E.; Demazeu, Y. (1991) European Workshop on Modeling Autonomous Agents in a Multi-agent World, 3., Kaiserslautern. Proceedings, Germany.
- Wooldridge, M. J. (2002) An introduction to multiagent systems. Chichester: John Wiley& Sons.

Segregação Sócio-espacial: um Estudo utilizando Sistemas Multiagentes

**Carlos Eduardo Pereira de Quadros, Josimara de Ávila Silveira,
Felipe Neves da Silva, Leonardo Martins Rodrigues, Liliane Silva Antqueira,
Stephanie Loi Brião, Suvania Acosta Oliveira, Tauã Milech Cabreira**

¹Programa de Pós Graduação em Modelagem Computacional

Universidade Federal do Rio Grande (FURG)

Av. Itália, Km 8 – Campus Carreiros – 96.201-900 – Rio Grande – RS – Brazil

Abstract. *This work is a study about the Social-Spatial Segregation that occurs, most often, unintentionally in some cities, especially in large population centers. To model this we used multi-agent simulation through the software NetLogo. With an existing example in its library of templates, we developed some examples with variations for the case of social classes segregation. Finally, we present the results of this study.*

Resumo. *Neste trabalho é realizado um estudo sobre a Segregação Sócio-espacial que ocorre, na maior parte das vezes, de forma involuntária em algumas cidades, principalmente em grandes centros populacionais. Para isso, utiliza-se simulação multiagente através do software NetLogo. Através de um exemplo existente em sua biblioteca de modelos, foram desenvolvidos exemplos com variações para o caso da Segregação de classes sociais. Por fim, são mostrados os resultados obtidos nesse estudo.*

1. Introdução

A segregação consiste no ato de segregar, separar ou isolar um determinado grupo de indivíduos. A expressão espacial da segregação é a “área natural”, definida por Zorbaugh como sendo uma área geográfica caracterizada pela individualidade física e cultural, resultante do processo de competição impessoal que geraria espaços de dominação dos diferentes grupos sociais, replicando ao nível da cidade os processos que ocorrem no mundo vegetal [Corrêa 1993].

Entre os diversos tipos de segregação existentes, podem-se destacar a segregação urbana, também conhecida como a segregação de diferentes grupos étnico-culturais e/ou diferentes grupos econômicos (classe alta, média e baixa) e a segregação de nacionalidades, que muitas vezes está diretamente relacionada ao conceito de xenofobia, exercido por grupos que discriminam outras pessoas devido a sua nacionalidade.

Em relação à segregação de nacionalidades, pode-se exemplificar a separação de latinos (cubanos, porto-riquenhos, mexicanos, etc.) e americanos que ocorre em diversos bairros de cidades dos Estados Unidos. Certos bairros destas localidades são ocupados quase que única e exclusivamente por pessoas de determinada nacionalidade ou descendência, configurando um tipo de segregação presente em diversas regiões do mundo.

Outro tipo de segregação amplamente conhecida é a segregação racial adotada pelo regime do Apartheid, na África do Sul. O regime, que vigorou entre os anos de 1948

e 1994, classificava a população em quatro nações distintas: brancos, negros, mestiços e indianos. O governo, formado pela minoria branca, detinha o poder e elaborava leis que cada vez mais suprimiam o direito à liberdade das demais “nações”. Dentre as principais leis que vigoraram durante o regime, pode-se destacar a Lei de Reserva dos Benefícios Sociais, que especificava locais públicos a serem frequentados por determinadas raças, caracterizando assim uma segregação múltipla, não apenas étnica, mas também espacial.

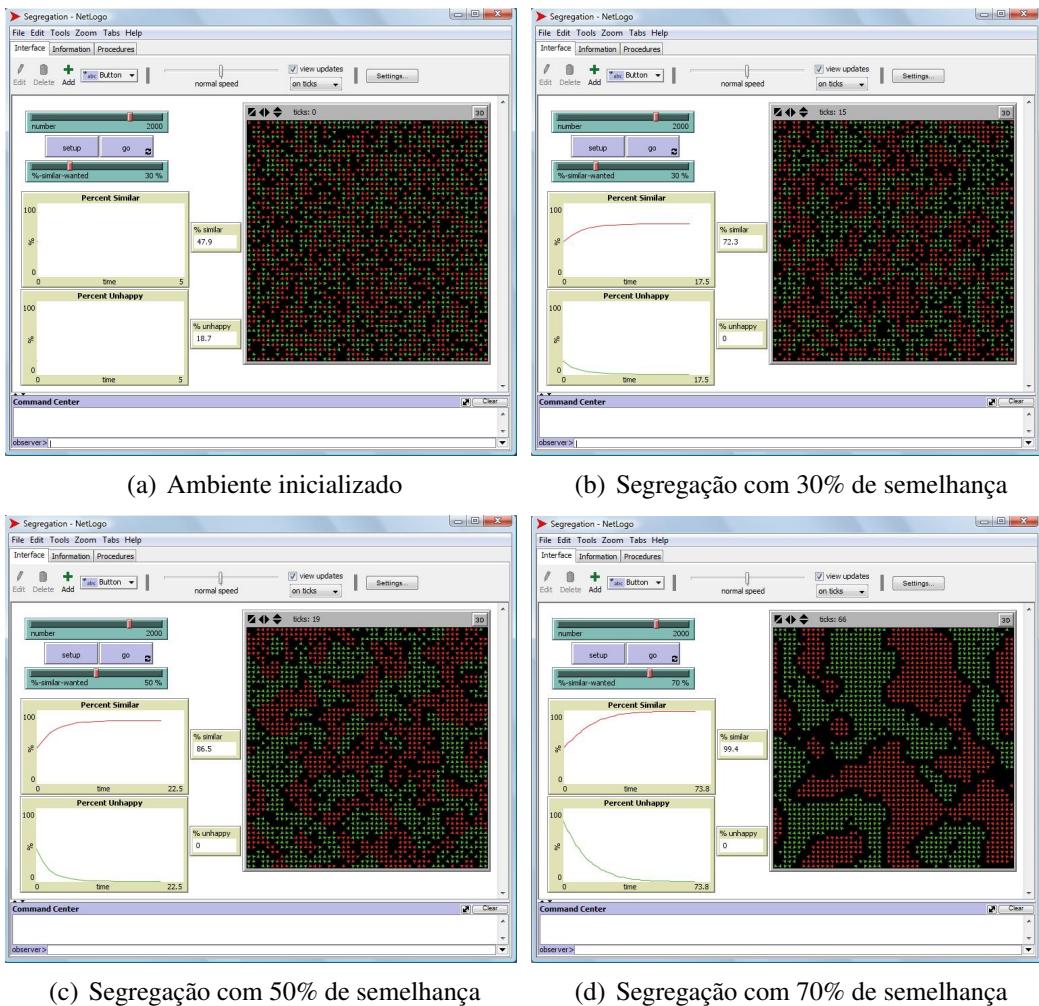
E por fim temos a segregação das classes sociais. Nas metrópoles brasileiras, uma das características mais marcantes é a segregação espacial das classes sociais em áreas distintas da cidade [Villaça 2001]. Este tipo de segregação traz inúmeros problemas às cidades. O primeiro é, obviamente, a desigualdade em si. Camadas mais pobres da população, com menos recursos, são justamente as que gastam mais com o transporte diário, que têm mais problemas de saúde por conta da falta de infraestrutura, que são penalizadas por escolas de baixa qualidade, e assim por diante. A própria segregação é não apenas reflexo de uma condição social, mas um fator que contribui para tornar as diferenças ainda mais profundas. Este tipo específico de segregação urbana pode ser dividido em “segregação voluntária” e “segregação involuntária” [Saboya 2001]. A segregação voluntária refere-se àquela na qual o indivíduo ou uma classe de indivíduos busca, por iniciativa própria, localizar-se próximo a outras pessoas da mesma classe social. Por sua vez, a segregação involuntária corresponde àquela em que as pessoas são segregadas contra a sua vontade, por falta de opção.

Este trabalho visa abordar o problema de segregação, voluntária e involuntária, aplicado às classes sociais nas zonas urbanas, através dos processos de simulação social. O objetivo do artigo é simular a população de uma metrópole e os processos de segregação de classes sociais através da modelagem baseada em sistemas multiagentes. No desenvolvimento desse projeto utilizou-se o software NetLogo [Wilensky 2011] para a simulação da sociedade. Os elementos desse software, aplicados ao problema, serão as *turtles* (agentes), simbolizando as pessoas, e os *patches* (ambiente), simbolizando a metrópole. Além disso, os agentes serão compostos de regras de comportamento individuais que definirão as suas ações no modelo.

2. Modelo *Segregation* na ferramenta NetLogo

Nos anos 70, Thomas Schelling desenvolveu um modelo de segregação racial baseado em suas observações de bairros norte-americanos, onde, em alguns casos, cerca 75% da população era de uma mesma raça. Poderia imaginar que isso se dava pelo racismo, e classes eram isoladasumas das outras. Mas Schelling tinha outra suspeita, de que essa segregação era formada a partir da interação entre os indivíduos, os quais decidiam viver próximos a seus semelhantes por opção própria e não por serem excluídos de outro lugar [Wilensky 1997]. Alguns resultados de segregação resultam de práticas de organizações, enquanto outros resultam a partir de escolhas individuais relacionadas à discriminação. Essa separação também pode ocorrer devido à sistemas de comunicação especializados, como as diferentes linguagens. Além disso, algumas segregações são o corolário de outros modelos de segregação: a residência está correlacionada com a localização do trabalho e o transporte [Schelling 1971].

O modelo de segregação presente na biblioteca de modelos da ferramenta NetLogo é baseado nas idéias de Schelling. Duas classes de indivíduos, uma de cor verde e outra

**Figura 1. Modelo “Segregation” na ferramenta NetLogo**

de cor vermelha, convive bem entre si, mas cada indivíduo quer ter certeza que vive perto de pelo menos alguns de seus semelhantes para estar feliz naquele lugar [Rauch 2002].

A simulação mostra que se formam grupos de indivíduos de uma mesma cor, o tamanho dos grupos formados varia de acordo com a porcentagem de vizinhos que um indivíduo necessita para ser feliz. A Figura 1 mostra telas do ambiente de simulação. Na primeira imagem, aparece o modelo inicializado com os agentes dispostos aleatoriamente pelo ambiente. Na segunda tela, aparece uma simulação finalizada com porcentagem de vizinhos necessários (30%) para que o agente seja feliz. Já na terceira e na quarta tela, as porcentagens são de 50% e 70%, respectivamente.

Na próxima Seção serão apresentadas alterações nesse modelo, onde é analisada a região do ambiente que pode ser habitada por agentes com determinadas faixas de renda fora de sua vizinhança.

3. Segregação de Classes Sociais

O trabalho demonstra a aplicação de um modelo comportamental por distribuição de agentes em um determinado espaço, como representação de um modelo habitacional, assumindo determinadas regras através de uma simulação sócio-econômica.

Em todas as metrópoles, observa-se a tendência de organização social do território expressar diferenças étnicas, raciais e sócio-econômicas, formando unidades de vizinhança que agrupam domicílios com características particulares [Katzman and Ribeiro 2008].

A seguir, serão mostrados os exemplos implementados com relação à modelagem da segregação de classes sociais, todos contando com a existência de três classes sociais.

3.1. Exemplo 1

O comportamento de cada indivíduo desta população será controlado por duas regras básicas: semelhança dos vizinhos e restrição econômica da região. Ao satisfazer as duas regras, o comportamento do indivíduo será o de permanecer no local onde se encontra. Caso contrário, o indivíduo irá deslocar-se em busca de um novo lugar. A partir desse comportamento local de cada indivíduo, o problema consiste em verificar e analisar a emergência ou não de um fenômeno global.

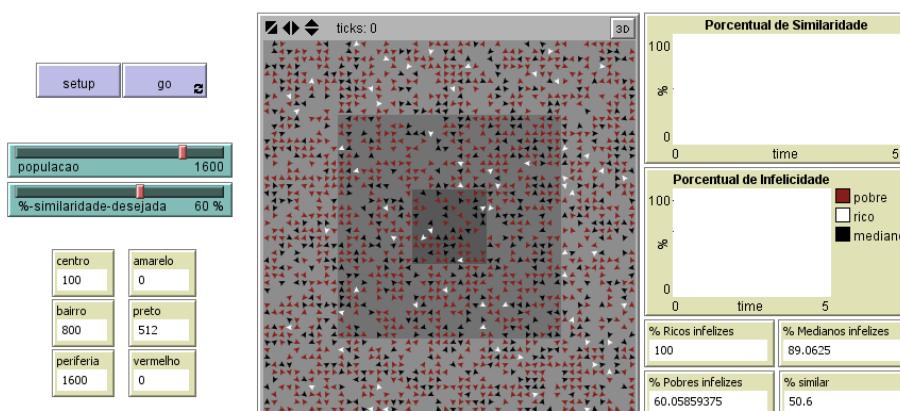


Figura 2. Interface do Modelo de Segregação das Classes Sociais

O Modelo de Segregação de Classes Sociais, apresentado na Figura 2, consiste na modelagem do espaço urbano e dos indivíduos que compõem este ambiente visando simular os processos de segregação social ocorridos nas metrópoles. O modelo estuda o comportamento de indivíduos de três classes sociais distintas, representados por agentes (*turtles*), em um ambiente dividido em três regiões com restrições econômicas. Nos agentes, cada cor representa um tipo específico de classe social: os agentes brancos pertencem à classe rica, os agentes pretos pertencem à classe média e os agentes vermelhos pertencem à classe baixa. Além disso, cada tartaruga detém um salário no intervalo correspondente a sua classe social.

O ambiente é dividido em faixas de escala de cinza, conforme ilustra a Figura 3. A região central é demarcada por um cinza mais escuro, já a região dos bairros é identificada por uma escala de cinza intermediária e, por fim, a região da periferia é representada pelo cinza mais claro. Em cada faixa da cidade, existe um requisito mínimo para que a tartaruga possa se estabelecer: o salário. No centro, apenas agentes com salário igual ou superior a cinco mil reais podem se instalar. Nos bairros, apenas agentes com salário igual ou superior a mil reais podem permanecer. Já na periferia, independentemente do salário, qualquer tartaruga pode permanecer nessa região.

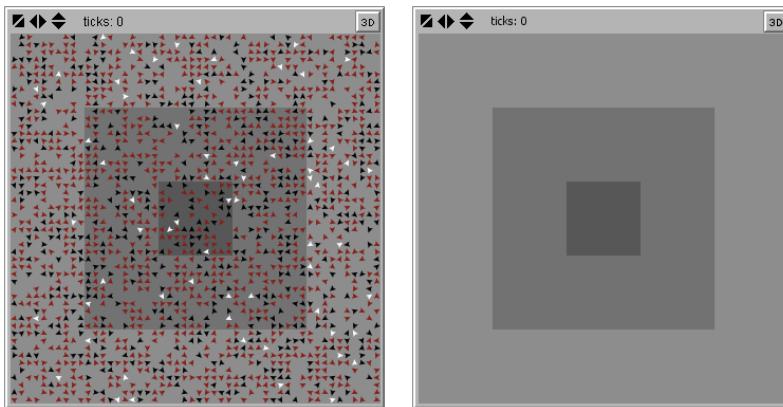


Figura 3. Representação das classes sociais (cores dos agentes) e do espaço urbano (escalas de cinza)

Os agentes possuem uma restrição própria em permanecer próximas a um número mínimo de seus semelhantes, ou seja, os agentes amarelos desejam conviver uma porcentagem “x” de agentes amarelos ao seu redor, e assim por diante. A simulação mostra como estas preferências individuais e as restrições de ambiente podem levar à emergência de um fenômeno global em um sistema social.

Inicialmente, os agentes brancos, pretos e vermelhos são aleatoriamente distribuídos no ambiente, onde a maioria dos agentes está infeliz, pois não possui vizinhos semelhantes (mesma classe social) suficientes e/ou não possuem renda suficiente para manter-se em seu lugar atual. Diante disso, os agentes infelizes pulam para novas regiões da vizinhança, podendo alterar o equilíbrio do local, fazendo com que outros agentes abandonem a região também.

Neste exemplo, o comportamento dos agentes é regido por duas regras principais. A primeira regra estabelece que cada região do ambiente só pode ser ocupada por agentes que possuem renda suficiente para manter-se no local. Por sua vez, a segunda regra especifica que os agentes devem conviver apenas com seus semelhantes, de acordo com uma porcentagem de aceitação mínima determinada na interface do modelo.



Figura 4. Evolução na felicidade e similaridade da população em três estágios da simulação

Ao longo do processo de simulação, o número de agentes infelizes diminui, mas o ambiente torna-se mais segregado, conforme apresenta a Figura 5. Ao final do processo, todas os agentes vermelhos encontram-se na periferia, pois não possuem renda suficiente para manter-se em outras regiões. Os agentes pretos, por sua vez, aglomeraram-se em um grande grupo na região dos bairros e em suas “fronteiras” com a periferia, podendo até mesmo formar pequenos grupos na região da periferia. E por fim, os agentes ricos são

divididos em pequenos grupos entre as três regiões e, em alguns casos, acabam ficando isolados. Assim, ao término da simulação, pode-se analisar que o percentual de infelicidade dos agentes em todas as classes sociais se tornou nulo, ou seja, todos os agentes ficaram felizes.

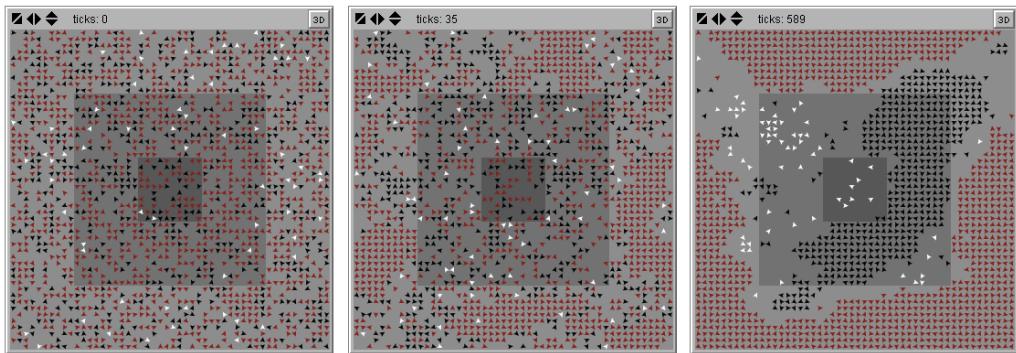


Figura 5. Emergência do fenômeno global da segregação social ao longo da simulação

3.2. Exemplo 2

Na simulação apresentada neste exemplo, são adotados apenas dois itens que caracterizam os indivíduos: a renda de cada agente que é definida por sua classe social (A, B ou C), e o custo de vida em três áreas do espaço de simulação para permanência dos agentes nas mesmas.

Para ter um ciclo de simulação terminado todos os agentes devem estar na condição de “feliz” e para que isso ocorra são trabalhados os seguintes critérios: a classe social do agente e se ele possui alguma similaridade com outro agente vizinho.

A segregação ocorre por similaridade, porém um agente da Classe_C (agente azul) não teria condições econômicas de manter-se por muito tempo na área central do cenário devido ao alto custo de vida. A Classe_A (agente vermelho) mesmo tendo condições econômicas de se manter em qualquer área do cenário não viveria em um determinado lugar que por ter o menor custo de vida consequentemente possui os piores serviços à disposição. Os agentes da Classe_B (agente amarelo) possuem a melhor condição dentro do cenário de simulação, pois possuem poder econômico suficiente para se manter em qualquer um dos três lugares simulados.

Na Rodada 1 Máxima (todos os tipos de agentes com o número máximo no ambiente) foi verificado que em todos os tamanhos de caminhada houve no mínimo um caso em que o número de *ticks* ultrapassou 5000, pois o que emergiu da simulação é a resultante da falta de espaço para o agente da classe que não habitaria uma área muito distante do centro da cidade e por não encontrar neste local nenhum vizinho similar a ele. A Figura 6 mostra essa situação.

No caso supracitado, a finalização da simulação ocorre da seguinte forma a Área central fica ocupada pelas classes A e B, a Área intermediária fica ocupada pelas classes A, B e C, e a Área mais afastada fica ocupada pelas classes B e C sendo que agentes da Classe_A percorrem todo o cenário e não encontram local que torne a sua condição para

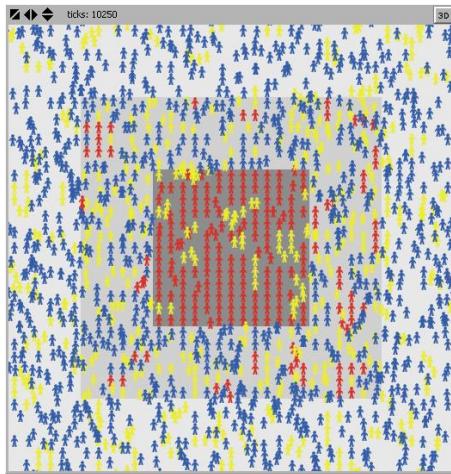


Figura 6. Simulação com o número máximo de agentes

“feliz”, pois o centro já está todo ocupado e para habitar a área intermediária ele deve encontrar algum agente similar.

Na Rodada 2 Média (todos os tipos de agentes com o número médio no ambiente) o fenômeno da falta de espaço não ocorreu, pois há espaço para todos os agentes das classes sociais mudarem para a condição de “feliz” com taxas muito baixas se comparado a Rodada 1 Máxima. Utilizando a idéia da situação que emergiu em simulações que ultrapassaram 5000 ticks, percebe-se um efeito não diferente do que acontece na vida real onde a Classe_A trabalha em uma cidade e mora em outra por diversos fatores, entre eles a falta de estrutura para acomodar e satisfazer a todos, falta de entretenimento, possuir ensino local de qualidade entre outros fatores. A Figura 7 ilustra o caso descrito.

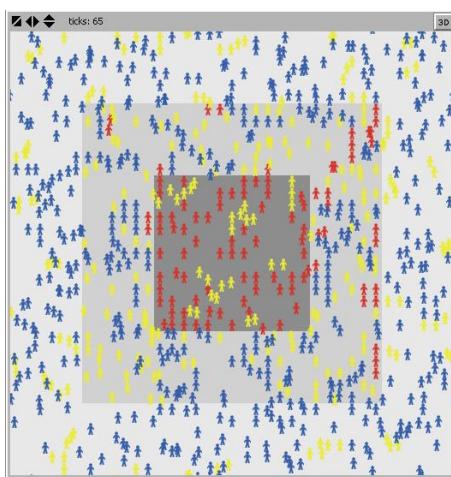


Figura 7. Simulação com o número médio de agentes

3.3. Exemplo 3

Neste exemplo, os indivíduos de cor verde representam a população de classe econômica alta, os de cor amarela simulam a classe média, enquanto o grupo de cor vermelha representa a classe com menor poder aquisitivo da população.

A área central representa uma região nobre do mapa, com custo mais elevado, permitindo que apenas os indivíduos do grupo verde se estabeleçam nessa região. A região periférica do ambiente representa exatamente o oposto, uma área de baixo custo de habitação, permitindo que indivíduos de qualquer um dos grupos propostos assumam uma das posições que compõe a região. Por fim, entre as duas áreas citadas existe uma região com custo intermediário, possibilitando que tanto indivíduos do grupo verde como do grupo amarelo habitem esse espaço.

Para representar a segregação social, utilizou-se o conceito de felicidade, a qual os indivíduos só atingem quando certa porcentagem de seus vizinhos pertence a seu grupo social. Como cada grupo se adapta melhor a uma área diferente do mapa, outra variável do ambiente foi definida, sendo que essa aumenta a porcentagem de vizinhos necessária para que os indivíduos atinjam a felicidade à medida que se afastam de sua região de conforto (grupo verde na área central e amarelo na intermediária). A interface do modelo utilizado no exemplo pode ser visualizada na Figura 8.

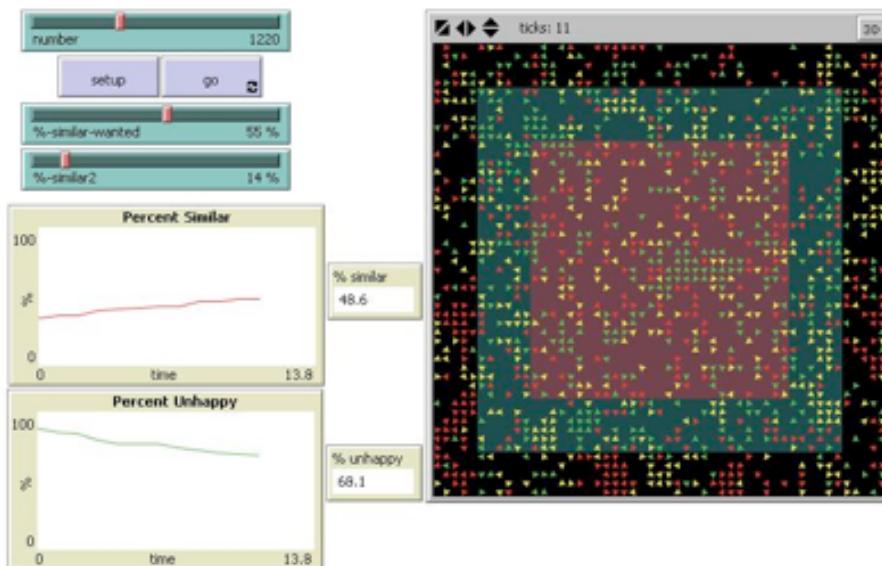


Figura 8. Interface do modelo com a simulação em andamento.

O ambiente onde os agentes interagem possui 2601 (51x51) *patches*, tendo a região rosa 841 *patches*, a verde 840 *patches* e a preta com 920 *patches*. Como a área periférica pode ser ocupada por qualquer um dos grupos de agentes definidos, ela foi definida um pouco maior do que as outras.

O usuário do modelo pode interagir com a simulação alterando o valor de três variáveis. A primeira é “*number*” que determina o número de agentes que compõem a simulação, podendo ser atribuído um valor de até 2500 indivíduos divididos igualmente entre os grupos. A segunda variável que pode ser alterada, “*%-similar-wanted*”, é porcentagem de vizinhos da mesma classe que um indivíduo deseja para que ele permaneça em um *patch*, ou seja, atinja a felicidade. Relacionada a ela, a terceira variável controlada pelo usuário, “*%-similar2*”, é um aumento na porcentagem de vizinhos semelhantes que um agente necessita para permanecer em um *patch* que não seja a condizente com sua classe, mas que seja possível para ele, como por exemplo, os agentes amarelos na

periferia, ou os verdes na área central e na periferia.

A influência dos valores atribuídos para as variáveis no resultado final da simulação pode ser visualizada nos exemplos apresentados na Figura 9 a seguir, onde ambas as simulações foram realizadas utilizando-se os mesmos valores para “*number*” e “%*-similar-wanted*”, diferenciando-se apenas o valor determinado para “%*-similar2*”, a qual possui valor 15 para o exemplo à esquerda e 75 para o exemplo à direita.

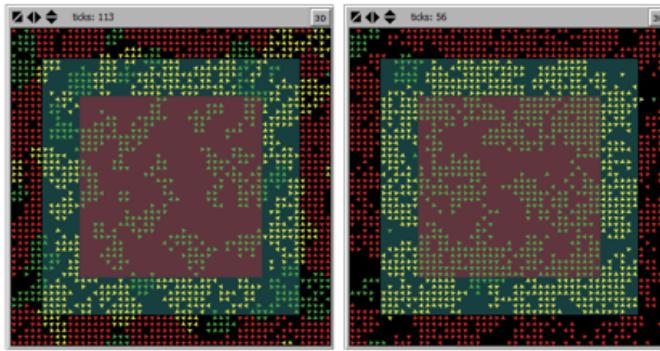


Figura 9. À esquerda, resultado da simulação utilizando o valor 15 para “%*-similar2*”. À direita, resultado com valor 75 para a mesma variável.

Quando se escolhe um valor baixo para “%*-similar2*”, ou seja, determina-se uma baixa influência da região na felicidade do agente, eles tendem a se espalhar pelas diferentes regiões do ambiente, já quando se utiliza um valor alto, os indivíduos tendem a se concentrar em suas regiões preferenciais.

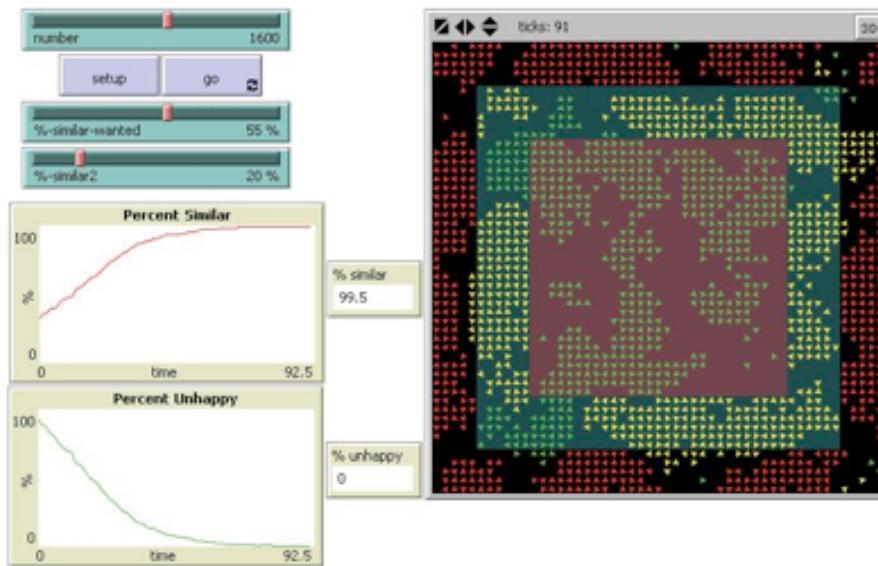
O gráfico “Percent Similar” mostra ao usuário a porcentagem de agentes que está localizado próximo a seus similares, esse valor pode não chegar a 100% no final da simulação, pois alguns agentes podem terminar sem nenhum vizinho. O gráfico “Percent Unhappy” mostra a porcentagem de agentes que ainda não satisfizeram sua condição para que estejam felizes (*happy*), a simulação acaba quando 0% dos agentes estiverem infelizes (Figura 9).

A Figura 10 mostra o cenário ao fim da simulação, onde 100% dos agentes estão felizes com sua localização no espaço. Pode-se observar que 99,5% dos agentes possuem similares na sua vizinhança, os 0,5% de agentes que não possuem estão isolados no espaço.

Outro ponto a ser ressaltado é o de que para cada simulação realizada, independente dos valores definidos para as variáveis, sempre se obtém um resultado diferente, que é gerado a partir da autonomia e da aleatoriedade com que os agentes procuram uma posição em que fiquem felizes no ambiente.

3.4. Exemplo 4

A ideia nesta implementação seria a de simular uma situação na qual a maior parte dos agentes que representam as classes A (renda alta), B (renda média) e C (renda baixa) concentre-se respectivamente nas regiões centrais, intermediárias (entre centro e periferia) e periférica de uma cidade, respectivamente.

**Figura 10.** Fim da simulação.

Inicialmente, determinou-se o seguinte: a cor *Green* representa uma zona “nobre”, a cor *Green + 1* representa a zona “intermediária” e a cor *Green + 1.5* representa uma zona “periférica”. Em seguida, estipulou-se um critério no qual os agentes de cores azul, amarelo e vermelho deveriam permanecer, no final da simulação no ambiente estabelecido, representando as três classes, respectivamente.

Com a finalidade de tornar a simulação mais próxima da realidade, atribuiu-se probabilidades para permanência de cada agente nas três divisões do ambiente, sendo elas: centro, intermediária e periferia. Logo, foram estabelecidos os seguintes critérios:

- Os agentes de cor azul teriam 96% de chance de parar no centro, ou seja, estar feliz na cor *Green*, 2% de parar na zona intermediária correspondente a cor *Green + 1* e 2% de parar na periferia, cor *Green + 1.5*.
- Os agentes de cor amarela teriam 5% de chance de parar no centro, 90% na região intermediária e 5% na periferia.
- Os agentes de cor vermelha teriam 1% de chance de parar no centro, 3% na região intermediária e 96% na periferia.

A Figura 11 mostra uma tabela que resume os critérios estabelecidos.

Agentes	Critério de parada		
	Centro	Intermediário	Periferia
Azul	96%	2%	2%
Amarelo	5%	90%	5%
Vermelho	1%	3%	96%

Figura 11. Tabela resumindo as definições impostas.

A Figura 12 mostra mais detalhes com relação à simulação realizada no *software* NetLogo.

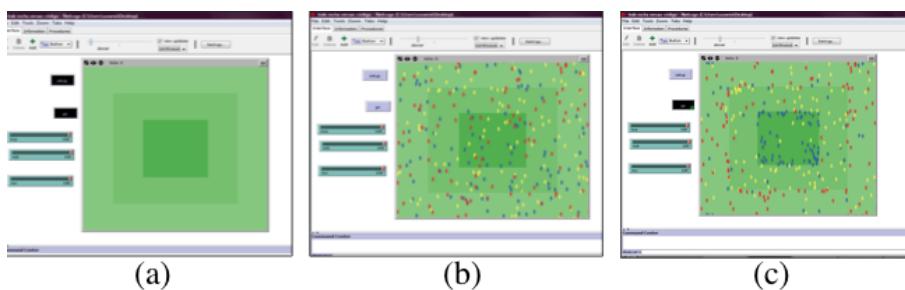


Figura 12. (a) Região correspondente para cada classe: A, B e C. (b) Ambiente com 100 agentes. (c) Final da simulação.

De acordo com as simulações realizadas, variando a quantidade de agentes, conseguiu-se chegar ao objetivo inicial que era o de simular uma situação real envolvendo as classes A, B e C dentro de uma cidade. Foi possível programar de maneira que a maior parte da classe A ficasse concentrada no centro, a maioria dos agentes da classe B, na zona intermediária e a maioria da classe C na periferia. Isso só foi possível por que foi determinado para cada agente da classe A, B e C, respectivamente, probabilidades de parada na região central, intermediária e periférica.

4. Conclusão

Após o desenvolvimento do modelo proposto e da análise dos resultados obtidos, concluiu-se que a utilização do software NetLogo na simulação de sistemas multiagentes baseados no problema da segregação das classes sociais obteve êxito na modelagem de uma sociedade real. A partir de comportamentos individuais baseados em algumas regras, atingiu-se a emergência de um fenômeno global denominado de segregação de classes sociais.

Os resultados obtidos com o modelo ilustram alguns aspectos da realidade de algumas metrópoles, no entanto, este é um tema complexo, composto de diferentes variáveis e elementos que dependem da localidade em que se aplica. As cidades possuem realidades distintas, apresentando comportamentos de segregação dos mais variados possíveis. Além disso, diversas variáveis podem ser levadas em consideração no que diz respeito aos diferentes tipos de segregação que podem ocorrer, tais como sexo, idade, idioma, religião, gostos e hábitos.

Desta forma, necessita-se expandir o trabalho através da adição de novos componentes, visando constituir um modelo mais genérico ou pelo menos, mais abrangente, e que possibilite modelar as diversas realidades presentes em nossas metrópoles, como as inúmeras classes sociais e as áreas impróprias para moradia.

Referências

- Corrêa, R. L. (1993). *O espaço urbano*. Ática, São Paulo, 2th edition.
- Katzman, R. and Ribeiro, L. C. Q. (2008). Metrópoles e sociabilidade: os impactos das transformações socioterritoriais das grandes cidades na coesão social dos países da América Latina. In *Cadernos Metrópole*. São Paulo, n. 20, p. 241-261, 2. sem. 2008.
- Rauch, J. (2002). Seeing around corners. *The Atlantic Monthly*, 289(4):35–48.

- Saboya, R. (2001). Urbanidades - segregação espacial urbana. <http://urbanidades.arq.br/2009/05/segregacao-espacial-urbana/>.
- Schelling, T. (1971). Dynamic models of segregation. In *Journal of Mathematical Sociology*. 1:143-186.
- Villaça, F. (2001). *Espaço intra-urbano no Brasil*. Studio Nobel: FAPESP: Lincoln Institute, São Paulo.
- Wilensky, U. (1997). *NetLogo Segregation model*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Wilensky, U. (2011). Guia do usuário: Netlogo. <http://ccl.northwestern.edu/netlogo/docs>.

Simulando a Execução de Políticas Públicas através de Jason e CArtAgO

Iverton Adão da Silva dos Santos, Antônio Carlos da Rocha Costa

Programa de Pós-Graduação em Modelagem Computacional – Centro de Ciências Computacionais
– Universidade Federal do Rio Grande (FURG)
Av Itália, Km 8 – Campus Carreiros – 96.201-900 – Rio Grande – RS – Brazil

{iverton.santos, ac.rocha.costa}@gmail.com

Abstract. This paper introduces a set of concepts and pre-structured mechanisms (API, Classes and Operations) that contribute to the research on multiagent-based simulation of Public Policies. In particular, it makes use of the Jason and CArtAgO tools, which support high-level programming of cognitive agents and virtual environments, respectively.

Resumo. O presente trabalho apresenta um conjunto conceitos e de mecanismos pré-estruturados (na forma de uma API, Classes e Operações) que contribuem para a pesquisa em simulação Políticas Públicas através de sistemas multiagentes. Em particular, utiliza as ferramentas Jason e CArtAgO, as quais fornecem suporte em alto nível para a programação de agentes cognitivos e ambientes virtuais respectivamente.

1. Introdução

Este trabalho está incluído nos esforços do projeto Modelagem e Simulação de Políticas Públicas (MSPP) [COSTA; DIMURO, 2010], o qual espera como principal resultado poder oferecer um conjunto de conceitos, princípios metodológicos e uma sistemática para a modelagem e simulação, baseadas em agentes, de políticas públicas destinadas a contribuir para a governança de contextos socioeconômicos onde organizações (públicas e/ou privadas) operem como um sistema de institucionais formais.

Nessa perspectiva, e de uma forma mais específica, a contribuição deste trabalho está relacionada aos aspectos instrumentais do processo de simulação, procurando fornecer uma série de mecanismos pré-estruturados (na forma de uma API, Classes e Operações) para programação daqueles planos e normas, de modo que possam ser utilizados sistematicamente para programar e simular políticas públicas utilizando sistemas multiagentes por meio das ferramentas Jason e CArtAgO.

Desta forma, este artigo está estruturado da seguinte maneira. Na próxima seção (Seção 2) discorre-se resumidamente sobre Políticas Públicas, relacionando seus principais componentes e agentes envolvidos. Na seção (Seção 3), são apresentados resumidamente aspectos da plataforma de programação de agentes Jason; na sequência (Seção 4), discorre-se brevemente sobre o *framework* de implementação de ambientes virtuais CArtAgO. Na Seção 5, considerando o objetivo central deste artigo, é desenvolvida uma proposta de mecanismos pré-estruturados para apoio ao desenvolvimento de simulações de políticas públicas. Buscando contextualizar o trabalho, a Seção 6 apresenta os trabalhos com objetivos correlatos. Por fim, são expostas algumas considerações sobre os resultados alcançados e as oportunidades futuras.

2. Políticas Públicas

2.1 Componentes de uma Política Pública

Uma política é um conjunto de princípios que orientam e/ou condicionam decisões e ações dos agentes atuantes no contexto em questão, especialmente no que diz respeito às ações de utilização dos recursos disponíveis nesse contexto [EASTON, 1965]. Por sua vez, políticas públicas são políticas relativas à utilização de recursos públicos ou sociais [HILL, 2009], os princípios determinantes de uma política refletindo os valores que a inspiraram.

Tradicionalmente, o estudo das políticas públicas se faz com base em um conceito de *ciclo de política*, envolvendo as diversas etapas por que passa a criação e operação de uma política [HILL, 2009]:

- Identificação e formulação da questão a ser tratada, no contexto socioeconômico em foco;
- Formulação e análise de soluções alternativas, isto é, das possíveis políticas para o tratamento da questão;
- Escolha de uma solução para ser implementada, isto é, escolha da política a ser implementada;
- Implementação da política;
- Avaliação dos efeitos da política e consequente revisão/reformulação da mesma.

Este trabalho foca a etapa da implementação e, mais propriamente, a subetapa de execução de uma política já implementada ou em fase de implementação.

Para tanto, consideramos que toda política é objetivada e implementada na dupla forma de um *conjunto de normas* a serem aplicadas aos agentes socioeconômicos envolvidos na questão enfocada pela política, assim como um *conjunto de planos de funcionamento* a serem seguidos pelos agentes governamentais que tem potencial de intervenção naquele contexto [COSTA, DIMURO et al., 2010].

Formalmente, dizemos então que uma política é um par $\text{Pol}=(\text{Nrms}, \text{Plns})$, onde Pol é a política, Nrms é o conjunto de normas e Plns é o conjunto de planos.

De acordo com [SILVA; TROTTMAN; et al, 2011] a simulação por agentes, a partir da qual é possível simular o comportamento de agentes por meio de modelos computacionais, permite evidenciar a relação custo/benefício de uma política. Isto porque a partir dessa ferramenta podem ser inseridos dados em programas que simulem o estímulo que deve ser dado aos agentes para que apresentem o comportamento esperado. Em linhas gerais, possibilita averiguar se as indicações e determinações dadas pela política terão, ou não, os reflexos previstos no contexto a que a política se refere.

2.2 Agentes Envolvidos em uma Política Pública

Há, portanto, dois componentes básicos em uma política pública que se põe em execução:

- planos, para serem realizados pelos agentes governamentais envolvidos na execução da política;
- normas (essencialmente, obrigações e proibições), para serem seguidas tanto pelos agentes componentes da sociedade, quanto possivelmente pelos agentes governamentais.

Há, assim, três tipos básicos de agentes envolvidos na execução de uma política pública:

- agentes emissores de políticas, com capacidade de expressar formalmente as políticas para os agentes envolvidos na execução da mesma (neste trabalho, consideramos um único

agente emissor de política, chamado agente *governo*);

- agentes sociais, aos quais a política se destina, com o objetivo de influir sobre seus funcionamentos, no contexto da questão enfocada pela política;

- agentes governamentais, que desempenham funções complementares à execução da política pelos agentes sociais (um tipo particular de agente governamental é o agente *fiscal*, responsável pela fiscalização do seguimento das normas pelos agentes sociais). Outro, é o agente *executor* o qual realiza um plano que pode alterar o ambiente ou promover interações como contratos de agentes sociais.

A definição dos mecanismos pré-estruturados para apoio à simulação de políticas pressupõe, então, que esses três tipos de agentes estejam presentes na simulação, conforme mostrado na Fig. 1.

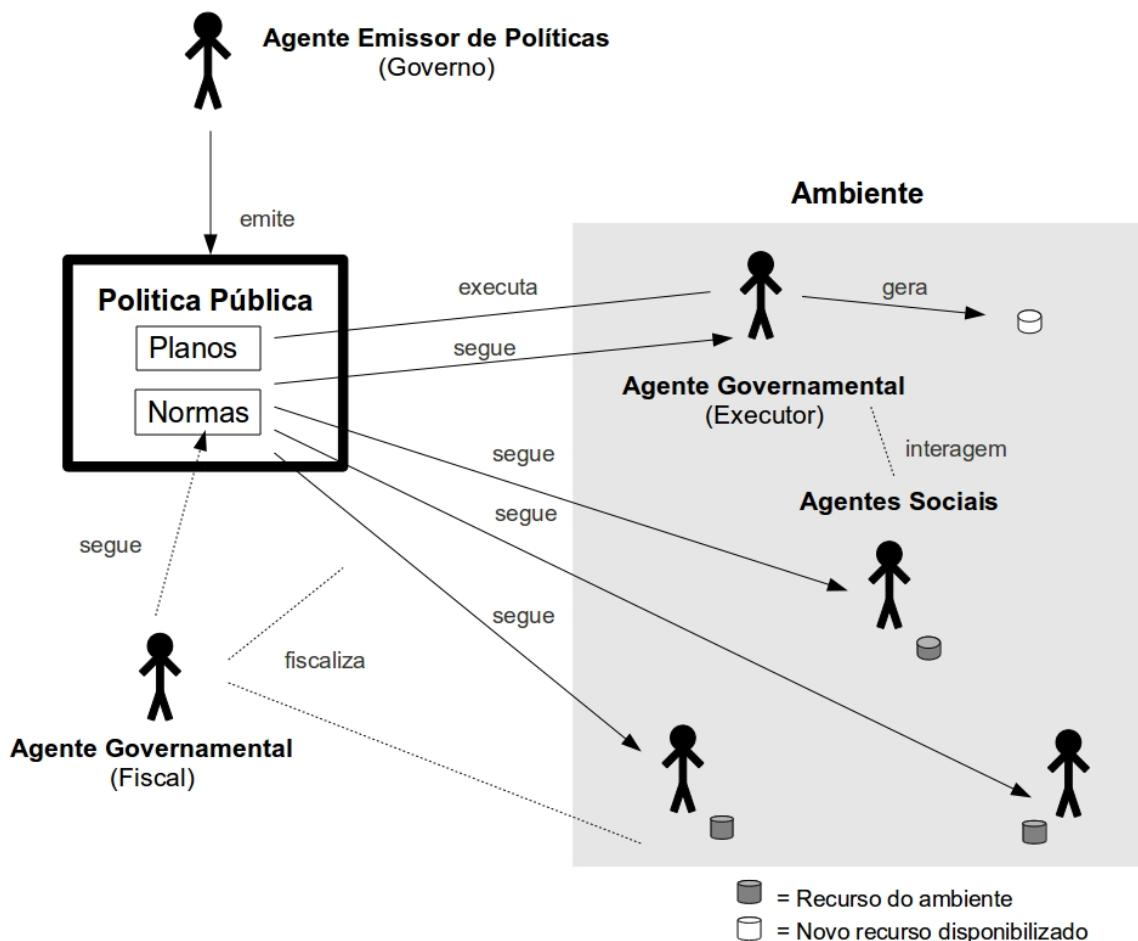


Figura 1: Fluxo dos elementos de uma política pública.

3. Jason

Jason (*A Java-based AgentSpeak Interpreter Used with Saci For Multi-Agent Distribution Over the Net*) [BORDINI; HUBNER; WOOLDRIDGE, 2007] permite a implementação de agentes cognitivos baseados na arquitetura BDI (*Belief, Desires and Intentions*) [RAO; GEORGEFF, 1992], sendo a programação realizada na linguagem AgentSpeak(L) [BORDINI; VIEIRA, 2003].

Esse ambiente inclui comunicação entre agentes baseada na teoria de atos de fala [BORDINI; VIEIRA, 2003].

Outra característica da plataforma é possibilitar que um sistema multiagente possa ser executado de maneira distribuída em uma rede com a utilização do SACI (*Simple Agent Communication Infrastructure*) [BORDINI; HUBNER; et al, 2004] ou da plataforma JADE [KRZYSZTOF; MACIEJ, 2005]. Além disso, Jason é multi-plataforma (característica herdada de sua implementação em Java) e está disponível sob licença GNU GPL.

4. CArtAgO

CArtAgO (*Common ARTifact infrastructure for AGents Open environments*) [RICCI; SANTI; PIUNTI; et al, 2010] é um *framework* que possibilita a implementação de ambientes virtuais para sistemas de agentes. Este *framework* permite a implementação do ambiente como uma camada computacional que encapsula as funcionalidades e os serviços de *artefatos* (objetos não-autônomos) que os agentes podem explorar em tempo de execução.

Conceitualmente, o **CArtAgO** é baseado no meta-modelo *Agents & Artefacts* (A&A) [OMICINI; RICCI; VIROLI, 2008] para modelar sistemas multiagentes. Este modelo introduz uma metáfora de alto nível retirada da ideia de que humanos trabalham de forma cooperativa com o seu ambiente: *agentes* são como entidades computacionais que realizam algum tipo de tarefa orientada para alcançar um objetivo (em analogia aos trabalhadores humanos), e *artefatos* são como recursos e ferramentas dinamicamente construídas, manipuladas e compartilhadas pelos agentes para dar suporte e realizarem suas atividades individuais e coletivas (como artefatos no contexto humano). Assim, programadores de agentes podem desenvolver artefatos os quais são instanciados no ambiente e podem prover serviços para os agentes, podendo inclusive realizar uma comunicação com serviços externos do tipo *web-services*.

Os principais elementos utilizados no CArtAgO são as *propriedades observáveis*, as *operações* e os *sinais*. As propriedades observáveis servem para representar atributos de recursos do ambiente (por exemplo, cor, altura, largura), as operações são implementadas de forma a prover uma interface de utilização de um recurso (por exemplo, sendo um motor um recurso do ambiente, poder-se-ia ter operações ligar, acelerar e desligar providas aos agentes). Por fim, os *sinais* possibilitam programar que os recursos avisem aos agentes sobre uma situação, os quais podem utilizar tais sinais para tomada de decisão (por exemplo, enviar um sinal caso a temperatura ultrapasse um limite).

O **CArtAgO**, realizado em uma tecnologia *Open Source*, está disponível em <http://cartago.sourceforge.net>, e se mostra ao programador do Jason através de uma API baseada na linguagem Java.

5. Mecanismos Pré-estruturados para a Simulação de Políticas Públicas

A proposta de fornecer mecanismos pré-estruturados tem por objetivo facilitar a programação de simulações de políticas públicas nas ferramentas Jason e CArtAgO, no sentido de prover interfaces que visam abstrair e encapsular uma série de elementos que são básicos para aquela finalidade.

Mais especificamente, estes mecanismos se concretizam na forma de uma API com classes, funções, artefatos, operações e planos, os quais são construídos com base nos recursos das linguagens/plataformas AgentSpeak e Java.

Nas próximas seções são apresentados detalhadamente os passos já realizados em relação a esta proposta.

5.1 Norma de Obrigaçāo com manutenção contextual/condicional

Nesta abordagem, teve-se por finalidade viabilizar ao agente emissor de políticas, a criação de um tipo de norma de obrigação, na qual este pode estabelecer que um agente (*social* ou *executor*) tem de verificar de acordo com uma periodicidade se uma condição contextual existe, e caso exista, o mesmo terá obrigação de realizar um objetivo específico. Assim, os parâmetros de criação de uma norma de obrigação são: Id (identificador único), Tipo (neste caso obrigação), Ação, Condição e Periodicidade. Também é possível modificar valores de parâmetros estabelecidos em norma, bem como remover uma norma em vigor.

O parâmetro periodicidade também pode ser utilizado como elemento de um mecanismo de fiscalização. Assim, por exemplo, um agente com papel de fiscal poderia ter um objetivo no qual verificaria dada a passagem de tal período se a obrigação foi cumprida pelos agentes.

Em termos de implementação, a norma se dá por meio do artefato **NormaObrig** com a propriedade observável norma, no seguinte formato `defineObsProperty("norma", Id, Tipo, Acao, Condicao, Period)` e com as operações `criaNorma`, `removeNorma` e `modificaNorma`, conforme mostrado na Fig. 2.

```
public class NormaObrig extends Artifact {

    @OPERATION public void criaNorma(Object Id, Object Tipo, Object Acao,
String Condicao, Object Period){
        defineObsProperty("norma", Id, Tipo, Acao, Condicao, Period);
    }
    @OPERATION public void removeNorma(Object Id, Object Tipo, Object Acao,
Object Condicao, Object Period){
        removeObsPropertyByTemplate("norma", Id, Tipo, Acao, Condicao,
Period);
    }
    @OPERATION void modificaNorma(Object Id, Object Tipo, Object Acao, Object
Condicao, Object Period){
        ObsProperty prop = getObsProperty("norma");
        prop.updateValues(Id, Tipo, Acao, Condicao, Period);
    }
}
```

Figura 2: Artefato NormaObrig.

No exemplo abaixo, estabeleceu-se que numa periodicidade de 1000 unidades de tempo, um agente tem obrigação de pagar imposto de 10%, caso sua renda esteja num intervalo entre os valores 1000,00 e 1500,00:

```
criaNorma(Id, obrigatorio, imposto(10), "renda(R) & R > 1000 &
R < 1500", 1000)[artifact_id(Id)];
```

Desta forma, através da observação de um artefato do tipo **NormaObrig** e do esquema de código da Fig. 3, um agente pode perceber a criação ou modificação de uma norma, e, consequentemente, executar um plano que realizará a manutenção da mesma de acordo com os parâmetros estabelecidos.

```
+norma(Id, Tipo, Acao, Condicao, Freq) [artifact_id(Id)]: true
<- .drop_intention(manutencao(seguir_norma(Id, _, _, _, _)));
!manutencao(seguir_norma(Id, Tipo, Acao, Condicao, Freq)).

+!manutencao(seguir_norma(Id, Tipo, Acao, Condicao, Freq)): true
<- !seguir_norma(Id, Tipo, Acao, Condicao);
.wait(Freq);
!!manutencao(seguir_norma(Id, Tipo, Acao, Condicao, Freq)).
+!seguir_norma(Id, Tipo, Acao, Condicao): condicao(Condicao) &
Tipo == "obrigatorio"
<- .println("seguindo norma: ", Id, Tipo, Acao, Condicao);
.term2string(ACTION,Acao);
!ACTION.

+!seguir_norma(Id, Tipo, Acao, Condicao): not condicao(Condicao)
<- .println("Esta norma não abrange minha condição! ", Condicao).

condicao(Condicao) :- .term2string(COND, Condicao) &
.eval(X, COND) &
X == true.
```

Figura 3: Esquema de plano de manutenção de uma norma de obrigação.

5.2 Normas Proibitivas com penalidades e fiscalização

As normas proibitivas com penalidades foram criadas para possibilitar que o agente emissor de políticas estabeleça ações que não devem ser executados sobre um determinado recurso do ambiente, sendo que a desobediência de um agente (*agente social* ou *executor*) a tal regra implica em uma penalidade (por exemplo, uma multa). A fiscalização, ou seja, aplicação da penalidade é realizada pelo agente que tem por papel cumprir tal tarefa (o *agente fiscal*).

Assim, os parâmetros de criação de uma norma de proibição são Id (identificador único), Tipo (neste caso, proibição), Ação, Penalidade e Parâmetro da Penalidade (no caso de uma multa, o parâmetro seria o valor da mesma). Também é possível modificar valores de parâmetros estabelecidos em norma, bem como remover uma norma de proibição em vigor.

O processo de implementação da norma de proibição, semelhantemente ao caso anterior, se dá por meio de um artefato **NormaProib** com a propriedade observável norma, no entanto segue o seguinte formato `defineObsProperty("norma", Id, Tipo, Acao, Penalidade, ParPenalidade)` e com as operações `criaNorma`, `removeNorma` e `modificaNorma`, conforme Fig. 4, abaixo.

```
public class NormaProib extends Artifact {

    @OPERATION public void criaNorma(Object Id, Object Tipo, Object Acao,      String
Penalidade, Object ParPenalidade) {

        defineObsProperty("norma", Id, Tipo, Acao, Penalidade,ParPenalidade);
    }

    @OPERATION public void removeNorma(Object Id, Object Tipo, Object          Acao,
Object Penalidade, Object ParPenalidade) {
        removeObsPropertyByTemplate("norma", Id, Tipo, Acao, Penalidade, ParPenalidade);
    }
}
```

```

@OPERATION void modificaNorma(Object Id, Object Tipo, Object Acao,          Object
Penalidade, Object ParPenalidade) {
    ObsProperty prop = getObsProperty("norma");
    prop.updateValues(Id, Tipo, Acao, Penalidade, ParPenalidade);
}
}

```

Figura 4: Artefato NormaProib.

No exemplo abaixo, estabeleceu-se que é proibido a ação queimar sobre o recurso horta, sendo que a penalidade a ser aplicada em caso de desobediência será multa, com valor de 10.

```
criaNorma(Id, proibido, queimar, multa, 10) [artifact_id(horta)];
```

Por padronização, qualquer recurso implementado deve gerar uma percepção quando uma ação for invocada, sendo esta percepção num formato `+acao(ACAO, ACTOR, PAR)`. Onde os parâmetros indicam respectivamente a ação executada, quem realizou, e um valor utilizado na execução da ação (por exemplo, se ação fosse plantar teríamos a quantidade plantada). Isto pode ser feito através do recurso de *signals* do CArtAgO, no caso deste exemplo específico, na seguinte forma, `signal("acao", "queimar", getOpUserName(), 10);`

Desta forma, através da observação de um artefato do tipo **NormaProib** e do esquema de código da Fig. 5, um agente (*social* ou *executor*) pode perceber a criação, modificação ou remoção de uma norma de proibição, e, consequentemente, optar por restringir ou não seus comportamentos de acordo com tal norma. De mesma forma, um agente executando o papel de *fiscal* sobre o recurso disponibilizado, pode perceber quais são as proibições que devem ser verificadas afim de aplicar possíveis penalidades.

```

+norma(Id, Tipo, Acao, Penalidade, Ppenalidade) : Tipo == "proibido"
<- .println("Governo criou norma: ", Tipo, ":", Acao);
    .term2string(SAcao, Acao);
    .abolish(ok(SAcao));
    +proibido(SAcao);
    .perceive.

-norma(Id, Tipo, Acao, Penalidade, PPenalidade) : Tipo == "proibido"
<- .println("Governo removeu norma: ", Tipo, ":", Acao);
    .term2string(SAcao, Acao);
    .abolish(proibido(SAcao));
    +ok(SAcao);
    .perceive.

```

Figura 5: Esquema de plano para percepção de normas.

Em linhas gerais, o agente com papel de *fiscal* percebe uma ação executada no recurso e é capaz de inferir se a mesma é proibida e aplicar a penalidade, por meio do seguinte trecho de código da Fig. 6.

```

+acao(ACAO, ACTOR, PAR) : e_proibido(ACAO)
<- .println("Acao: ", ACAO, "ACTOR: ", ACTOR, "PARAMETRO: ", PAR);
?penalidade(ACAO, PENALIDADE, PPENALIDADE);
.println("Penalidade: ", PENALIDADE, PPENALIDADE);
?papel(PAPEL);
.send(ACTOR,tell, penalidade(PENALIDADE, PPENALIDADE));
.println("Penalidade submetida ao agente: ", ACTOR).

e_proibido(ACAO) :- .term2string(TACAO, ACAO) &
    proibido(TACAO).

```

Figura 6: Esquema de plano para percepção de ações.

5.3 Delegação de planos para serem executados por agentes governamentais

Como visto na Seção 2, uma política estabelece obrigações e proibições para os agentes sociais e para os agentes governamentais (por meio de normas), assim como planos que devem ser executados pelos agentes governamentais. O intuito deste segundo mecanismo é, então, apoiar a simulação dos procedimentos pelos quais o agente governo delega planos de ações aos agentes governamentais.

Em termos de implementação, a sintaxe dos planos delegados aos agentes governamentais segue o formato AgentSpeak, ou seja,

```
+!evento : contexto -> corpo.
```

Assim, definiu-se um artefato denominado **Plano** com a propriedade observável `defineObsProperty("plano", Id, PlanEvent, PlanContext, PlanBody, Freq)` e as operações `criaPlano` e `modificaPlano`, conforme Fig. 7, abaixo:

```
public class Plano extends Artifact {

    @OPERATION public void criaPlano(Object Id, String PlanEvent, String
        PlanContext, String PlanBody, int Freq) {
        defineObsProperty("plano", Id, PlanEvent, PlanContext, PlanBody, Freq);
    }

    @OPERATION void modificaPlano(Object Id, Object PlanEvent, Object
        PlanContext, Object PlanBody, int Freq){
        ObsProperty prop = getObsProperty("plano");
        prop.updateValues(Id, PlanEvent, PlanContext, PlanBody, Freq);
    }
}
```

Figura 7: Artefato Plano.

Portanto, os parâmetros das operações representam respectivamente: identificador único do plano, evento de disparo, contexto de aplicação, corpo e uma periodicidade (caso o plano deva ser executado repetidas vezes).

A operação `modificaPlano` permite que o agente com papel de governo modifique o contexto ou o corpo de um plano já delegado, de forma que o agente responsável pela execução do mesmo passa a re-executá-lo de acordo com tais modificações.

Por fim, através da observação de um artefato do tipo Plano e do esquema de código da Fig. 8, um agente poderá perceber a criação ou modificação de um plano, e, posteriormente, executá-lo.

```
+plano(Id, PlanEvent, PlanContext, PlanBody, Freq) [artifact_id(Id)]:
    existe_plano(PlanEvent)
    <- .term2string(PPlanEvent, PlanEvent);
    .drop_intention(executePlan(PPlanEvent,_));
    .concat("+"!, PlanEvent, PlanEvent1);
    .relevant_plans(PlanEvent1, LP, LL);
    .print("Lista de planos encontrados: ", LL);
    .nth(0,LL,PlanPast);
    .remove_plan(PlanPast);
    .concat("+"!, PlanEvent, ":", PlanContext, "<-", PlanBody, PlanFull);
    .add_plan(PlanFull);
    .print("Plano ", PlanEvent, " ATUALIZADO");
    !executePlan(PPlanEvent, Freq).

+plano(Id, PlanEvent, PlanContext, PlanBody, Freq) [artifact_id(Id)]:      not
    existe_plano(PlanEvent)
```

```

<- .concat("!", PlanEvent, " : ", PlanContext, " <- ", PlanBody,      PlanFull);
.add_plan(PlanFull);
.print("PlanFull: ", PlanFull);
.print("Plano ", PlanEvent, " CRIADO");
.term2string(PPlanEvent, PlanEvent);
!executePlan(PPlanEvent, Freq) .

existe_plano(PlanEvent) :- .concat("!", PlanEvent, PlanEvent1) &
.relevant_plans(PlanEvent1, LP) &
.length(LP,X) &
X > 0.

+!executePlan(PPlanEvent, Freq) : true
<- .wait(Freq);
.println("Frequencia aguardada: ", Freq);
!PplanEvent;
!!executePlan(PPlanEvent, Freq) .

-!FALHA: true <- println("Contexto não existente...") .

```

Figura 8: Esquema para percepção de planos.

6. Trabalhos correlatos

É importante destacar que a busca por contribuir no âmbito de políticas públicas é alvo das mais diversas áreas como Administração, Economia, Ciências Sociais, Ciências Políticas etc. Contudo, o interesse aqui está em contribuir provendo recursos computacionais à mesma. Neste sentido, relacionamos alguns trabalhos na área de computação e destacamos como são suas tratativas para abordar tal tema. Nesta perspectiva, será possível contextualizar e comparar a contribuição deste trabalho de uma forma mais clara.

Um dos projetos que abrange este tema é o projeto e-POLICY [e-POLICY, 2012] o qual visa desenvolver um sistema de apoio à decisão para auxiliar os formuladores de políticas em seu processo de decisão. No mesmo, os impactos sociais são derivados a partir de dados recuperados por participação em redes sociais. Nele, tanto o decisor político quanto os cidadãos são auxiliados nas tomadas de decisões e nos processos de participação por meio de ferramentas avançadas de visualização. Além disso, o projeto e-POLICY propõe a avaliação dos impactos econômicos, sociais e ambientais da política tanto a nível global como individual.

O segundo trabalho nesta direção é o projeto denominado OCOPOMO (Open Collaboration for Policy Modeling) [OCOPOMO, 2012] o qual tem como objetivo central demonstrar que, com as técnicas adequadas, a integração da modelagem política formal, da geração de cenários e da colaboração aberta em larga escala não é apenas possível, mas essencial em todos os níveis de formação política seja local, regional, nacional ou global. Assim, busca fornecer aos governos e aos operadores de políticas públicas um ferramental para melhor dominar a complexidade no desenvolvimento de suas estratégicas políticas.

Em ambos projetos, e-POLICY e OCOPOMO, pretende-se usar um sistema multiagente para simular os efeitos das políticas analisadas sobre os sistemas sociais que elas pretendem regular.

Em [BROWN. L.; HARDING. A, 2002] apresenta-se um terceiro trabalho o qual se dá por um estudo sobre a modelagem social e políticas públicas, contudo no que se refere ao potencial e utilidade da técnica de microsimulação aplicada a políticas públicas.

Por fim, uma série de trabalhos focam em construir simuladores computacionais específicos para tratar problemas pontuais, como de políticas públicas para saúde, transito e meio-ambiente. Porém, esta abordagem não é relacionada a prover recursos para suporte a políticas públicas num sentido instrumental geral.

O levantamento bibliográfico realizado mostra, portanto, que há projetos e trabalhos que procuram fornecer uma espécie de suporte computacional para o processo de elaboração de políticas públicas. No entanto, pode-se dizer que embora sendo possível relacionar estes projetos, até o momento não se encontrou na literatura (além do projeto MSPP) esforços direcionados ao desenvolvimento de padrões de projeto para simulações de políticas públicas baseadas em agentes. Igualmente, não se encontraram trabalhos voltados, especificamente, para a implementação desses mecanismos numa direção de padrões de projeto para as ferramentas Jason e Cartago, não sendo possível relacionar trabalhos com semelhanças mais diretas a esta abordagem.

7. Considerações finais

É possível citar dois resultados principais alcançados por este trabalho. O primeiro é o levantamento e definição dos principais elementos de uma política pública, no que se refere a uma perspectiva de simulação de políticas públicas em sistemas multiagentes: normas, planos, agentes sociais, agentes governamentais.

O segundo está no nível de implementação. Neste sentido, os recursos aqui apresentados podem servir como instrumentos iniciais para programação, utilizando Jason e CArtAgO, de um sistema multiagente que envolva aqueles aspectos básicos constituintes em uma política pública, mais especificamente procedimentos (planos) e elementos normativos simples (como obrigações e proibições).

Atualmente, o trabalho encontra-se na fase da definição e implementação de um método para que agentes fiscais verifiquem se as normas de obrigação e os planos estabelecidos estão sendo cumpridos, assim como foi feito para as normas de proibição, que são fiscalizadas e tem penalidade aplicada caso os agentes as descumpriam.

Como continuidade, pretende-se realizar um estudo de como fazer que os mesmos mecanismos propostos possam ser utilizados dentro dos diferentes níveis organizacionais, como grupos de agentes e instituições.

Por fim, espera-se que os resultados obtidos pela concretização da proposta deste trabalho, integrados com os esforços do projeto MSPP, possam prover contribuições metodológicas e ferramentais para a simulação baseada em agentes de políticas públicas.

8. Referências

- BORDINI, R. H.; HUBNER, J. F.; WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak using Jason**. University of Liverpoll: Wiley, 2007.
- BORDINI, R. H.; VIEIRA R. **Linguagens de Programação Orientada a Agentes: uma introdução baseada em AgentSpeak(L)**. Revista de Informática Teórica e Aplicada – UFRGS, , 2003, V 10, N. 1, 32 p.
- BORDINI, R. H.; HUBNER, J. F. et al. **Jason: a java-based agentspeak interpreter used with SACI for multi-agent distribution over the net**. Manual, first release. [S.I.], 2004.
- BROWN. L.; HARDING. A.; **Social Modelling and Public Policy: Application of Microsimulation Modelling in Australia** *Journal of Artificial Societies and Social Simulation* vol. 5, no. 4 Disponível por WWW em <<http://jasss.soc.surrey.ac.uk/5/4/6.html>> , 2002.
- COSTA, A. C. R.; DIMURO, G. P.; et al.; **Modelagem e Simulação de Políticas Públicas** <http://mspp.c3.furg.br/> acesso em 03/2010.

EASTON, D.; **A Framework for Political Analysis**. Prentice-Hall, Englewood Cliffs, 1965.

Engineering the policy making life cycle (ePolicy) Disponível por WWW
<http://cress.soc.surrey.ac.uk/web/projects/59-epolicy>, 2012.

KRZYSZTOF, C.; MACIEJ, G.; PAWEL, K.; MICHAL, S.; MARCIN, P. **Efficiency of JADE Agent Platform**, Scientific Programming, 2005.

HILL, M.: **The Public Policy Process**. Pearson Longman, London, 2009. (5th ed.).

OMICINI, A.; RICCI, A.; VIROLI, M. **Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems**, 17(3):432–456, Dec. 2008.

Open Collaboration for Policy Modeling Disponível por WWW <http://www.ocopomo.eu/>, 2012.

RAO, A. S.; GEORGEFF, M. P. **An Abstract Architecture for Rational Agents**. In: IN-INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING (KR'92), 3., 1992. Proceedings... Morgan Kaufmann, 1992. p.439–449.

RICCI, A.; SANTI, A.; PIUNTI, M et al. **CArtAgO (Common ARTifact infrastructure for AGents Open environments)**. Disponível por WWW em <http://cartago.sourceforge.net/> acesso em 03/2010.

SILVA, V. C.; TROTTMAN, P.; COELHO, F. S.; SARTI, M. F.; **A Abordagem de Sistemas Complexos em Administração (Pública): conceitos, agenda de pesquisa e uma aplicação na subárea de Administração/Políticas Pública(s)**. XIV SemeAd Seminários em Administração, 2011, ISSN 2177-3866

Modelando a Organização Social de um SMA para Simulação dos Processos de Produção e Gestão Social de um Ecossistema Urbano: o caso da Horta San Jerónimo da cidade de Sevilla, Espanha

Flávia C. P. Santos¹, Glenda Dimuro², Thiago F. Rodrigues¹, Diana F. Adamati¹, Graçaliz P. Dimuro¹, Antônio C. R. Costa¹, Esteban de Manuel Jerez²

¹PPGMC, PPGComp, C3, Universidade Federal do Rio Grande, Rio Grande, Brasil

²Depto de Expresión Gráfica Arquitectónica, Universidad de Sevilla, Sevilla, Espanha

{faflasan, trodrigues02, dianaada, gracaliz, glenda.dimuro}@gmail.com

Abstract. *The aim of this work is to describe the application of a methodology for the development of MAS organization in the context of the simulation of urban ecosystems. We present the modeling of the organization of a MAS for the simulation of the social production process and management of the San Jerónimo Urban Vegetable Garden (Seville, Spain). The conceptual modeling of the social interactions, norms and the routines of the roles in the vegetable garden organization was done with conceptual maps and ellipses. The modeling of the organization was built using the organizational model MOISE+, specifying the structural and functional dimensions. The limitations of the chosen organizational model in the context of social organizations is also discussed.*

Resumo. *O objetivo deste trabalho é descrever a aplicação de uma metodologia para o desenvolvimento de organização de SMA no contexto da simulação de ecossistemas urbanos. Apresenta-se a modelagem da organização de um SMA para a simulação dos processos de produção e gestão social da Horta Urbana de San Jerónimo, localizada em Sevilla, Espanha. A modelagem das interações sociais, normas constitutivas e regulativas, e as rotinas dos papéis organizacionais da horta foi realizada com mapas conceituais e elipses. A modelagem da organização foi construída com o modelo organizacional MOISE+, especificado nas dimensões estrutural e funcional. Discute-se também as limitações do modelo escolhido no contexto de organizações sociais.*

1. Introdução

Para afrontar os distintos problemas causados pela sociedade industrial, é preciso a utilização de paradigmas capazes de tratar os novos agentes sociais e os novos conflitos surgidos da flexibilidade gerada pela civilização industrial da era de informação, incluindo, obviamente, temas que abarquem as questões ecológicas e econômicas desta sociedade [Touraine 2005]: uma reforma no raciocínio científico que produza um “pensamento do contexto e do complexo” [Morin 2010], ou seja, que una o que era antes compartmentado, que respeite o diverso ao mesmo tempo em que reconhece a unidade, um pensamento que não isole, mas que considere o objeto de estudo por sua relação com o entorno social, econômico, político, ambiental, i.e., que aceite a incerteza de suas ações.

Ao analisar este pensamento complexo e partir de um ponto de vista mais ecológico e sistêmico da vida, surge uma alternativa capaz de diminuir a degradação social, ambiental e econômica em que vivemos, que está relacionada com o conceito de sustentabilidade. Embora muitos autores estimem que o sucesso desta “nova” terminologia seja, em boa parte, devido à própria ambiguidade conceitual que lhe acompanha [Naredo 1996], se pode orientar sua aplicação desde um enfoque complexo. Há muitas outras pretensões além das sugeridas pelo Relatório de Brundtland¹, e, sendo assim, se passa a considerar que a sustentabilidade não se refere apenas ao tipo de interação humana com o mundo que preserva seu meio ambiente para não comprometer os recursos naturais das futuras gerações. A aplicação do conceito sob o enfoque da complexidade inclui as reflexões da ecologia social e urbana, que consideram, respectivamente, a relação holística entre os seres humanos e o meio ambiente – em especial como a ação humana costuma incidir destrutivamente sobre a natureza [Vieria and Bredariol 1998] – e a utilização da ecologia científica, simulação social e ambiental, inteligência artificial, sistemas multiagentes (SMA), etc., para entender e interpretar a realidade urbana [Bettini 1998, Gilbert and Troitzsch 1999].

Para reparar danos ambientais é necessário solucionar questões sociais e econômicas, que implicam mudanças de mentalidades e comportamentos, ampliando a participação e implicação de cidadãos na defesa do seu entorno. É neste ponto que se faz a conexão entre a ecologia urbana e a produção e gestão social do habitat [Lobo 1998, Ortiz 2010, Pelli 2010, Romero et al. 2004]. Transpor a sustentabilidade da teoria à prática significa conceber o ser humano e o território onde a maioria da espécie se desenvolve – as cidades – como parte da natureza, sob o conceito de “ecossistemas urbanos” [Terradas 2001, Dimuro and Jerez 2011]. Um ecossistema urbano não é uma simples agregação de espaços aleatórios, mas um todo conectado com redes dentro de redes com causas e efeitos; um habitat com uma estrutura coerente com os paradigmas culturais e necessidades específicas de um determinado grupo e contexto; um processo de incremento incessante de informações; um território fisicamente fechado, mas aberto a fluxos de energia e recursos.

O conceito de produção e gestão social de ecossistemas urbanos pode ser compreendido como a geração de novas situações, físicas ou relacionais, mediante a construção, transformação ou eliminação de objetos físicos e/ou de objetos relacionais com o objetivo de assegurar, nas novas situações produzidas, o cumprimento de suas funções sociais e ambientais [Ortiz 2010, Pelli 2007, Pelli 2010]. Isto inclui a participação cidadã nos processos de planejamento e transformação urbana, articulando distintos agentes envolvidos (governo, instituições, técnicos, cidadãos), formando uma rede estruturada e apoiada em mecanismos e ferramentas que possibilitem a distribuição igualitária de poder na tomada de decisões, de modo que todos os agentes possam participar e dialogar ativamente em todo o processo de um determinado projeto, desde a sua planificação até sua gestão. A produção e a gestão social de ecossistemas urbanos contribuem ao fortalecimento de práticas comunitárias, ao aumento da responsabilidade por um projeto coletivo, ao exercício da democracia, ao desenvolvimento de ações mais solidárias, incluindo tanto temas produtivos e econômicos, como ambientais.

¹Documento denominado “Our Common Future” (1987), que define Desenvolvimento Sustentável como o que satisfaz as necessidades presentes, sem comprometer a capacidade das gerações futuras de suprir suas próprias necessidades.

Este trabalho aborda de forma interdisciplinar a produção e gestão social de um ecossistema urbano – um esforço conjunto de interrelacionar saberes buscando interpretações coletivas, utilizando como caso de estudo para a simulação multiagente a atual tendência de (re)aproximar o campo à cidade através de hortas urbanas. A organização escolhida é o projeto de hortas sociais realizado no Parque de San Jerónimo (Sevilha/Espanha), impulsionado pela ONG Ecologistas em Acción.

O artigo apresenta a primeira fase da modelagem da organização de um SMA, construída utilizando o modelo organizacional MOISE+ [Hübner 2003], identificando as interações sociais, normas constitutivas e regulativas, e as rotinas dos papéis organizacionais que compõem a Horta San Jerónimo (HSJ), com a ajuda de mapas conceituais e elipses. Este estudo permite analisar de forma complexa os comportamentos e rotinas dos papéis assumidos pelos agentes no sistema, e como interagem de acordo com o contexto, ou seja, como o ecossistema é produzido e dirigido. A aplicação desse estudo deverá, no futuro, contribuir não apenas para a visualização da realidade atual, mas também para verificar como possíveis alterações em ações, comportamentos e papéis dos agentes envolvidos, principalmente desde o ponto de vista da sua participação nos processos de tomada de decisões, podem transformar esta realidade, desde o ponto de vista social, ambiental e econômico, e contribuir para a sustentabilidade do projeto.

O artigo está organizado da seguinte forma. Na Seção 2, identificam-se os papéis que compõem a organização da HSJ e suas rotinas, com base em normas constitutivas e regulativas, construídas a partir do regulamento da HSJ. Na Seção 3, apresenta-se parte da modelagem realizada utilizando o modelo organizacional Moise+, nas dimensões estrutural e funcional. A Seção 4 discute as limitações do modelo para utilização na modelagem de organização sociais. A Seção 5 é a Conclusão.

2. Identificação dos papéis, normas, e rotinas dos papéis

A HSJ é uma iniciativa da ONG Ecologistas em Acción com o objetivo de fomentar a participação social em práticas de agricultura orgânica, mediante o uso e desfrute de hortas de *lazer*, e realização de atividades vinculadas com a educação ambiental. Ocupa cerca de 1,5 hectares do Parque Municipal de San Jerónimo e está divida em parcelas individuais cultiváveis (ao redor de 42 unidades com dimensões que variam de 75 a 150m²) designadas a hortelãos por um prazo de dois anos prorrogáveis – sempre que cumpram com as normas e regras estabelecidas no regulamento definido pela ONG. Suas principais características são o fato de ser uma horta social sem fins lucrativos, ou seja, a produção é dedicada para o autoconsumo e ser apoiada economicamente por financiamento municipal e colaboração dos participantes.

Do ponto de vista organizacional, a ONG presta assessoria técnica aos hortelãos, divididos em *hortelão titular* (responsável pela parcela, neste artigo referido simplesmente por *hortelão*), *hortelão auxiliar* (hortelão que trabalha em uma parcela, normalmente um familiar, mas não é o responsável) e *aspirante a hortelão* (cidadãos que fazem parte da lista de espera). As interações internas da HSJ podem ser observadas no mapa conceitual mostrado na Fig. 1, discutido com mais detalhe em [Dimuro et al. 2011], onde também são apresentadas as interações externas ao projeto da HSJ. Com base no regulamento da ONG Ecologistas em Acción, foi elaborada a descrição formal das normas que regem o sistema social da HSJ, que foram classificadas em constitutivas e regulativas. A Fig. 2

mostra exemplos destas regras, discutidas em maior detalhe em [Santos et al. 2012]. Por exemplo, durante o uso da horta, o agente hortelão precisa permissão junto a organização para construir um depósito para guardar utensílios. Esta requisição é formalizada por um requerimento e como resultado dessa ação, tem-se ou não a construção do depósito. A fiscalização desta norma (constitutiva) é realizada pela organização da horta.

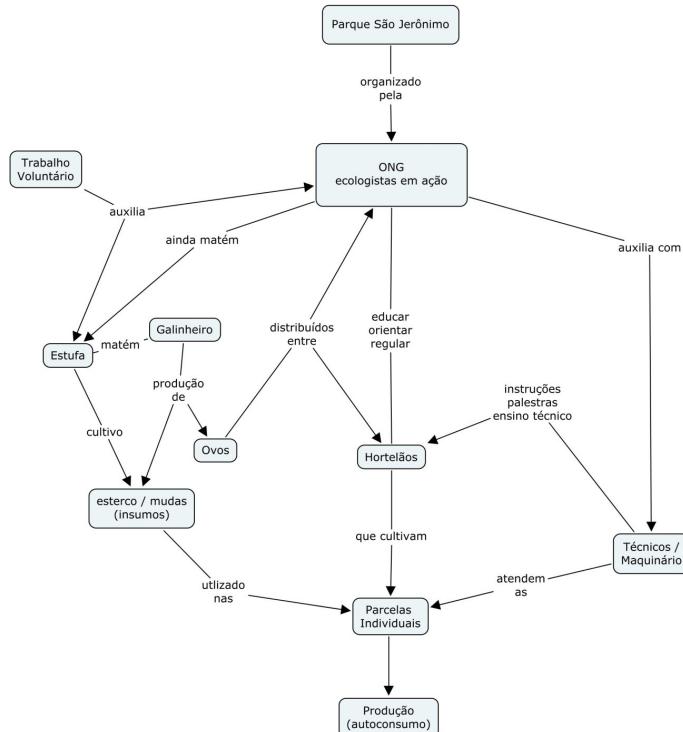


Figura 1. Mapa conceitual do projeto da HSJ

Tipo da Norma	Situação em que se aplica	Ação Normatizada					Verificador da ação	Sanções (Punições, Recompensas)
		Pré-condição	Normatização	Id da ação	Resultados	Papel realizador		
Constitutiva	Durante o uso da horta	Requerimento de auxílio junto à organização	Permissão	Construção de um depósito para guardar utensílios	Constrói o depósito; Não constrói o depósito	Hortelão	Organização da horta	-
Regulativa	Durante o uso da horta, mensalmente	Possuir uma parcela	Obrigação	Pagar mensalidade	Ganha auxílio de cultivo E Continua na horta; Não ganha auxílio de cultivo E Sai da horta (depende do número de faltas graves)	Hortelão	Organização da horta	Punição: Falta grave (cumulativa)
Constitutiva	Durante o uso da horta	Informar à organização da horta	Permissão	Designar um ajudante	Ajudante designado O UX Negação do pedido	Hortelão	Organização da horta	-
Regulativa	Durante o uso da horta	Possuir uma parcela	Proibição	Modificação do desenho da horta	Continuar na horta; Sair da horta (depende do número de faltas graves)	Hortelão	Organização da horta	Punição: Falta grave (cumulativa)

Figura 2. Parte da tabela de normas da HSJ

Na sequência, identificam-se os papéis organizacionais que compõem a HSJ: um papel Abstrato (\mathcal{P}_{soc}) que é a raiz da árvore de papéis, Hortelão ($\mathcal{P}_{hortelao}$), Aspirante a Hortelão ($\mathcal{P}_{aspirante\ a\ hortelao}$), Hortelão Auxiliar ($\mathcal{P}_{hortelao\ auxiliar}$), Técnicos ($\mathcal{P}_{tecnico}$), Secretaria ($\mathcal{P}_{secretaria}$) e Administração ($\mathcal{P}_{administracao}$) da ONG. Para organizar e estabelecer os comportamentos e rotinas dos diferentes papéis, assim como a periodicidade dessas rotinas, foram utilizadas elipses, no sentido dos diagramas de Venn. A utilização

de elipses auxilia na visualização das rotinas de cada papel, o que facilita o entendimento do comportamento dos agentes no sistema, bem como a identificação das interações entre eles e com o ambiente. Como exemplo, a Fig. 3 mostra a modelagem em elipses das rotinas da Secretaria da ONG, descritas como:

Rotinas Diárias: receber documentação de candidatos a participar no projeto, denominados de Aspirante a Hortelão; realizar inscrição do Aspirante a Hortelão; receber requisição de transferência de posse de parcela.

Rotinas Mensais: receber pagamento de mensalidade, paga pelo hortelão para cobrir custos com água (goteamento), material para controle de pragas, uso de ferramentas comuns entre hortelões, etc.; informar datas de assembléias (em mural); cadastrar vinculação de Hortelão Auxiliar, informado pelo hortelão titular.

Rotina BIANUAL: receber pedido de permanência do hortelão no projeto.

Rotina Sazonal: encaminhar a documentação recebida para Administração da ONG.

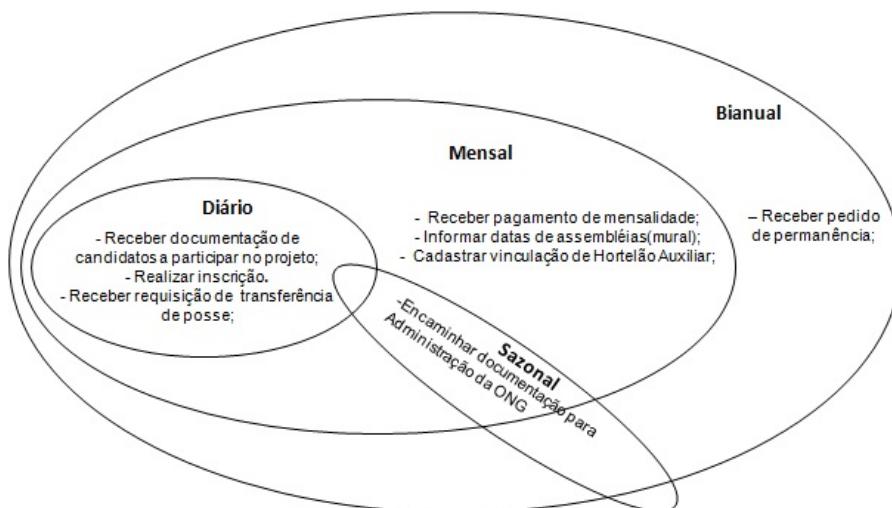


Figura 3. Elipses das rotinas da secretaria ONG

3. Modelagem da HSJ no MOISE⁺

O modelo organizacional MOISE⁺ [Hübner et al. 2002] foi desenvolvido para modelar a organização de SMA e consiste na especificação de três dimensões: a estrutural, onde definem-se papéis e ligações de heranças e grupos; a funcional, onde é estabelecido um conjunto de planos globais e missões para que as metas sejam atingidas; e a deôntica, que é a dimensão responsável pela definição de qual papel tem obrigação ou permissão para realizar cada missão. Neste artigo, por restrições de espaço, somente as duas primeiras dimensões são apresentadas para o sistema social da HSJ.

3.1. Especificação Estrutural - EE

Em uma Especificação Estrutural (EE), os níveis individual, social e coletivo podem ser definidos a partir de três conceitos: papéis (conjunto de restrições comportamentais que um agente aceita ao entrar em um grupo), relações entre papéis (nível social - relações impostas às interações entre os papéis) e grupos (nível coletivo - representa um conjunto de agentes com afinidades maiores e objetivos mais próximos).

Para representar o caso da HSJ no modelo organizacional MOISE⁺ na definição de organização hierarquizada, uma definição de grupo, sub-grupo e seus respectivos papéis são apresentados na Fig. 4. Assim, a EE pode ser representada por uma tupla:

$$ss = (\mathcal{RG}, \mathcal{R}_{ss}, \sqsubset) \quad (1)$$

onde \mathcal{RG} é o conjunto de especificação de grupos raízes de ss , \mathcal{R}_{ss} é o conjunto de todos os papéis da EE, \sqsubset é a relação de herança sobre os papéis de \mathcal{R}_{ss} .

Na Fig. 4, demonstra-se o exemplo do modelo estrutural para a organização da HSJ, onde são especificados papéis, grupo e sub-grupos, relações entre papéis e grupos. Nesta EE, tem-se especificado um grupo HSJ como grupo raiz e seus sub-grupos PARCELA e ONG. Os papéis que podem ser assumidos nestes sub-grupos são, respectivamente: hortelão e hortelão auxiliar, administração, secretaria e técnico da ONG. As relações entre estes papéis podem ser de autoridade (que é o caso da administração da ONG com relação à secretaria, técnico e hortelão), comunicação e compatibilidade (entre hortelão auxiliar e aspirante a hortelão), etc.

Assim, a EE é dada como:

$${}_{ss}\mathcal{HSJ} = \langle \{gt\}_{\mathcal{HSJ}}, \mathcal{R}_{\mathcal{HSJ}}, \sqsubset \rangle \quad (2)$$

onde o conjunto de papéis é

$$\begin{aligned} (\mathcal{R}_{\mathcal{HSJ}} = & \{\mathcal{P}_{soc}, \mathcal{P}_{hortelao}, \mathcal{P}_{administracao}, \mathcal{P}_{hortelao\ auxiliar}, \\ & \mathcal{P}_{aspirante\ a\ hortelao}, \mathcal{P}_{tecnico}, \mathcal{P}_{secretaria}\}); \end{aligned}$$

a relação de herança sobre os papéis é dada como:

$$\begin{aligned} & (\mathcal{P}_{soc} \sqsubset \mathcal{P}_{hortelao}, \mathcal{P}_{soc} \sqsubset \mathcal{P}_{administracao} \wedge \\ & \mathcal{P}_{hortelao} \sqsubset \mathcal{P}_{hortelao\ auxiliar}, \mathcal{P}_{administracao} \sqsubset \mathcal{P}_{tecnico}, \mathcal{P}_{administracao} \sqsubset \mathcal{P}_{secretaria}). \end{aligned}$$

A especificação de grupo e sub-grupo da Fig. 4 foi formalizada da seguinte forma:

$$gt = \langle \mathcal{R}, \mathcal{SG}, \mathcal{L}^{intra}, \mathcal{L}^{inter}, \mathcal{C}^{intra}, \mathcal{C}^{inter}, np, ng \rangle \quad (3)$$

onde $\mathcal{R} \subseteq \mathcal{R}_{ss}$ é o conjunto de papéis que podem ser assumidos em grupos criados a partir da especificação de grupo gt ; $\mathcal{SG} \subseteq \mathcal{G}_T$ é o conjunto de sub-grupos possíveis no grupo gt ; \mathcal{L}^{intra} e \mathcal{L}^{inter} são os conjuntos de ligações internas e externas ao grupo gt ; \mathcal{C}^{intra} e \mathcal{C}^{inter} são os conjuntos de compatibilidades internas ou externas ao grupo gt ; $np: \mathcal{R}_{ss} \rightarrow \mathbb{N} \times \mathbb{N}$ é um mapeamento parcial para cardinalidade de papéis (indica um valor mínimo e máximo para cada papel); $ng: \mathcal{SG} \rightarrow \mathbb{N} \times \mathbb{N}$ é um mapeamento parcial para cardinalidade de sub-grupos (indica um valor mínimo e máximo para cada sub-grupo).

No nível coletivo, as ligações passam a possuir um escopo: internas ou externas ao grupo. Na Fig. 4, um exemplo de ligação interna de *autoridade* é o caso em que um agente com papel hortelão tem autoridade sobre hortelão auxiliar do seu grupo (Parcela), ou seja, grupo ao qual os dois pertencem, o que denota-se por $link(\mathcal{P}_{hortelao}, \mathcal{P}_{hortelao\ auxiliar}, aut) \in \mathcal{L}^{intra}$. Se $link(\mathcal{P}_{hortelao}, \mathcal{P}_{hortelao\ auxiliar}, com) \in \mathcal{L}^{inter}$, então qualquer agente com papel hortelão pode se comunicar com outros agentes

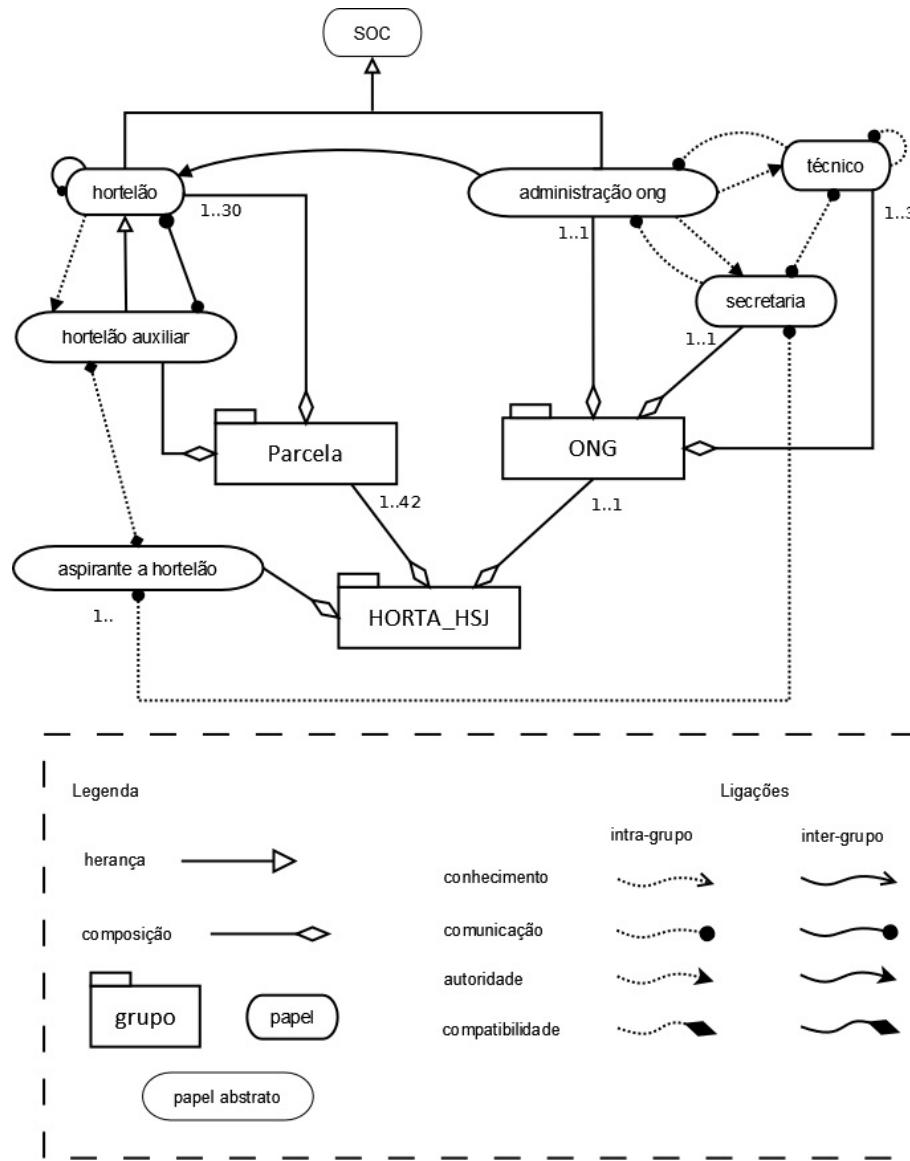


Figura 4. Especificação Estrutural da Horta San Jerónimo com modelo MOISE⁺

com papel hortelão auxiliar no contexto do grupo raiz \mathcal{HSJ} no qual todos os papéis estão incluídos, mostrando um exemplo de ligação externa de *comunicação*.

As compatibilidades também possuem um escopo quando incluídas em um grupo e são válidas para os agentes do grupo. Na Fig. 4, um exemplo de compatibilidade interna é o caso em que um agente com papel $P_{hortelao\ auxiliar}$ em um grupo pode assumir o papel $P_{aspirante\ a\ hortelao}$ no mesmo grupo, o que denota-se por $P_{hortelao\ auxiliar} \bowtie P_{aspirante\ a\ hortelao} \in \mathcal{C}^{intra}$. E, para especificar que um agente pode assumir papéis em grupos diferentes, adiciona-se \mathcal{C}^{inter} , ou seja, compatibilidade externa ao seu grupo.

Através das funções np e ng em uma especificação de grupo estabelecem-se as cardinalidades para os papéis e sub-grupos, definindo limites mínimos e máximos para estes elementos do grupo. Estabelece-se, desta forma, o que determina a *boa formação* do grupo. Por exemplo, na especificação de grupo \mathcal{HSJ} , pode-se dizer que $np(P_{hortelao}) = (1, 30)$ e $ng(gt\ PARCELA) = (1, 42)$, portanto o grupo \mathcal{HSJ} é consi-

derado bem formado se de um a trinta agentes assumem o papel de hortelão e se houver de uma a quarenta e duas parcelas no grupo. Caso contrário, o grupo não é bem formado.

Neste trabalho a EE descrita na Fig. 4 demonstra a criação de três especificações de grupo, onde \mathcal{HSJ} é o grupo raiz, PARCELA e ONG são sub-grupos. A especificação destes grupos é mostrada na Fig. 5.

Na formalização da EE, observa-se que a especificação do grupo $gt \mathcal{HSJ}$ possui um único papel que pode ser assumido por algum agente, que é o de *aspirante a hortelão*, pois este não faz parte ainda do grupo PARCELA, ou seja, ele é um candidato a hortelão. No sub-grupo ONG, os papéis de *secretaria*, *técnico* e *administração*, assim como no sub-grupo PARCELA, os papéis *hortelão* e *hortelão auxiliar*, podem ser assumidos por algum agente, e suas ligações podem ocorrer através de links que correspondem ao tipo de ligação que cada papel apresenta. Por exemplo, qualquer hortelão pode se comunicar com qualquer hortelão auxiliar, independentemente de grupo.

Há ainda a relação de compatibilidade no grupo HSJ, onde um hortelão auxiliar pode assumir o papel de aspirante a hortelão, pois este pode, ao mesmo tempo em que exerce a função de hortelão auxiliar em uma parcela, estar inscrito em uma lista de espera como candidato a ter sua própria parcela e este papel é chamado de aspirante a hortelão. Uma instância do grupo HSJ é bem formada se possuir uma instância do sub-grupo ONG e no mínimo uma e no máximo quarenta e duas parcelas do sub-grupo ($gt \mathcal{PARCELA}) \rightarrow (1,42)$.

3.2. Especificação Funcional - EF

De acordo com [Hübner 2003], estabelecer procedimentos para realizar determinadas atividades é uma forma de melhorar a eficiência de uma sociedade, da mesma forma como ocorre nas sociedades na natureza. A EF no MOISE⁺ é constituída por um conjunto de ES (*schemes sociais*), que é um conjunto de **metas** estruturado por meio de **planos**. As Metas Globais representam o estado do mundo que é desejado pela organização, diferenciando-se da meta local pelo fato desta ser de um único agente. Cada meta global g (no MOISE⁺) é associada a uma combinação de três valores que indicam:

1. Nível de satisfatibilidade: indica se a meta já foi alcançada - valor *satisfied* ou *unsatisfied*, ou a meta é impossível de ser alcançada - valor *impossible*.
2. Nível de alocação: indica se já existe algum agente comprometido com a satisfação da meta - valores *committed* e *uncommitted*.
3. Nível de ativação: indica se as pré-condições necessárias para que a meta seja satisfeita estão presentes - valor *permitted* e *forbidden*. Por exemplo, meta “inscrição no projeto” é *forbidden* até que a inscrição seja efetivada na secretaria.

O valor inicial de uma meta é: *unsatisfied*, *uncommitted*, *forbidden* e seu valor vai sendo alterado no decorrer do funcionamento do sistema.

O **nível individual (missões)** representa um conjunto coerente de metas globais que podem ser atribuídas a um agente através de um de seus papéis. O agente que se compromete com uma missão é responsável pela satisfação de todas as metas desta missão, podendo ser ele mesmo a executá-las ou passar a outro agente esta tarefa (Fig. 6).

O **nível coletivo (esquema social)** é uma árvore de decomposição de metas globais na qual a raiz é a meta do ES e a decomposição da meta é feita através de planos

$$gt \mathcal{HSJ} = \langle$$

$\{\mathcal{P}_{aspirante\ a\ hortelao}\},$	$\%R$
$\{gt_{PARCELA}, gt_{ONG}\},$	$\%SG$
$\{\},$	$\%L^{intra}$
$\{link(\mathcal{P}_{hortelao}, \mathcal{P}_{hortelao}, com), link(\mathcal{P}_{hortelao}, \mathcal{P}_{hortelao\ auxiliar}, com)\}$	$\%L^{inter}$
$\{\},$	$\%C^{inter}$
$\{\mathcal{P}_{hortelao\ auxiliar} \bowtie \mathcal{P}_{aspirante\ a\ hortelao}\},$	$\%C^{intra}$
$\{\mathcal{P}_{aspirante\ a\ hortelao} \mapsto (1, 1)\}$	$\%np$
$\{gt_{PARCELA} \mapsto (1, 42), gt_{ONG} \mapsto (1, 1)\}$	$\%ng$

$$gt \mathcal{ONG} = \langle$$

$\{\mathcal{P}_{tecnico}, \mathcal{P}_{secretaria}, \mathcal{P}_{administracao}\},$	$\%R$
$\{\},$	$\%SG$
$\{link(\mathcal{P}_{administracao}, \mathcal{P}_{tecnico}, aut), link(\mathcal{P}_{tecnico}, \mathcal{P}_{tecnico}, com),$	
$link(\mathcal{P}_{administracao}, \mathcal{P}_{secretaria}, aut), link(\mathcal{P}_{secretaria}, \mathcal{P}_{administracao}, com),$	
$link(\mathcal{P}_{tecnico}, \mathcal{P}_{administracao}, com), link(\mathcal{P}_{tecnico}, \mathcal{P}_{secretaria}, com),$	
$link(\mathcal{P}_{secretaria}, \mathcal{P}_{aspirante\ a\ hortelao}, com)\}$	$\%L^{intra}$
$\{link(\mathcal{P}_{administracao}, \mathcal{P}_{hortelao}, aut)\}$	$\%L^{inter}$
$\{\},$	$\%C^{intra}$
$\{\},$	$\%C^{inter}$
$\{\mathcal{P}_{tecnico} \mapsto (1, 3), \mathcal{P}_{secretaria} \mapsto (1, 1), \mathcal{P}_{administracao} \mapsto (1, 1)\}$	$\%np$
$\{\}\}$	$\%ng$

$$gt \mathcal{PARCELIA} = \langle$$

$\{\mathcal{P}_{hortelao}, \mathcal{P}_{hortelao\ auxiliar}\},$	$\%R$
$\{\},$	$\%SG$
$\{link(\mathcal{P}_{hortelao}, \mathcal{P}_{hortelao\ auxiliar}, aut), \}$	$\%L^{intra}$
$\{\},$	$\%L^{inter}$
$\{\},$	$\%C^{intra}$
$\{\},$	$\%C^{inter}$
$\{\mathcal{P}_{hortelao} \mapsto (1, 30)\}$	$\%np$
$\{\}\}$	$\%ng$

Figura 5. Especificação Estrutural da HSJ

denotados através do operador “=” . Por exemplo, na Fig. 6, onde
 $\text{manter a horta} = g_{\text{tecnico}}, g_{\text{aspirante}}, g_{\text{administracao}},$
 $g_{\text{secretaria}}, g_{\text{hortelao}}, g_{\text{hortelaoauxiliar}}$

a meta *manter a horta* é decomposta em 6 sub-metas indicando que ela será satisfeita como resultante de suas sub-metas

$g_{\text{tecnico}}, g_{\text{aspirante}}, g_{\text{administracao}}, g_{\text{secretaria}}, g_{\text{hortelao}}, g_{\text{hortelaoauxiliar}}$ também serem satisfeitas.

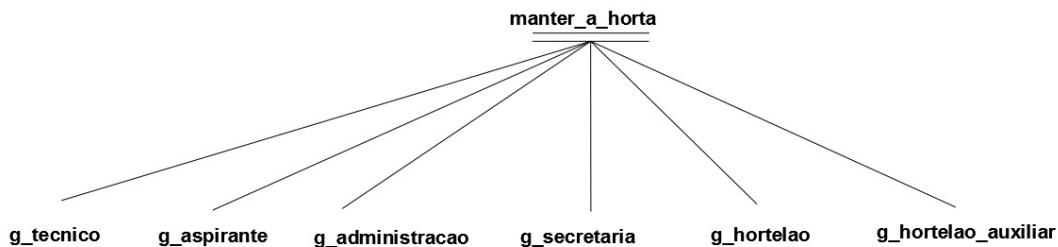


Figura 6. Esquema Social parcial da Estrutura da HSJ com modelo MOISE⁺

3.2.1. Operadores Usados na Construção de um Plano

São três os operadores para construção de um plano:

Sequência “;”: a Fig. 7 mostra um exemplo usado na especificação do estudo de caso da HSJ. O plano “*sazonais administração* = *decidir o que fazer com parcelas reservadas, decidir o que fazer com parcelas de desistencia*” significa que a meta “*sazonais administracao*” será satisfeita se a meta “*decidir o que fazer com parcelas reservadas*” for satisfeita e depois também a meta “*decidir o que fazer com parcelas de desistência*” o for. O nível de satisfatibilidade de “*decidir o que fazer com parcelas reservadas*” é dada por:

$$\begin{aligned} \text{isSatisfied}(\text{sazonaisadministracao}) &\Leftarrow \\ \text{isSatisfied}(\text{decidir o que fazer com parcelas reservadas}) \wedge \\ \text{isSatisfied}(\text{decidir o que fazer com parcelas de desistencia}) \end{aligned}$$

Escolha “|”: o plano $gx = gy|gz$ significa que a meta gx será satisfeita se uma, e somente uma, das metas gy ou gz for satisfeita, isto é:

$$\begin{aligned} \text{isSatisfied}(gx) &\Leftarrow \text{isSatisfied}(gy) \wedge \neg \text{isSatisfied}(gz) \\ &\quad \neg \text{isSatisfied}(gy) \wedge \text{isSatisfied}(gz) \end{aligned}$$

Paralelismo “||”: o plano $g_{\text{administracao}}$ é atingido seguindo um plano de paralelismo, significando que a meta $g_{\text{administracao}}$ será satisfeita quando ambas as metas “*sequenciais administração*” e “*sazonais administração*” também o forem. Contudo, ao contrário da escolha, as duas ou mais sub-metas podem ser buscadas em paralelo. Tem-se que:

$$\begin{aligned} \text{isSatisfied}(g_{\text{administracao}}) &\Leftarrow \text{isSatisfied}(\text{sequenciaisadministracao}) \wedge \\ &\quad \text{isSatisfied}(\text{sazonaisadministracao}) \end{aligned}$$



Figura 7. Esquema Social parcial da Estrutura usando operador Sequencial

4. Restrições da Modelagem para Sistemas Sociais

Durante a modelagem do sistema social da HSJ através do modelo MOISE⁺ foram encontradas algumas dificuldades, sendo necessário realizar adaptações ao modelo. Em sistemas como o da HSJ, muitos processos são realizados de forma repetida ou periódica, ou durante a vida do sistema (i.e., enquanto o sistema existir, tal processo continuará a ser executado). Tais objetivos podem ser chamados de objetivos de manutenção, denotando um processo que deve ser executado durante toda a existência do sistema (como o respeito às regras do regulamento da HSJ). O modelo MOISE⁺ não suporta processos de manutenção, não tendo estruturas que permitam que um objetivo seja atingido múltiplas vezes sem ser considerado satisfeito.

Além disso, o modelo não suporta a situação onde um objetivo pode ser realizado através de duas ou mais maneiras diferentes, sem a necessidade do cumprimento de todas. Atualmente, as estruturas disponíveis são de escolha ou paralelismo, onde a primeira exige que apenas uma das alternativas seja realizada e a última que todas sejam cumpridas, mas em qualquer ordem. A abordagem inicial para a modelagem dos processos no sistema social seria organizá-los independente de papéis, ou seja, organizá-los em termos do que ocorreria em sequência ou em paralelo. Isso não foi possível devido à falta de estruturas no modelo.

A opção de projeto adotada para contornar essa situação foi levar parte do sincronismo e coordenação dos processos para dentro dos agentes. Dessa forma, o agente deverá procurar informações dentro da sua sociedade para inferir se um objetivo pode ser atingido naquele instante ou não. Isso envolverá a recuperação de informação em outros agentes e no ambiente. Embora se afaste da abordagem de se modelar a população de agentes e sua organização separadamente, essa é a solução adotada até o momento.

5. Conclusão e Trabalhos Futuros

O modelo organizacional Moise+ contribui na modelagem da organização de um SMA, pois permite um detalhamento dos papéis sociais e suas relações. Isso pode ser observado nos exemplos demonstrados anteriormente, através de sua estrutura de ligações entre os papéis e um conjunto de planos globais (*schemes*), juntamente com a ligação destes como

os papéis definidos.

Este trabalho apresentou duas das três dimensões do modelo MOISE⁺: especificação estrutural e funcional. Para essas definições, percebeu-se que o modelo MOISE⁺ apresentou algumas limitações, que dificultaram a modelagem do sistema social da HSJ. Tais limitações incluem a inexistência da relação de disjunção entre objetivos e a representação de objetivos de manutenção. A inclusão de tais estruturas permitiria que o modelo fosse aplicado mais facilmente a sistemas sociais, já que tais situações são muito comuns neste tipo de sistema.

O esquema social e modelagem da dimensão deôntica da HSJ serão tema de artigo futuro, que apresentará a modelagem completa da organização. Como trabalho futuro, a implementação de um sistema de simulação multiagente para o modelo proposto será desenvolvida.

Referências

- Bettini, V. (1998). *Elementos de Ecología Urbana*. Editorial Trotta, Madrid.
- Dimuro, G., Dimuro, G., Costa, A. C. R., Pinheiro, T. V. T., Grol, C. V., Rodrigues, T., and Santos, F. C. P. (2011). Modelagem do sistema multiagente para simulação de processos de gestão social em ecossistemas urbanos, estudo de caso: Horta San Jerónimo. Relatório Técnico, FURG/Universidad de Sevilla.
- Dimuro, G. and Jerez, E. M. (2011). La comunidad como escala de trabajo en los ecosistemas urbanos. *Revista Ciencia y Tecnología*, 10:101–116.
- Gilbert, N. and Troitzsch, K. G. (1999). *Simulation for the social scientist*. Open University Press, Philadelphia.
- Hübner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes*. Tese de Doutorado, USP, São Paulo.
- Hübner, J. F., Sichman, J. S., and Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in MAS. In Lugo, G. A. G., Hübner, J. F., and Tacla, C. A., editors, *Brazilian Symposium on Artificial Intelligence - SBIA 2002, Porto de Galinhas*, number 2507 in LNAI, pages 118–128, Berlin. Springer.
- Lobo, C. G. (1998). *Vivienda y Ciudad Posibles*. Escala, Bogotá.
- Morin, E. (2010). *¿Hacia el abismo? Globalización en el siglo XXI*. Paidós, Madrid.
- Naredo, J. M. (1996). Sobre el origen, el uso y el contenido del término sostenible. Disponível em <http://habitat.aq.upm.es/cs/p2/a004.html>.
- Ortiz, E. (2010). Derecho a la ciudad, producción social y gestión participativa del hábitat. la promoción de iniciativas comunitarias incluyentes en la ciudad de méxico. *Hábitat y Sociedad*, 1:55–70. Disponível em <http://habitatsociedad.us.es>.
- Pelli, V. (2007). *Habitar, participar, pertenecer. Acceder a la vivienda - incluirse en la sociedad*. Nobuko, Buenos Aires.
- Pelli, V. (2010). La gestión de la producción social del hábitat. *Hábitat y Sociedad*, 1:39–54. Disponível em <http://habitatsociedad.us.es>.
- Romero, G., Mesías, R., Enet, M., Oliveras, R., García, L., Coipel, M., and Osorio, D. (2004). *La participación en el diseño urbano y arquitectónico en la producción social del hábitat*. CYTED-HABYTED-Red XIV.F, Mexico.
- Santos, I. S., Rodrigues, T. F., Dimuro, G. P., Costa, A. C. R., Dimuro, G., and Manuel, E. (2012). Towards the modeling of the social organization of an experiment of social management of urban vegetable gardens. In *WESAAC 2011 Post-Proceedings*. IEEE, Los Alamitos. (to appear).
- Terradas, J. (2001). *Ecología urbana*. Rubes Editorial, Barcelona.
- Touraine, A. (2005). *Un nuevo paradigma para comprender el mundo de hoy*. Ediciones Paidós Ibérica S.A., Barcelona.
- Viera, L. and Bredariol, C. (1998). *Cidadania e política ambiental*. Editorial Record, RJ.

TITAN: Um jogo de estratégia simulado utilizando conceitos de sistemas multiagentes

Maicon R. Zatelli¹, José R. F. Neri¹, Daniela M. Uez¹, Rafael F. Callegaro¹

¹ Departamento de Automação e Sistemas

Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

{maicon, dmuez}@das.ufsc.br, {jrf.neri, rafaelfrizzocallegaro}@gmail.com

Resumo. *O paradigma de agentes está sendo cada vez mais utilizado na área de desenvolvimento de jogos. As características dos agentes, como autonomia, proatividade e interação social, podem auxiliar no desenvolvimento de jogos mais realistas e detalhados. Este trabalho tem como objetivo integrar diversas ferramentas e conceitos da área de IA para o desenvolvimento de um jogo de estratégia. Além de agentes, foram utilizados os conceitos de organização, ambiente, reputação, sistemas especialistas e ontologias.*

1. Introdução

O paradigma de agentes tem sido cada vez mais utilizado pelos desenvolvedores de jogos. As características dos agentes, como autonomia, proatividade e interação social, podem auxiliar no desenvolvimento de jogos mais realistas e detalhados [Carvalho 2004]. Além disso, jogos são cenários perfeitos para permitir o uso conjunto de diversos conceitos de Inteligência Artificial (IA). Esta é a principal motivação que levou ao desenvolvimento deste trabalho: integrar diversas ferramentas e conceitos da área de IA para solução de um problema. Desta forma optou-se pelo desenvolvimento de um jogo de estratégia, chamado Titan. O desenvolvimento do jogo ocorreu no contexto de uma disciplina e utilizou-se duas diferentes arquiteturas de agentes (reativa e BDI), os agentes foram desenvolvidos em diferentes linguagens de programação (JADE e Jason) e também foram utilizados os conceitos de organização, ambiente, reputação, ontologias e sistemas especialistas.

Este documento está estruturado em seis seções. Na primeira foi dada uma breve introdução a respeito do objetivo deste artigo. Na segunda é apresentada uma visão geral sobre o jogo. Na terceira é descrito como o jogo foi desenvolvido, detalhes da implementação e onde cada ferramenta foi utilizada. Na quarta são mostrados os principais desafios enfrentados durante o desenvolvimento. Por fim, na quinta seção, são enfatizadas as principais contribuições e na última seção estão as considerações finais.

2. Definição do Problema

Titan é um jogo de estratégia onde as equipes disputam o domínio de Titan, um satélite que orbita o planeta Saturno. A história se passa no futuro, quando os recursos na Terra escassearam e os humanos buscam em outros lugares do universo os recursos que faltam. O principal objetivo do jogo é garantir o domínio sobre Titan através da destruição das equipes inimigas. Titan possui diversos poços de recursos que as equipes devem encontrar e explorar. É pela obtenção destes recursos que é possível ampliar os exércitos. Cada equipe é composta por um ou mais times aliados, que trocam informações sobre o ambiente e sobre os adversários. Por sua vez, um time é composto por uma base, um conselho,

capitães, soldados, robôs, naves e exploradores. Além disso, cada time possui um *blackboard*, que nada mais é que um repositório de informações globais usado com o objetivo de permitir que os agentes compartilhem informações com os demais agentes do time, e um sistema especialista (SE), para auxiliar nos planos de ataque.

Inicialmente, um time tem uma base, que é responsável pelo gerenciamento dos seus recursos e também pela criação do conselho. O conselho é um agente que tem como responsabilidade organizar a defesa e o ataque, obedecendo às estratégias definidas pelo capitão e negociadas com os aliados. No decorrer do jogo, guerreiros são criados de acordo com as necessidades de ataque ou defesa de cada time, se a base possuir quantidade suficiente de recursos para criá-los. Os guerreiros sabem quem são seus aliados, movem-se pelo ambiente, atiram, reconhecem os inimigos, os poços e os obstáculos existentes. Existem cinco tipos de guerreiros: robô, nave, explorador, soldado e capitão. Cada guerreiro tem uma função diferente no cenário do jogo e também características distintas, conforme listado abaixo:

- Soldado: é o guerreiro responsável pela defesa da base e pelo cumprimento das missões de ataque;
- Robô: é um guerreiro que possui as mesmas responsabilidades do soldado, porém tem mais força do que ele;
- Nave: é um tipo de guerreiro que tem a capacidade de sobrevoar o ambiente. As naves têm um campo de visão maior do que os guerreiros terrestres e não são impedidas pelos obstáculos, podendo voar sobre os inimigos, bases e poços. Seu principal objetivo é explorar o ambiente, localizando as bases inimigas e poços;
- Explorador: é o guerreiro cuja função principal é extrair recursos dos poços encontrados no ambiente;
- Capitão: é o guerreiro que tem como responsabilidade planejar e negociar ataques juntamente com outros capitães aliados. O capitão tem acesso a uma base de conhecimento que permite a ele planejar os ataques e também pode, quando necessário, criar outros guerreiros.

Os guerreiros utilizam o *blackboard* para armazenar as informações sobre a localização dos poços, dos inimigos e dos aliados que são vistos no ambiente. Sempre que encontrar um poço ou um inimigo que ainda não tinha sido descoberto, os guerreiros disponibilizam essa informação no *blackboard* para o restante do time. Essas informações são usadas pelo capitão para, juntamente com o SE, decidir se um ataque deverá ou não ser realizado.

Diferentemente do que acontece com outros jogos desenvolvidos com agentes, Titan é um jogo simulado. Os times iniciam ataques, organizam defesas e buscam poços sem necessidade de intervenção de um jogador humano. A única interação humana no jogo acontece na configuração inicial do ambiente, onde o jogador define a quantidade de times e de equipes existentes e, se desejar, o SE utilizado pelos times. Depois de iniciado o jogo, as batalhas se desenrolam sem necessidade de intervenção humana até que uma das equipes seja vencedora.

3. Desenvolvimento

Os agentes foram implementados utilizando diferentes arquiteturas. A base é um agente reativo implementado em JADE [Telecom 2011], enquanto que os guerreiros são agentes

BDI implementados em Jason [Bordini et al. 2007]. Outros conceitos da área de agentes foram utilizados no desenvolvimento do jogo. O capitão, por exemplo, tem acesso a um SE para decidir quando os ataques devem ser realizados e um *blackboard* é utilizado para auxiliar os times a montar uma visão global do ambiente. Os agentes de times aliados cooperam entre si para realizar ataques contra as bases inimigas, os quais são negociados pelos capitães de cada time. Então o capitão repassa as decisões da negociação ao conselho, que é responsável por organizar as unidades de ataque para que realizem o ataque. Ao mesmo tempo, o conselho cria unidades de defesa a fim de garantir a defesa da própria base. Assim, ao mesmo tempo que existe uma hierarquia entre os agentes, muitas decisões também podem ser tomadas pelos guerreiros independentemente de ordens recebidas de um agente superior.

3.1. O ambiente

De forma geral, o ambiente é o espaço onde todos os agentes vivem, ou seja, onde percebem e atuam [Demazeau 1995]. O ambiente do jogo foi desenvolvido utilizando o CArtAgO [Ricci 2011]. Todo o ambiente é implementado dentro de um único artefato e nele estão os times, os agentes, os poços, os obstáculos, o *blackboard* e o SE. É neste artefato que está descrito o comportamento do ambiente, permitindo que os agentes realizem uma série de operações e percepções. Entre elas pode-se citar a locomoção pelo ambiente, extração de recursos, percepção de outros agentes ou bases, ataque a agentes ou bases, percepção de obstáculos, entre outros.

3.2. A organização

Numa organização há uma espécie de união entre os agentes que formam um grupo. A organização pode ser usada pra representar objetivos globais e os agentes passam a se relacionar para atingirem esses objetivos. Como metas locais e globais nem sempre são compatíveis, uma organização fornece uma visão geral para verificar se as ações dos agentes respeitam os objetivos globais ou não [Gers 2002].

Os agentes que formam um time precisam trabalhar conjuntamente para que o objetivo de destruir o inimigo seja alcançado. Para garantir a organização desse sistema utilizou-se o modelo organizacional Moise [Hubner 2003]. Conforme pode ser visto na Figura 1, na especificação estrutural foram especificados quatro grupos. O grupo `time` (grupo raiz) é composto pelos subgrupos `exploração`, `defesa` e `ataque`. Uma instância do grupo `time` deve possuir uma única instância do grupo `exploração`, uma do grupo `defesa` e uma do grupo `ataque`. Além disso, o grupo precisa ter um conselho mas não precisa necessariamente ter um capitão, já que, durante o jogo, o capitão pode ser morto e o grupo ficará sem capitão até que um novo seja criado. Os agentes do grupo `exploração` podem assumir o papel de nave ou o papel de explorador e a nave tem autoridade sobre o explorador. O grupo `defesa` é composto de zero ou infinitas unidades de defesa e o grupo `ataque` é composto de zero ou infinitas unidades de ataque.

Os papéis de explorador, nave, unidade de defesa, unidade de ataque e capitão herdam as características do papel abstrato guerreiro e todos os guerreiros podem se comunicar entre si. Agentes do tipo robô e soldado podem assumir o papel de unidade de ataque ou unidade de defesa, de acordo com a necessidade do time. O papel assumido por um agente é definido pelo conselho no momento em que o agente é criado. Se uma

unidade é criada e a defesa já está completa, a unidade assume o papel de ataque. Finalmente, o conselho tem autoridade sobre os guerreiros e estes têm conhecimento do conselho, enquanto que o capitão tem autoridade sobre as unidades de ataque e de defesa.

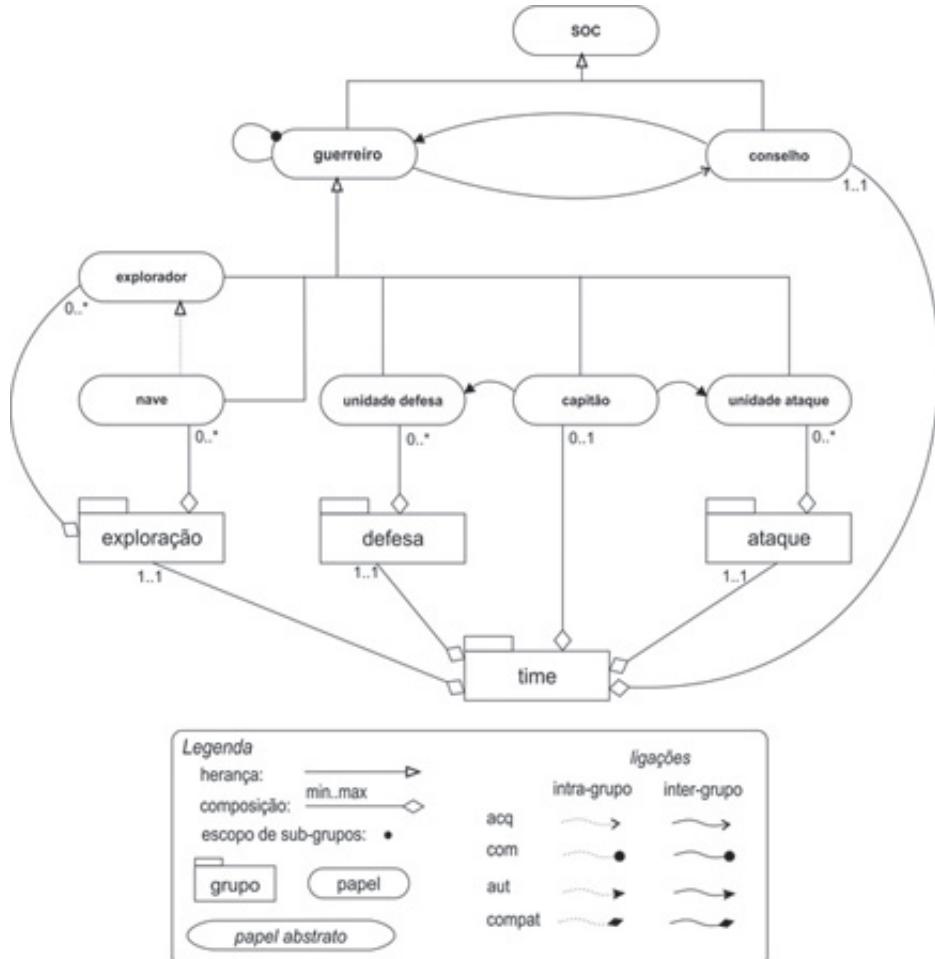
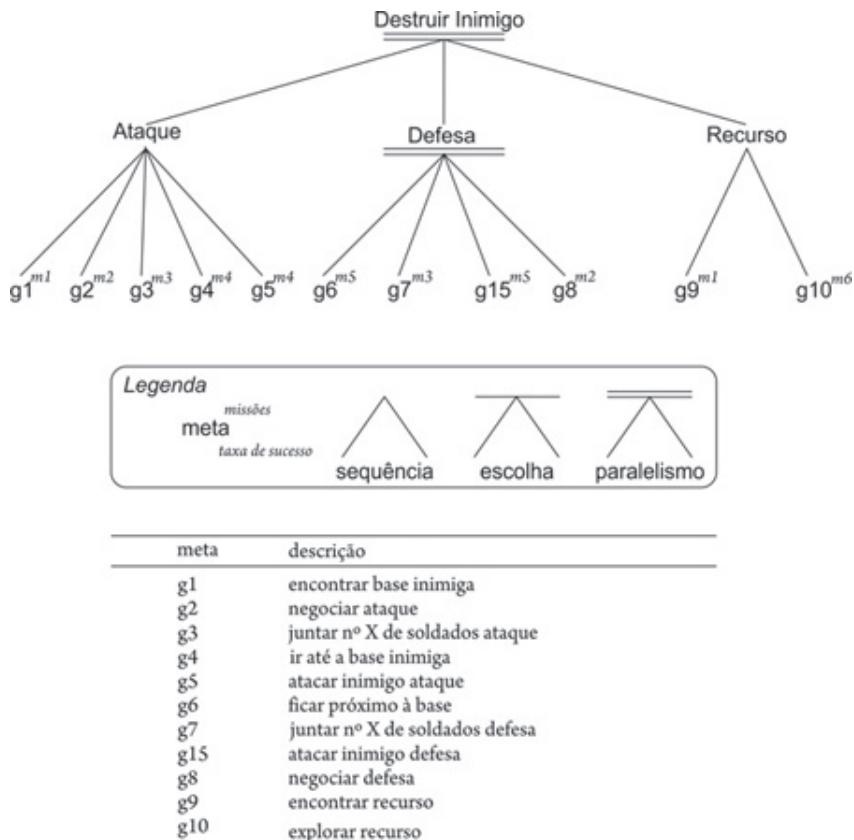


Figura 1. Especificação estrutural para o jogo Titan.

A Figura 2 apresenta um esquema social definindo a especificação funcional do jogo. Na raiz do esquema tem-se a meta global de destruir o inimigo. Para satisfazer a meta global, três submetas em paralelo precisam ser satisfeitas: a meta **ataque**, a meta **defesa** e a meta **recurso**. Essas três metas são decompostas em três planos, e esses planos são compostos por metas folhas. As metas dos planos **ataque** e **recurso** devem ser alcançadas em forma sequencial, ou seja, só é possível alcançar a meta **g2** depois que a meta **g1** for alcançada. Já as metas do plano **defesa** podem ser alcançadas de forma paralela, ou seja, pode-se alcançar a meta **g8** e **g6** ao mesmo tempo.

As missões são assumidas pelos agentes no momento em que estes assumem um papel em um grupo. Conforme especificado na Figura 2, o agente que se compromete com a missão **m1** é responsável pela satisfação de todas as metas desta missão, no caso, as metas **g1** e **g9**. Já o agente que assumir a missão **m2** deve alcançar as metas **g2** e **g8**.

A relação entre a especificação estrutural e especificação funcional é feita pela especificação normativa. Na especificação normativa são descritas as missões com as

**Figura 2. Especificação funcional para o jogo.**

quais um papel tem permissão ou obrigação de se comprometer. Na Figura 3 é possível observar que o papel de nave está obrigado a alcançar as metas da missão m1, o capitão tem obrigação de alcançar a missão m2, e assim por diante.

papel	relação deônica	missão
nave	obrigação	m1
capitao	obrigação	m2
conselho	obrigação	m3
unidade ataque	obrigação	m4
unidade defesa	obrigação	m5
explorador	obrigação	m6

Figura 3. Especificação normativa para o jogo.

3.3. Comunicação entre agentes

Um dos aspectos mais importantes no desenvolvimento de um SMA é a interação entre os agentes. O principal mecanismo que permite esta interação é a comunicação através da troca de mensagens. Como sistemas multiagentes nem sempre são sistemas fechados, desenvolvidos por um único grupo de desenvolvedores, foi preciso criar protocolos padronizados objetivando que os agentes possam se comunicar com outros agentes mesmo que não tenham sido desenvolvidos especificamente para interagirem entre si. Algumas linguagens, como KQML, KIF e a FIPA-ACL, foram criadas com este fim. Neste tra-

lho utilizou-se duas dessas linguagens. A KQML é utilizada pela ferramenta Jason e o padrão FIPA-ACL é utilizado pela ferramenta JADE.

A *Knowledge Query and Manipulation Language* (KQML) é uma linguagem externa para a comunicação de agentes. Essa linguagem define um formato de envelope para as mensagens, no qual um agente pode explicitar a força das intenções ilocucionárias da mensagem. Cada mensagem tem uma performativa, que pode ser qualquer classe de mensagens, e um número de parâmetros, compostos por pares de atributos e valores [Wooldridge 2009].

Por sua vez, a *Foundation for Intelligent Physical Agents*(FIPA) desenvolveu a linguagem FIPA-ACL que é similar à linguagem KQML. A linguagem FIPA-ACL também define uma linguagem externa para as mensagens, porém diferentemente da KQML, a FIPA ACL possui somente 20 performativas para a interpretação das mensagens e não obriga uma linguagem específica para o conteúdo das mensagens [Wooldridge 2009].

Outra forma utilizada para comunicação entre os agentes é através do uso de um sistema de *blackboard*. Um sistema de *blackboard* é composto por uma coleção de entidades conhecidas como fontes de conhecimento, cada uma possuindo um conhecimento especializado, e uma estrutura de dados compartilhada que essas fontes de conhecimento utilizam para a comunicação. As fontes de conhecimento, neste caso os agentes, são capazes de ler e escrever no *blackboard*, assim contribuindo com uma solução parcial do problema [Wooldridge 2009].

No jogo, os agentes utilizam estas duas formas de comunicação: a troca de mensagens e o *blackboard*. A troca de mensagens é utilizada pelos agentes para comunicar o conselho quando nascem ou morrem, quando a base inimiga foi destruída e para incluir informações no *blackboard*. Além disso, os capitães usam mensagens para negociar os ataques com os aliados, os quais podem ser realizados com base nas informações do *blackboard* e com ajuda do SE.

3.3.1. Negociação

Negociação é uma técnica utilizada para que agentes possam chegar a um acordo sobre algum assunto de interesse mútuo. A negociação normalmente procede em uma série de etapas, com todos os agentes envolvidos fazendo uma proposta em cada etapa. As propostas dos agentes são feitas de acordo com a estratégia definida por eles, porém deve seguir um protocolo, ou seja, deve respeitar um conjunto de propostas possíveis em cada etapa. Se um acordo é alcançado então a negociação termina. Um exemplo de uma questão simples que envolve um cenário de negociação é onde dois agentes estariam negociando um preço de alguma coisa [Wooldridge 2009].

A fim de coordenar os diversos times aliados durante os ataques, o jogo utiliza uma espécie de negociação. A negociação é feita entre os capitães de times aliados. Um capitão só pode entrar em uma negociação se não estiver em outra negociação e se o time possuir unidades de ataque livres, ou seja, que não foram designadas para nenhuma missão. Antes de iniciar a negociação, o capitão precisa saber a quantidade de guerreiros conhecidos dos inimigos e quem são os comandantes aliados. De posse dessas informações o capitão inicia a negociação, que ocorre em quatro fases:

1. Início: o capitão informa aos aliados qual o inimigo que será atacado;
2. Inventário: levantamento da quantidade de guerreiros de todos os aliados que podem fazer parte do ataque;
3. Definição: consulta ao plano de ataque para saber se o ataque deve ser realizado ou não;
4. Ataque: envio de notificação às unidades para realizar o ataque.

A negociação é abortada em cada fase se algum aliado recusar a participar do ataque. Nesse caso, a negociação é reiniciada com os capitães que estão dispostos a colaborar. Por fim, sempre que a negociação for cancelada e reiniciada, todos os capitães participantes serão notificados.

3.3.2. Reputação dos agentes

Bromley define reputação como sendo uma ferramenta social que objetiva reduzir a incerteza na interação entre indivíduos com atributos desconhecidos [Bromley 1993]. A interação entre os agentes é um aspecto inerente aos sistemas multiagentes, visto que os agentes precisam interagir para que os objetivos globais ou particulares possam ser atingidos. Assim, é comum que os sistemas multiagentes utilizem modelos de reputação para auxiliar os agentes na hora de determinar quem será seu parceiro de negócios. Nesse sentido, a reputação pode ser interpretada como um valor, determinado pela sociedade, que demonstra o grau de confiabilidade de um determinado agente.

No jogo, a reputação foi utilizada para permitir que os capitães escolham, dentre seus aliados, aqueles que estão mais dispostos a cooperar com o ataque. A reputação é calculada da seguinte maneira: sempre que um time se dispuser a ajudar os aliados durante uma negociação, um ponto será acrescido à sua reputação. Se o time não puder ajudar, um ponto será descontado de sua reputação. Caso a negociação seja finalizada com sucesso, ou seja, se ao final da negociação os aliados decidirem atacar o inimigo, o time que iniciou o processo é recompensado com o aumento de um ponto de reputação para cada time aliado que participou do processo. Já no caso da negociação fracassar e não haver ataque, nenhum time participante é pontuado. O uso do sistema de reputação diminui a quantidade de mensagens trocadas e torna a negociação mais rápida. Nesse processo, times que não estão dispostos a colaborar têm uma reputação baixa e, portanto, não são contados pelos aliados no início da negociação.

3.4. Sistema Especialista

Um sistema especialista é um sistema que possui a capacidade de emular a habilidade de decisão de um especialista humano sobre um certo domínio de conhecimento. Um exemplo clássico de um SE é o MYCIN [Buchanan and Shortliffe 1984], que pretendia dar assistência no tratamento de infecções sanguíneas em humanos [Jackson 1998].

No jogo, o capitão tem acesso a um SE que o auxilia a definir se o ataque a determinada base inimiga deve ou não ser realizado, levando em conta a quantidade de guerreiros disponíveis, a quantidade de guerreiros que o inimigo possui, recursos, entre outros dados relevantes sobre os times. Cada time pode utilizar um SE próprio, o que diferencia o comportamento dos capitães durante a negociação. Para descrever o SE utilizou-se a ferramenta Jess [Friedman-Hill 2011].

3.5. Ontologia

Uma ontologia descreve os conceitos de um domínio de conhecimento e define relações entre eles. O principal objetivo da ontologia é capturar o conhecimento consensual sobre algum domínio de interesse. Studer define ontologia como "uma especificação explícita e formal de uma conceitualização compartilhada". Ela deve ser compreensível, acessível e manipulável pelos agentes que farão uso da mesma [Studer et al. 1998].

No jogo, a ontologia foi utilizada pelo agente base para que este soubesse qual tipo de papel cada um dos outros agentes pode assumir. Por exemplo, o papel de capitão só pode ser assumido pelo agente do tipo capitão, o papel de explorador só pode ser assumido pelo agente do tipo explorador ou nave. Essa restrição não pode ser feita a partir do Moise e é importante para o jogo pois um agente do tipo capitão não possui as características necessárias para assumir o papel de explorador.

Para o desenvolvimento da ontologia foi utilizada a ferramenta Protégé [Research 2011], e a integração com o sistema foi feita através do uso do *reasoner* Hemit [Group 2011], que tem a finalidade de efetuar o raciocínio da ontologia. Por meio do mecanismo de inferência disponibilizado pelo Hemit foi possível inferir quais tipos de agentes podem assumir cada papel. Para que seja possível trabalhar com este recurso foi elaborada uma ontologia que detalha os equipamentos e títulos que um agente deve possuir para que possa assumir determinado papel. A Figura 4 exibe a classe *Equipamento*, que especifica a lista de equipamentos disponíveis no jogo. Cada agente pode possuir desde zero ou até vários destes equipamentos. Já a Figura 5 permite visualizar a classe *Título*, que contém a lista de títulos que uma unidade pode obter. O título é usado para permitir que um agente possa assumir os papéis de Capitão ou Conselho, que não implicam na necessidade de um agente possuir algum equipamento específico.

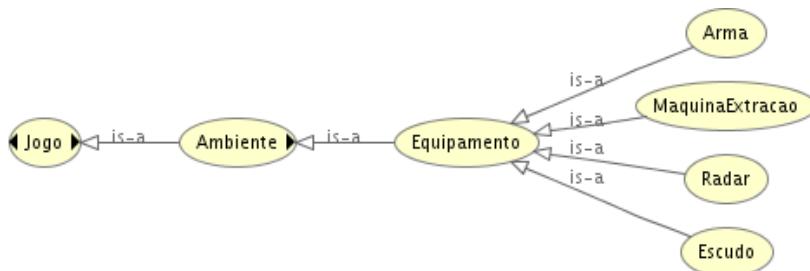


Figura 4. Classe Equipamento.

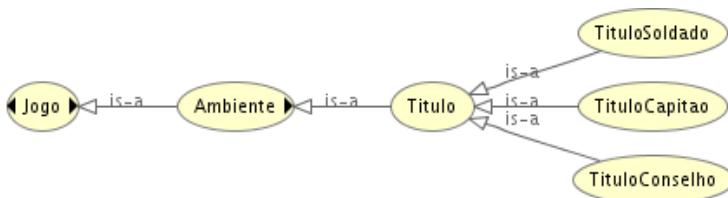


Figura 5. Classe Título.

Após a elaboração destas classes foi necessário relacionar elas por meio de propriedades. Como exemplo, um papel de ataque só pode ser assumido por um agente que

possua uma arma de equipamento, um papel de exploração só pode ser assumido por um agente que possua um radar ou uma máquina de extração de recursos, um papel de capitão só pode ser assumido caso o agente possua um título de capitão, e assim por diante. O diagrama de todas as inferências obtidas pelo *reasoner* Hemit entre papéis e agentes é mostrado na Figura 6.

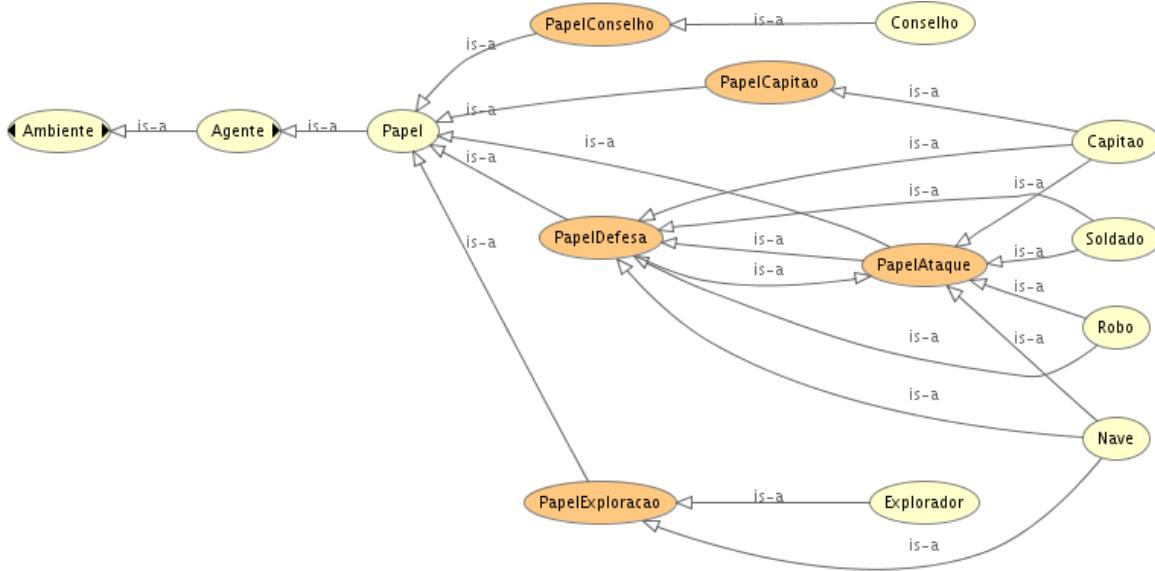


Figura 6. Ontologia inferida da classe Agente.

Na ontologia também foram implementadas a descrição dos artefatos contidos na classe Artefato e a relação de acesso a estes artefatos, ou seja, a definição de quais agentes possuem acesso um determinado artefato. A classe Artefato tem como objetivo fazer uma separação funcional do ambiente desenvolvido no CArtAgO, ilustrando que tipos de artefatos existem no jogo, visto que na implementação foi optado por utilizar um único artefato que controla todo o ambiente do jogo.

3.6. Interface Gráfica

Com a finalidade de visualizar o comportamento do jogo optou-se por desenvolver uma interface gráfica simples. Nesta interface, cada time é representado por uma cor. A Figura 7 mostra a descrição dos times que estão no jogo. Neste caso são quatro times: time 0, de cor azul, time 1, de cor verde, time 2, de cor vermelha e time 3, de cor cinza. Para cada time é apresentado a quantidade total de unidades, a distribuição dessas unidades por papéis e a lista de aliados, se houver. O time 0 (azul), por exemplo, não possui aliados. Já o time 1 (verde) possui dois aliados: o time 2 (vermelho) e o time 3 (cinza).

Na Figura 8 pode-se visualizar o ataque coordenado dos times verde e cinza contra a base do time azul. Nesta representação gráfica, os soldados e robôs de ambos os times aliados atiram contra a base. Os tiros são representados por linhas amarelas. As letras r representam robôs, os s representam soldados, o c representa o capitão, o círculo maior representa a base e o asterisco circundado representa um poço. Poços ainda não explorados são representados em amarelo, poços já explorados são representados na cor preta.

```
Id: 0 Unidades: 1 Mana: 200 Aliados: null  
PapelConselho(0)\PapelDefesa(1)\PapelExploracao(0)\PapelCapitao(1)\PapelAtaque(1)  
Id: 1 Unidades: 19 Mana: 14300 Aliados: 2 / 3  
PapelConselho(0)\PapelDefesa(18)\PapelExploracao(1)\PapelCapitao(1)\PapelAtaque(18)  
Id: 2 Unidades: 11 Mana: 460 Aliados: 1 / 3  
PapelConselho(0)\PapelDefesa(11)\PapelExploracao(0)\PapelCapitao(1)\PapelAtaque(11)  
Id: 3 Unidades: 20 Mana: 17420 Aliados: 1 / 2  
PapelConselho(0)\PapelDefesa(19)\PapelExploracao(2)\PapelCapitao(1)\PapelAtaque(19)
```

Figura 7. Descrição dos times.

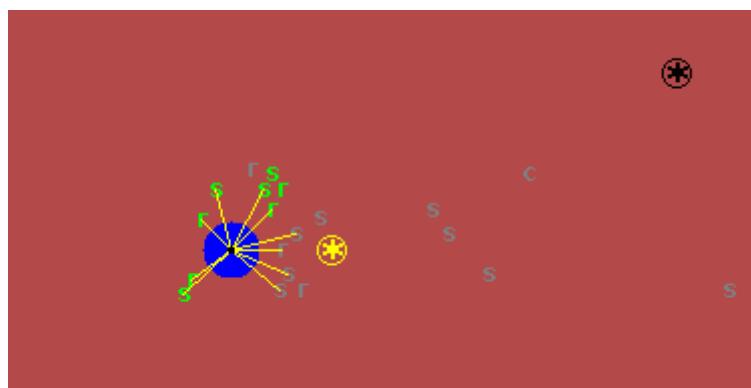


Figura 8. Visualização do jogo.

Cada agente tem seu campo de visão. O campo de visão determina a área na qual os objetos ou outros agentes podem ser vistos pelo agente. O tamanho do campo de visão depende do tipo de agente.

4. Desafios durante o desenvolvimento

Durante o desenvolvimento do jogo alguns problemas foram enfrentados. Por exemplo, quando diversos agentes precisavam acessar o ambiente desenvolvido na ferramenta CArtAgO e os acessos eram constantes, o ambiente tornava-se lento e acabava por não executar o jogo. Para corrigir o problema foi necessária uma alteração na ferramenta. No caso da interface gráfica desenvolvida em Java, a lentidão foi corrigida com a criação de uma *thread* específica para atualizar da interface. Também visando um melhor desempenho do jogo optou-se por desenvolver um único artefato contendo todas as operações que os agentes podem executar no ambiente, ao invés de diversos artefatos.

A ferramenta JADE também apresentou alguns problemas. Em primeiro lugar, descobriu-se que o JADE não interage com o ambiente CArtAgO. Esse problema teve de ser contornado com a criação de uma representação do agente JADE no ambiente do CArtAgO. Essa foi a maneira encontrada para que um agente desenvolvido em JADE pudesse interagir com o ambiente. Mesmo assim, o agente JADE tem acesso somente a algumas percepções e ações. Essa restrição precisou ser colocada pois este agente não tem mecanismos de controle de concorrência. Nos demais agentes o acesso concorrente é controlado pelo CArtAgO. O agente JADE foi concebido como um agente reativo, já que a linguagem não oferece funcionalidades para criação de agentes BDI. A principal função

deste agente é gerenciar o *blackboard*, ou seja, para atualizar ou consultar o *blackboard* é necessário enviar mensagens para o agente JADE. Este procedimento visa evitar que uma quantidade grande de acessos ao ambiente seja feita, evitando que este se tornasse um gargalo para o sistema. Outro problema apresentado foi com relação a quando uma quantidade grande de mensagens era trocada. Neste caso, a execução se tornava lenta. Por isso acabou-se por reduzir o número de mensagens trocadas entre os agentes.

Por fim, o Moise apresentou problemas quando o agente era removido do sistema. Como o agente tem obrigação de realizar uma missão, ele não pode abandoná-la. Mesmo quando um agente é removido do sistema através da ação interna `kill_agent`, o Moise não libera o papel para outro agente. Para solucionar esse problema, foi feita uma alteração no esquema do Moise para permitir que os papéis tivessem um número indefinido de agentes. O Moise também apresentou problemas com missões do tipo *maintenance*, que são missões que sempre precisam ser executadas continuamente, e missões que precisam ser executadas diversas vezes pelo mesmo agente. Esses tipos de missão não são disponibilizadas pelo Moise.

5. Contribuições do trabalho

Devido aos problemas citados acima, foi possível realizar algumas colaborações para as ferramentas utilizadas. No projeto CArtAgO foi corrigido e reportado o problema que havia em relação a lista de eventos pendentes dos agentes, na qual estes eventos nunca eram apagados, causando certa instabilidade no jogo. Algumas contribuições foram dadas também ao projeto Moise, como a sugestão de desenvolvimento de metas do tipo *maintenance*. Esse tipo de meta deve ser executada pelo agente repetidas vezes até que este abandone seu papel. É uma meta que, na prática, nunca entra em um estado de satisfeita, mantendo-se sempre ativa. Outras colaborações para o projeto Moise foram reportar os problemas que ocorriam quando um agente abandonava o sistema, considerando principalmente o fato deste agente estar comprometido com uma meta obrigatória. Quanto aos problemas encontrados durante o uso da ferramenta JADE, descobriu-se que os desenvolvedores já têm conhecimento dessas falhas.

6. Considerações finais

Com o desenvolvimento do jogo foi possível observar o uso de cada conceito e ferramenta na prática. Como cada time pode utilizar um sistema especialista próprio, diferente dos demais, o sistema acabou por permitir times com comportamentos estratégicos distintos, tornando o jogo mais interessante. O uso do ambiente CArtAgO mostrou-se de grande ajuda pois, através da simulação de um ambiente virtual por meio de artefatos, foi permitido separar de maneira coerente o desenvolvimento dos agentes dos demais objetos envolvidos no sistema, os quais não possuem autonomia. Já o Jason demonstrou ser uma linguagem adequada para o desenvolvimento de agentes, o qual, diferentemente do JADE, permitiu a criação de agentes BDI de maneira bastante simples e intuitiva. Na dimensão da organização pôde ser aproveitada a ferramenta Moise, que apesar de alguns recursos ainda não estarem disponíveis, mostrou ser útil para controlar um grupo de agentes, levando-os a cumprir as metas de uma forma mais organizada e civilizada.

A ontologia permitiu o uso de um *reasoner* para mostrar ao usuário quais são os papéis disponíveis pela organização, porém também informar quantas unidades poderiam

assumir cada papel em tempo de execução. Dessa forma foi permitido ao usuário ter uma noção do que se passa durante o jogo em termos de números de unidades. Por fim, a reputação teve um papel importante durante a negociação, reduzindo significativamente a troca de mensagens entre capitães pertencentes a times que não estavam colaborando nos combates.

7. Agradecimentos

Agradecemos aos professores Jomi Fred Hübner (UFSC) e Ricardo José Rabelo (UFSC) por comentários e sugestões a respeito do trabalho.

Referências

- Bordini, R. H., Wooldridge, M., and Hübner, J. F. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason* (Wiley Series in Agent Technology). John Wiley & Sons.
- Bromley, D. B. (1993). *Reputation, Image and Impression Management*. John Wiley & Sons.
- Buchanan, B. G. and Shortliffe, E. H. (1984). *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Carvalho, F. (2004). Comportamento em grupo de personagens do tipo black&white.
- Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. In *In: Proceedings of the 1st. European Conference on Cognitive Science*. Saint-Malo, pages 117–132.
- Friedman-Hill, E. (2011). Jess, the rule engine for the java platform. Disponível em <http://www.jessrules.com/>.
- Gers, F. (2002). *Multi-agent system for distributed data fusion in peer-to-peer environment*. PhD thesis, University of Jyväskylä.
- Group, I. S. (2011). Hermit owl reasoner. Disponível em <http://hermit-reasoner.com/>.
- Hubner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes*. PhD thesis, Escola Politécnica da Universidade de São Paulo.
- Jackson, P. (1998). *Introduction to Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition.
- Research, S. C. F. B. I. (2011). Protégé. Disponível em <http://protege.stanford.edu/>.
- Ricci, A. (2011). Cartago. Disponível em <http://cartago.sourceforge.net/>.
- Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197.
- Telecom, I. (2011). Jade - java agent development framework. Disponível em <http://jade.tilab.com/>.
- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley, 2nd edition.

A Markovian Multiagent Musical Composer

Joel L. Carbonera¹, João L. T. Silva¹

¹Universidade de Caxias do Sul, Centro de Computação e Tecnologia da Informação
Caxias do Sul, Brazil, 95700-000

{jlcarbon, jltsilva}@ucs.br

Abstract. *In this paper we propose a computational model for stochastic melodic synthesis based on an approach developed through reactive multiagent systems. In general automatic musical composition approaches, the states transition probabilities are obtained from statistical analysis of musical segments provided by the user, which tends to generate musical results very similar to the initial musical segments. As alternative to these characteristics, we consider a computational model in three layers: the first layer aims to generate transition probabilities matrices (pitches and durations of musical notes, in this context) to supply the second layer. The second layer produces probabilistic models for generation of the melodic segments (PMMG), through the interaction with the user, in order to supply the stochastic melodic synthesis process at the third layer. This model aims to reduce the influence of the composer in the final composition, in order to build melodic structures not originally conceived by the composer. Nevertheless, this work does not cope with the aesthetic quality of the melodic results, since the work focus is the spontaneous generation of the initial information that will guide the stochastic melodic synthesis.*

Resumo. *Neste artigo, é proposto um modelo computacional para síntese melódica estocástica baseado em uma abordagem desenvolvida sobre sistemas multiagentes reativos. As abordagens tradicionais de composição musical automática obtém probabilidades de transição de estados através da análise estatística de fragmentos musicais fornecidos pelo usuário, o que tende a gerar resultados musicais muito semelhantes aos fragmentos musicais iniciais. Buscando alternativas a estas características, consideramos um modelo computacional em três camadas: a primeira camada é responsável pela geração de matrizes de probabilidades de transição de estados (alturas e durações de notas musicais). A segunda camada, através da interação do usuário, gera modelos probabilísticos para geração de segmentos melódicos (MPGSM) a partir das matrizes. A terceira camada finaliza o processo de síntese melódica estocástica com estes segmentos. Este modelo tem como objetivo reduzir a influência do compositor na geração final, a fim de buscar a construção de estruturas melódicas que não foram pensadas inicialmente pelo compositor. Desta forma, este trabalho não trata computacionalmente a análise da qualidade estética do resultado melódico, uma vez que o foco é a geração espontânea das informações iniciais que nortearão a síntese melódica estocástica.*

1. Introduction

The computer music is an inherently multidisciplinary field involving all areas of knowledge related to Computer Science and Music. The computer music area involves knowledge ranging from computer modeling of musical aspects to the study of aesthetic conceptions of artistic expression musical, including the study of techniques for the generation, storage and processing of sound, etc. Many works have been produced in the area of sound and music composition processing and renowned composers have been systematically using computational tools for musical composition [Cruz 2001, Fischman 2003, Burraston and Edmond 2005, Miranda 2000, Xenakis 1992, Papadopoulos and Wiggins 1999, MacGee and Schaffer 1997, Bryan-Kinns 2004].

Many musical synthesis systems produced by traditional initiatives are usually dependent on the user. In this sense, the musical constructions are not completely automatic. Traditionally, initial samples are provided manually by the user and the system of musical synthesis is limited to process these samples or data, in a static and pre-defined way. This paper investigates a close relation between music and Artificial Intelligence from a multiagent point of view. We intend to demonstrate how the collective behavior of reactive agents can lead to a quasi-automatic compositional system.

Concerning with stochastic models to musical synthesis, our work differs from others in many aspects. For instance, in the framework for generating Palestrina-style music using Markov models [Farbood and Schoner 2001], the authors aim to generate counterpoint segments rules from state-transition matrices. In this work, probabilistic counterpoint rules are implemented as a probability table starting by probability zero when the system finds illegal transitions. Thus, the generation of a counterpoint line is obtained by multiplying the individual values from each transition probability table, where each table captures one aspect of counterpoint line, as for instance, harmonic table, melodic table, cadence table, chromatic table and so on. The main idea is to analyze a previous composition and then to infer a probabilistic description of counterpoint rules. As argued by the authors, the final result can be comparable to those created by a knowledgeable musician, corroborating our claim about the similarity between the input and output compositions. Also, McCormack [McCormack 1996] has pointed some drawbacks on Markov process for music composition related to the generality of the transition table and the management of higher order models. When some real existing music is used as input into the system, the result is a Markov model which only generates music of the same style of the input. Also, higher order Markov models will require extra computational effort that can hinder real time performance and, also, probably to provide little support for structure at higher levels. In these cases, multiple layered models can be used where each event represents an entire Markov model in itself or another type of structure. In our work, we present a layered architecture which may cope in the future with these problems related to higher order Markov models.

2. Model Overview

The three layered architecture proposed in this work is illustrated in Figure 1. The lower layer (*Reactive generation of transition probability matrices*) encapsulates the process of reactive generation of probability transition matrices. In this layer, a multiagent environment is responsible for stochastic generation of transition probabilities based on a

reactive behavior. Two distinctive environments are in charge of built transition probability matrices for pitches and for durations through interaction of reactive software agents. The middle layer (*Definition of probabilistic models for melodic generation*), at Figure 1, represents the process of construction of probabilistic models for generating melodic segments based on the matrices produced by the multiagent environments. Finally, the uppermost layer (*Stochastic melodic synthesis*) represents the stochastic process of melodic synthesis.

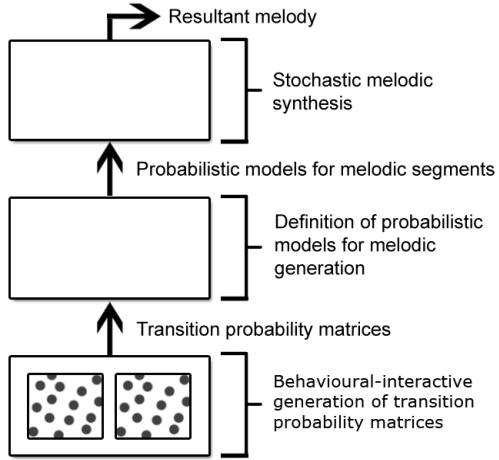


Figure 1. Three layered model of the emergent stochastic melodic composition system.

Next we present the three layers in details mainly focusing on emergent melodic generation model through reactive multiagent environments allied to Markov decision model.

2.1. Transition Probabilities Generation Layer

The state transition matrices (of pitches and durations) feed the process of stochastic melodic composition. The model to generate the probability matrices used in this work is based on dynamic interaction among reactive agents. In a computational context, reactive agents are autonomous pieces of software that act in a determined environment without human intervention. These agents are capable to interact with other agents, to perceive the environment where they are and to react based on percept stimulus without reasoning.

The multiagent environments are instances of an euclidean plane whose dimensions are defined by the user and where the agents move themselves. Each element from this euclidean plane represents $N \times N$ co-occurrence matrices where N represents the 12 pitches of the musical notes, i.e., the 12 semitones (from $C, C\#, \dots, B\flat, B$). The environment that generates duration transition probability matrices, N is 7, which represents the total number of musical notes duration in this work (*Semibreve, Minim, Crotchet, Quaver, Semiquaver, Demisemiquaver, Hemidemisemiquaver*) (see Figure 2).

On the environment, the agents pursue some goals that are user-defined points called *Target-points*. These *Target-points* are defined through a point (euclidean coordinates x, y) located inside the plane limits. The *Target-points* carrying out an attraction

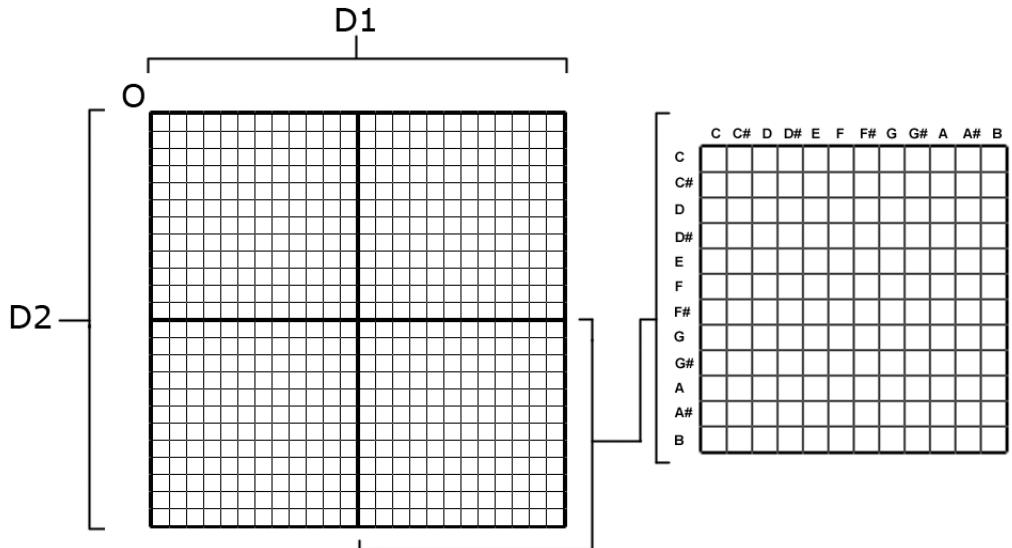


Figure 2. Example of environment structure composed by four co-occurrence matrices of pitch transition probability

force into the agents in such a way that each agent is obliged to pursue some of these *Target-points* while they move themselves randomly on the environment.

In this work, the agent behaviors are based on the subsumption architecture [Brooks 1986] and on potential fields approach [Mezencio 2002]. The behaviors are arranged in a hierarchical two level structure based on the behavior priority (Figure 3). The first level (greatest priority) represents the agent evasive behavior (including trajectory adjustment to avoid invasion of the perimeter of another agent) while the second level represents the behavior “go forward the target” (including trajectory adjustment to trace a direct line to the target-points).

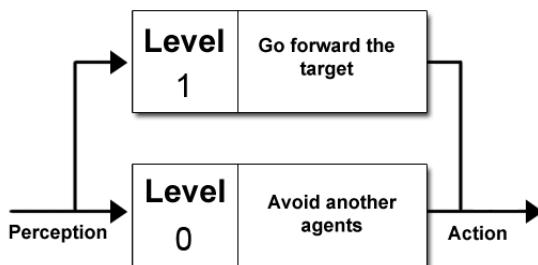


Figure 3. Hierarchical agent behaviour based on subsumption architecture.

The action of the agent consists in adjust the vector of direction and to move towards this vector at each perception-action cycle. The resultant vector is obtained through the generation of basic agent behaviours through the overlapping of the potential fields that influence the agent at some time. This overlapping is defined as the sum of the vectors of each one of the forces (repulsion - vec_rep and attraction - vec_atr) that influence the agent obtaining a vector that defines the speed and the direction of the agent movement (equations (b) and (c) on the Algorithm 1). The agent defines an action in the subsumption

hierarchy through its perception information. The *Invasion-Limit-Range (ILR)* algorithm 1 determines if some of the other agents have violated the restriction of minimal distance (r) in order to trigger an *Avoid-Agent-Collision behaviour* (equations (a) to (e) on the Algorithm 1) or a *Go-Forward behaviour* (equations (e)).

```

Input: set of perceptions ( $\mathcal{P}$ ), set of  $n$  agents ( $\mathcal{A}$ ), lim range ( $r$ )
where  $pos_{self}, pos_{target} \in \mathcal{P}$  ;
Output: direction ( $dir$ )
foreach agent  $g \in \mathcal{A}$  do
    Calculate  $ILR(g, pos_{self})$ ;
    if  $ILR(g, pos_{self}) \leq r$  then
        /* Avoid-Agent-Collision Behaviour */
         $avr\_pt = \sum_{i=1}^n pos_i/n;$  (a)
        Calculate  $vec\_rep(avr\_pt, pos_{self});$  (b)
        Calculate  $vec\_atr(pos_{self}, pos_{target});$  (c)
        Calculate  $dir = vec\_rep + vec\_atr;$  (d)
    else
        /* Go-Forward Behaviour */
        Calculate  $dir = vec\_atr(pos_{self}, pos_{target});$  (e)
    end
end

```

Algorithm 1: The *Invasion-Limit-Range (ILR)* algorithm that select the action based on the subsumption hierarchy.

Before running the system, the user provides some parameters to the environment and defines some Target-points. At the start of the execution, each agent is randomly placed over the environment and each one start their behaviors in order to reach the closer Target-point, always avoiding other agents. While moving themselves, the agents increase by one a value at the element on the co-occurrence matrices (first initialized with zero). This behaviour distributes some values over the co-occurrence matrices building the state transition matrices for pitches and durations.

2.2. The PMMSG layer

A probabilistic model to melodic segments generation (PMMSG) constitutes a data structure built from the set of transition probability matrices generated in the previous layer. The main goal associated to the PMMSG is to drive the melodic stochastic synthesis in an interactive way.

The user is able to define models that encapsulate a pair of transition probabilities matrices, which will define the distribution of pitch and duration in distinct segments of the melody. Thus, each PMMSG has the potential of generate local melodic segments with particular features, within the global melody. Formally:

Definition A Probabilistic Model to Melodic Segments Generation (PMMSG) is a couple $PMMSG\langle Q^P, Q^D \rangle$, where Q^P is a probability transition matrix of Pitches and is Q^D is a probability transition matrix of Durations.

These probability transition matrices are defined in terms of a Markov model definition. A Markov chain [Meyn and Tweedie 1993] is a specific kind of discrete-time stochastic process that given some present states, the future ones are chosen independently of the past states, which is called Markov property. In our work, the melodic synthesis is modeled as a succession of musical notes where these notes are defined by their pitch and duration.

In this work we use a traditional Markov Chain definition:

Definition A Markov Chain with state space $\mathcal{S} \subseteq \mathbb{I}$ is a discrete stochastic process $\{X_n; n \in \mathbb{I}\}$ if each X_n assumes values only in \mathcal{S} and

$$\begin{aligned} P(X_{n+1} = x | X_n = x_n, \dots, X_0 = x_0) = \\ P(X_{n+1} = x | X_n = x_n) \end{aligned} \quad (1)$$

holds for all $n \in \mathbb{I}$ and $x_0, x_1, \dots, x_n, x \in \mathcal{S}$.

Definition A **probability transition matrix** (PTM) is a square matrix $N \times N$, whose entries are all nonnegative and whose rows sum to 1. Each PTM describes the probabilities of a current state to change to a future state in a dynamic system. Considering a PTM \mathcal{Q} , $\mathcal{Q}(x'|x)$ represents the probability of going to the future state x' given that the existing state is x , and $x, x' \in \mathcal{Q}$.

Definition A **probability transition matrix of Pitches** \mathcal{Q}^P is a regular PTM where $N = 12$, describing the probabilities related to the 12 pitches of the musical notes, i.e., the 12 semitones ($C, C\sharp$ or $D\flat, D, D\sharp$ or $E\flat, E, F, F\sharp$ or $G\flat, G, G\sharp$ or $A\flat, A, A\sharp$ or $B\flat, B$) from the octave between the $C5$ (the *Do* two octaves above the piano's center *Do*) and $C6$.

Definition A **probability transition matrix of Durations** \mathcal{Q}^D is a regular PTM where $N = 7$, representing the probabilities related to the set of 7 musical notes duration (*Semibreve, Minim, Crotchet, Quaver, Semiquaver, Demisemiquaver, Hemidemisemiquaver*).

The functional dimension from the PMMSG that we intend in this work, is to keep in mind some characteristics from the Markov chains. In general, a human composition melody presents several distinct moments that is a kind of musical nuances and contrasts. A statistical analysis of this music will generate a transition probability matrix where these different moments (nuances, contrasts,...) would be reduced to a global statistic, which could very often mutilate the local characteristics of the composition (moments with own characteristics). In this way, even that a new music produced by those specific transition probability matrices could preserve the statistical global characteristics of the initial music, it is very unlikely also to preserve the lower level local characteristics. Some works [Oliveira 2003] have shown us that the music produced stochastically in the traditional ways, from a unique model of probabilistic state transition, can easily become monotonous (without those distinct moments). The use of PMMSGs in this work aims to by-pass the undesirable characteristics described above. In this way, through the concept of PMMSG, the generated melody is based on a set of melodic segments (partial regions, fragments of the melody) keeping its own local statistical characteristics. The pitches and durations of musical notes are always produced in function of a PMMSG during the process of music stochastic generation. Through the use of n PMMSG in the process

of melodic composition, the global melody will be the result of the interaction among (potentially) n local regions with their own melodic characteristics.

To build the PMMSGs, the user has to choose among those probability distribution of the state transition matrices built in the previous step (see Figure 4). The interface shows an area to select transition probability matrices (for pitches and durations) and also shows the content of the selected matrix (inferior area on Figure 4). In this visualization, the probabilities are represented by a set of shades of gray, which aids the user in the selection task. The shades of gray range from black, indicating a probability of 0%, to white, indicating a probability of 100%. To define a PMMSG, the user must to choose a pair of matrices, combining a transition table of pitches with a transition table of durations.

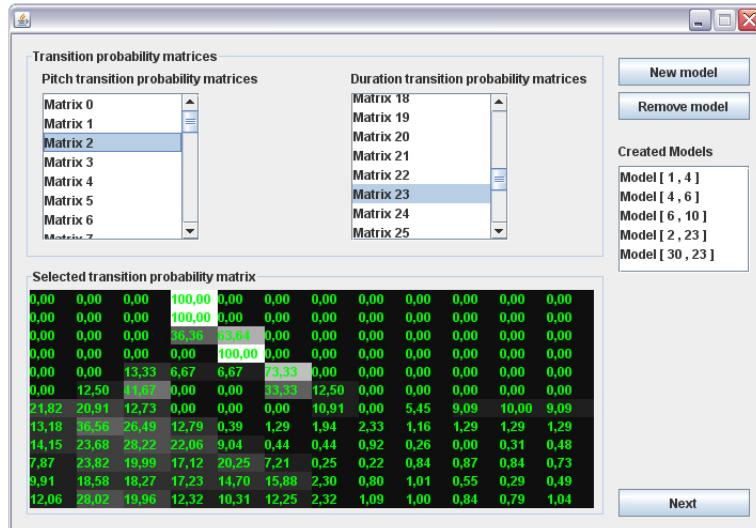


Figure 4. Graphical interface for user PMMSG definitions.

2.3. The Melodic Stochastic Synthesis Layer

In this layer, two Markov chains are generated containing N generated states, one of these chains defines the succession \mathcal{SC} of pitches (\mathcal{SC}^P) and the another defines the succession of durations (\mathcal{SC}^D).

The generation of the two sets ($\mathcal{SC}^P(r)$) and ($\mathcal{SC}^D(r)$) above take the following parameters: a chosen PMMSG from the set of the user-defined PMMSGs, an initial pitch and duration values, a number N of notes to be generated and a MIDI instrument timbre. Thus, an iterative process is triggered in order to generate each one of the N states through the next follow steps:

1. to identify which row in the transition probability matrix that is related to the current select pitch/duration in order to define the next potential pitches/duration;
2. to define sub-intervals in the $]0, 1]$ interval which are proportional to probabilities generated in the previous step. Each one of these subintervals represented a possible pitch/duration to be generated from a current pitch/duration. This step is illustrated in Figure 5;
3. to generate a random number x from the $]0, 1]$ interval. In order to define the next pitch/duration to be generated, one should to check which sub-interval (defined in step 2) the value x belongs to.

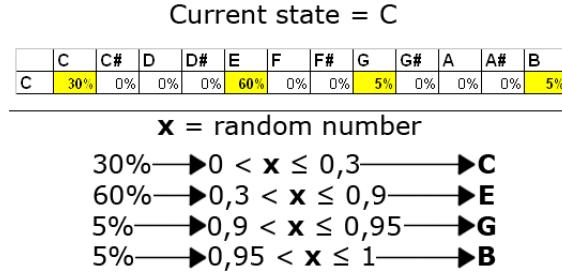


Figure 5. Example of the process of generation of a new pitch, from the current one and a transition probability distribution

At the end of this iterative process, we get a set of N pitches and a set of N durations. When combining the $i - th$ element from the set of resultant pitches to the $i - th$ element from the set of resultant durations, we obtain the $i - th$ musical note of the melody. So, going through the sets linearly (position to position) we obtain the final melody with N musical notes.

During the iterative process several PMMSG are used in order to avoid some monotony in the final melody. In order to perform the selection of the PMMSGs during each iteration, an utility cycle is define to each PMMSG. An utility cycle (uc) to some PMMSG P_g (uc^{P_g}) determines how many times this PMMSG is used into the iterations. At the beginning of each iteration a value to uc^{P_g} is defined from a random number between 1 and a “maximal utility cycle”.

Definition A *maximal utility cycle* (muc) of a PMMSG P_g (muc^{P_g}) determines a maximal limit of iterations:

$$muc^{P_g} = \frac{N}{M} \quad (2)$$

where N is a total number of musical notes to be generated (informed by the user) and M is the total number of PMMSGs defined at the previous layer.

When a PMMSG reaches the end of his utility cycle, a new PMMSG is selected. This selection is carried out by observing which are the last states (pitch and duration) produced. The new PMMSG must have transition probabilities defined for these last states. In this way, we can analyze which PMMSG satisfies these conditions in order to chose randomly one of them to substitute the previous PMMSG.

3. Experiments

The use of a reactive multiagent model aims to produce a perturbation parameter to the system in order to dissociate the resultant melody from any user cognitive bias or aesthetic preferences.

The main goal of the experiment is to demonstrate that the model proposed here is able to generate a global melody with distinct intermediary and local characteristics, through the user interaction with a behavioural-interactive metaphor, rather than making use of the user musical knowledge. In these experiments, each running is unique and irreproducible, due the behavioural-interactive nature of the transition probability matrices generation and the stochastic nature of the process of melodic synthesis.

The multi-agent environment requires some parameters in order to build the structure of the environments and to initialize the composition process. Some examples of these parameters include:

- *Width and height of the environment*, define the dimensions of the environments.
- *Lines and columns*, define how much co-occurrence matrices the environments will be divided.
- *Number of agents* that will interact in each environment.
- *Minimal distance* that each agent have to keep from other agents.
- *Repulsion Multiplier*, represents the strength of repulsion, relative to the intensity with which the agents calculate his evasive behaviour.

Figure 6 illustrates a fragment of the interface showing the multiagent environments and some paths built by the agents during an interaction session.

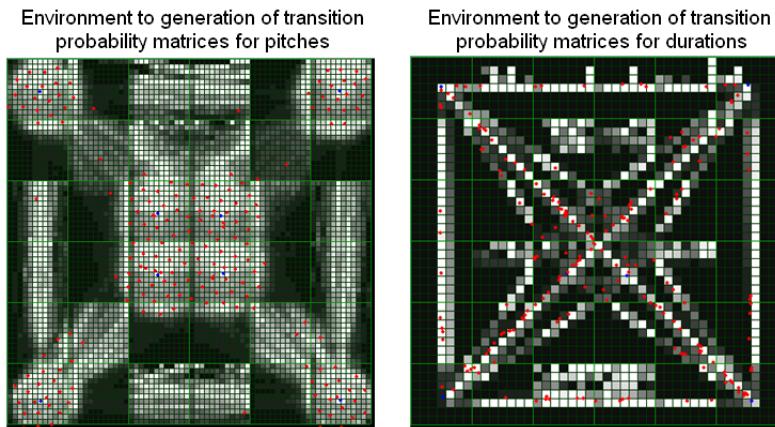


Figure 6. Example of the multiagent environment showing some resultant paths during the agents interactions. At left we have the transition probability matrix for pitches and at right, the transition probability matrix for durations. In each environment, each red dot represents an agent and each blue dot represents a target-point. The distinct shades of gray represent the frequency with which a cell is visited by some agent during the interactions within the environment

The value of some parameters and the number of Target-points defined by the user may produce a great density of Target-points and influence the agent global behaviour. This density will influence straightly the probability distributions in the environments and change “qualitatively” the emerging state transition matrices. Therefore, the stochastic nature of this process do not guarantee the quality of the generated probability transition matrices and do not suffer possible influence from the composer’s aesthetic desires.

After the definition of the 36 transition probability matrices of pitches and 36 transition probability matrices of durations, we have defined some metrics related to melodic regions that present distinctive characteristics. For instance, we were interested in the concentration of high/low pitches and long/short durations. One of the generated PMMSG (Figure 7) has presented a great concentration of transition probabilities for high pitches and short durations, on privileging the generation of high notes of short duration.

Others PMMSGs (Figure 8) from the experiments, has presented a greater concentration of transition probabilities for low pitches and long durations, on privileging the generation of low notes of long duration.

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
Whole note	0,00	0,00	0,00	3,77	5,66	0,00	2,83	4,72	16,98	10,38	8,49	47,17
Half note	0,00	0,00	0,00	1,74	6,38	5,32	7,45	3,55	18,44	16,31	5,32	35,49
Quarter note	0,00	0,00	0,00	0,00	10,24	5,74	4,92	0,00	17,21	8,61	11,89	41,39
Sixteenth note	0,00	0,00	0,00	0,85	8,05	0,42	7,63	5,08	10,17	17,37	5,93	44,50
Thirty-second note	0,00	0,00	0,00	4,02	4,46	7,59	3,12	3,57	12,50	11,16	17,86	35,72
Sixty-fourth note	0,00	0,00	0,00	4,85	3,96	10,13	0,00	4,85	13,22	9,25	18,50	35,24
Whole note	0,00	0,00	2,55	2,13	3,40	10,21	3,83	1,28	16,17	4,68	21,28	34,47
Half note	0,00	0,00	3,98	0,00	4,87	4,42	0,44	4,87	17,70	4,87	25,66	33,19
Quarter note	0,00	0,00	0,45	1,42	3,77	4,72	0,00	4,72	7,08	18,40	24,53	34,91
Sixteenth note	0,00	0,00	0,00	5,33	0,49	0,49	4,37	4,37	6,80	19,42	27,18	31,55
Thirty-second note	0,00	0,00	0,00	0,51	3,06	5,61	5,10	9,18	13,27	35,71	27,56	
Sixty-fourth note	0,00	0,00	0,00	0,00	1,97	8,37	4,93	4,93	17,21	33,00	29,59	

	Whole note	Half note	Quarter note	Eighth note	Sixteenth note	Thirty-second note	Sixty-fourth note
Whole note	0,00	0,00	0,00	0,00	0,00	0,00	100,00
Half note	0,00	0,00	0,00	0,00	0,01	0,03	99,96
Quarter note	0,00	0,00	0,00	0,00	1,56	2,53	95,91
Eighth note	0,00	0,00	0,00	1,29	1,73	1,58	95,40
Sixteenth note	0,00	0,00	0,00	0,00	1,65	0,28	98,07
Thirty-second note	0,00	0,00	0,00	0,00	0,00	1,70	98,30
Sixty-fourth note	0,00	0,86	0,00	0,00	0,00	0,00	99,14

Figure 7. Example (1) of the PMMSG formed by the probability transition matrix of pitches (the top table), and the probability transition matrix of durations (the bottom table).

With these generated PMMSGs we supply the entry to the process of stochastic synthesis in order to obtain a melody in which it can be observed regions with those characteristics described above. The final melody is showed at Figure 9 in a “piano-roll” view (where the musical notes are represented by rectangles which length denote the duration and which height represents the pitch) of the resultant melody where these peculiarities are pointed.

Through this scenario was possible to realize that the choice of the probability tables with which the user builds the PMMSG, from which the final melody will be produced, influences but not define the final melody. So, due to the behavioural-interactive characteristics from the process of probability transition matrices generation and due to the stochastic characteristics in the melodic synthesis process, the final melody is disconnected from cognitive processes related to the composers musical domain.

4. Discussion and future work

In this paper, we have presented a model to automatically generate a melody from the organization of musical elements related to probabilistic spaces which are produced through the interaction of autonomous agents in a process inspired by behavioural-interactive metaphors. In a distinctive way from traditional approaches, in this model the composer does not think in terms of musical parameters to compose music, but in terms of behavioural-interactive phenomena and probability distributions. Also, this behavioural-interactive aspect allied to the multiple parameterizing possibilities is able to make many different melodic compositions arise (see [Behrends 1999]).

With this work we got some insights about the application of a model for musical synthesis under three different levels: *a global level*, defined through the PMMSG distribution along the synthesis process; *an intermediary level* (where melodic segments may present particular aspects), defined by the probabilistic characteristics instantiated into the PMMSGs; and, *a local level* (characteristics of each note), defined in function of the transition from the last generated note. These insights are related to the hierarchy

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
C	61,80	23,22	5,99	3,75	3,00	2,24	0,00	0,00	0,00	0,00	0,00	0,00
C#	47,39	37,28	5,33	2,73	2,98	4,29	0,00	0,00	0,00	0,00	0,00	0,00
D	58,13	31,47	2,93	1,07	0,00	0,00	0,01	2,93	2,93	0,53	0,00	0,00
D#	66,56	14,05	4,01	3,01	3,68	0,67	0,00	0,00	0,00	3,01	3,68	1,33
E	27,46	33,20	7,38	8,61	8,61	3,28	0,00	0,00	0,00	0,00	0,00	2,85
F	2,64	5,26	12,28	19,30	3,51	7,02	14,91	17,54	11,40	6,14	0,00	0,00
F#	0,00	0,00	0,00	0,00	10,82	14,86	0,00	0,00	12,16	17,57	29,73	14,86
G	3,71	0,00	0,00	0,00	0,00	0,00	44,44	18,52	0,00	0,00	0,00	33,33
G#	0,00	0,00	0,00	0,00	0,00	0,00	0,00	43,75	56,25	0,00	0,00	0,00
A	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
A#	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
B	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

	Whole note	Half note	Quarter note	Eighth note	Sixteenth note	Thirty-second note	Sixty-fourth note
Whole note	81,58	17,11	0,40	0,46	0,00	0,11	0,34
Half note	66,94	23,27	0,00	1,22	4,49	4,08	0,00
Quarter note	47,83	32,61	2,17	13,04	4,35	0,00	0,00
Eighth note	30,43	39,13	24,64	4,35	1,45	0,00	0,00
Sixteenth note	25,88	31,76	20,00	8,24	14,12	0,00	0,00
Thirty-second note	21,33	8,00	30,67	32,00	8,00	0,00	0,00
Sixty-fourth note	14,49	1,45	63,77	20,29	0,00	0,00	0,00

Figure 8. Example (2) of the PMMSG formed by the probability transition matrix of pitches (the top table), and the probability transition matrix of durations (the bottom table).

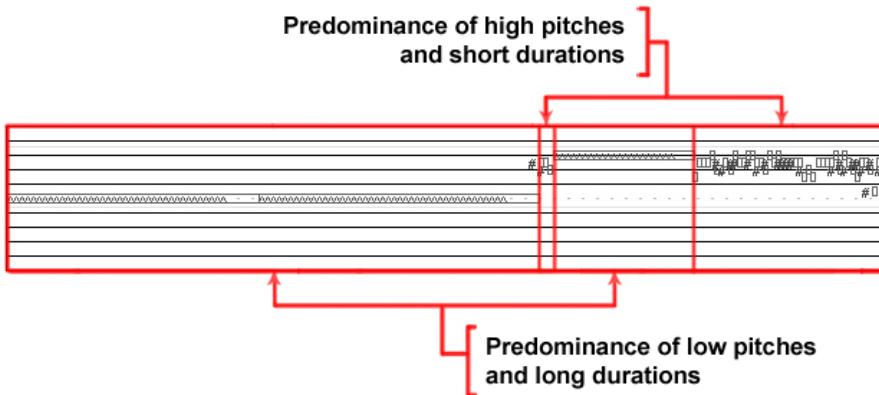


Figure 9. Example of a resultant melody (represented in a “piano roll” view), in which we can identify some melodic regions with particular characteristics.

of the melody global design (comprising the vision of Xenakis about the contemporary music [Xenakis 1992]). In this way, we intend to explore these global/local approaches through new multi-agent interaction models in order to experiment other configurations and organizational multi-agent structures. We can explore in deep our multiple layers of representation and to explore stochastic generation through higher order models of Markov chains in the process of melodic synthesis.

In future work we intend to evaluate the system proposed here as an environment of support for musical creativity, that is, not as an approach for synthesis of musical pieces, but as an environment capable of instigating the composer’s creativity and generate creative material that can be worked by the user, within his/her own creative process. Since the melody generated in this environment is an indirect result of the interaction between the composer and a behavioral-interactive metaphor, we take as hypothesis that the environment promotes the emergence of creative insights, performing a dissociation between the melodic result and the user’s musical knowledge. Thus, in new studies we

also intend to investigate a way to evaluate this hypothesis, checking whether the environment is able to generate interesting melodic structures that would not be conceived by the composer, given his/her musical background.

References

- Behrends, E. (1999). Música: Estrutura e Acaso. *Colóquio/Ciencias*, (24):17–24.
- Brooks, R. A. (1986). A Robust Layered Control System for a Mobile Robot. *IEEE - Journal of Robotics and Automation*, 2(1):14–23.
- Bryan-Kinns, N. (2004). Daisyphone: The Design and Impact of a Novel Environment-for Remote Group Music Improvisation. *ACM Symposium onDesigning Interactive Systems*, pages .135–144.
- Burraston, D. and Edmond, E. (2005). Cellular Automata in Generative Electronic Music and Sonic Art: a Historical and Technical Review. *Digital Creativity*, 16:165–185.
- Cruz, M. A. S. (2001). Técnicas de Computação Sônica Aplicadas ao Design de Software Musical. Master's thesis, State University of Campinas.
- Farbood, M. and Schoner, B. (2001). Analysis and synthesis of palestrina-style counterpoint using markov chains. In *Proceedings of International Computer Music Conference*, Havana, Cuba.
- Fischman, R. (2003). Clouds, pyramids, and diamonds: Applying schrödinger's equation to granular synthesis and compositional structure. *Computer Music Journal*, 27(2):47–69.
- MacGee, D. and Schaffer, J. W. (1997). *Knowledge-Based Programming for Music Research*. A-R Editions, Inc.
- McCormack, J. (1996). Grammar-based music composition. *Complex International*, 3.
- Meyn, S. P. and Tweedie, R. L. (1993). *Markov chains and stochastic stability*. Springer-Verlag.
- Mezencio, R. (2002). Implementação do Método de Campos Potenciais para Navegação de Robôs Móveis Baseada em Computação Reconfigurável. Master's thesis, Departamento de Ciências de computação e estatística - Instituto de Ciências matemáticas e de computação - universidade de São Paulo.
- Miranda, E. R. (2000). The art of rendering sounds from emergent behaviour: Cellular automata granular synthesis. *euromicro*, 02:2350.
- Oliveira, L. F. d. (2003). As contribuições da ciência cognitiva à composição musical. Master's thesis, Faculdade de Filosofia e Ciências da Universidade Estadual Paulista Júlio de Mesquita Filho.
- Papadopoulos, G. and Wiggins, G. (1999). AI methods for algorithmic composition: A survey, a critical view and future prospects. In *Proc. AISB'99 Symp. Musical Creativity*, pages 110–117.
- Xenakis, I. (1992). *Formalized Music: Thought and Mathematics in Music*. Pendragon, revised edition.

Extending the Framework TAO with Norms for Multi-Agent Systems

Emmanuel S. S. Freire¹, Mariela I. Cortés¹, Enyo J. T. Gonçalves², Yrleyjânder S. Lopes³, Leandro L. C. de Souza¹

¹Computer Science Department– Universidade Estadual do Ceará (UECE)
Avenida Paranjana, 1700 – Campus do Itaperi – Fortaleza – CE – Brazil

²Computer Science Department– Universidade Federal do Ceará (UFC), Quixadá, Brazil
 {savio.essf, yrleyjander}@gmail.com, mariela@larces.uece.br, enyo@ufc.br

Abstract. *The increasing complexity of systems poses a challenge to Software Engineering. The existence of Normative Multi-Agent Systems, where the agents' behavior is governed by norms, promotes the need for an ontology capable of defining these related concepts. In this context, we highlight TAO, a conceptual framework for (representing) MAS, used as a foundation to the MAS-ML modeling language. However, the support for representing of the norm concepts is limit. This paper describes the extending the TAO through of the creation of new abstraction norm and its relationships. Additionally, the structure of some abstractions defined in the previous version was changed because of the inclusion of the norm concepts. Through a case study, we demonstrate the use of the proposed extension to represent the elements of a virtual marketplace.*

1. Introduction

An adequate definition of the conceptual framework is critical to understand the business (conceptual modeling) and elaborate a coherent solution (computer modeling) in the context of the development project of complex systems [Dieste et al., 2001]. The focus of the conceptual modeling is to provide the domain understanding related to the described problem. Conceptual models describe the problem found out by the user and describe the way of a software system to solve the problem [Dieste et al., 2001]. In order to produce a solution, computational models can be generated based on conceptual models.

Computational models describe how a software system solves a problem. Because the existence of different concepts proposed to Multi-Agent Systems (MAS) entities the conceptual frameworks are important to establish a base for modeling languages or agent oriented frameworks. The Taming Agents and Objects (TAO) [Silva et al., 2003] is a conceptual framework which provides the basis for software engineering methods based on agents and objects. It is the basis of the modeling language MAS-ML [Silva, Choren and Lucena, 2008], that allows the modeling of all the entities and their properties and relationships defined in TAO. In Addiction, MAS-ML has a support tool called MAS-ML tool [Gonçalves et al., 2011]. This tool allows the modeling of the static diagrams defined in the language.

In order to cope with the heterogeneity, autonomy and diversity of interests among the different members, governance (or law enforcement) systems have been defined. The

governance systems define a set of norms (or laws) that must be followed by the system entities. Norms provide a means for regulating the agents' behavior by describing their permissions, prohibitions and obligations [Figueiredo and Silva, 2010].

Norms are used to regulate the behavior of the agents in MAS by describing the actions that can be performed or states that can be achieved (permissions), actions that must be performed or states that must be achieved (obligations), and actions that cannot be performed or states that cannot be achieved (prohibitions). They represent a way for agents to understand their responsibilities and the responsibilities of the others. Norms are used to cope with the autonomy, different interests and desires of the agents that cohabit the system [López y López, 2003].

In TAO, the modeling of the norms concepts is limited to: (i) allow to regulate the behavior of agents, sub organizations and organizations that inhabit an environment, (ii) define what system resources that will have their access restricted and (iii) specify sanctions that apply if the norms are violated or fulfilled. Thus, there is a need to adapt the concepts of TAO in order to support norm concepts. However, TAO allows the modeling of all the typical entities and their properties that compose a MAS.

This paper proposes the extension of TAO for the representing of norm concepts. The extension is done through the creation of the norm abstraction and its relationships. Additionally, the structure of some abstractions defined in the previous version was changed because of the inclusion of the norm concepts. The representation of these abstractions and their relationships are done through of the templates. The paper is structured as follows: the TAO and the norms for MAS are described in Section 2. Section 3 presents the related work. The extension of the TAO is presented in Section 4. Section 5 presents a case study involving the proposed extension. Finally, conclusions and future work are in Section 6.

2. Background

2.1. Taming Agents and Objects (TAO)

The framework TAO (Taming Agents and Objects) provides an ontology that covers the fundamentals of Software Engineering based on agents and objects. With that is possible to support the development of MAS in large-scale [Silva et al., 2003]. This framework elicits an ontology that connects consolidated abstractions, such as objects and classes, and “emergent” abstractions, such as agents, roles and organizations, which are the foundations for agent and object-based software engineering. TAO presents the definition of each abstraction as a concept of its ontology, and establishes the relationships between them. The Figure 1 shows the abstractions and relationships of TAO. The abstractions of TAO are defined as follows:

- Object: It is a passive or reactive element that has state and behavior and can be related to other elements.
- Agent: It is an autonomous, adaptive and interactive element that has a mental state. Its mental state has the following components: (i) beliefs (everything the agent knows), (ii) goals (future states that the agent wants to achieve), (iii) plans (sequences of actions that achieve a goal) and (iv) actions.

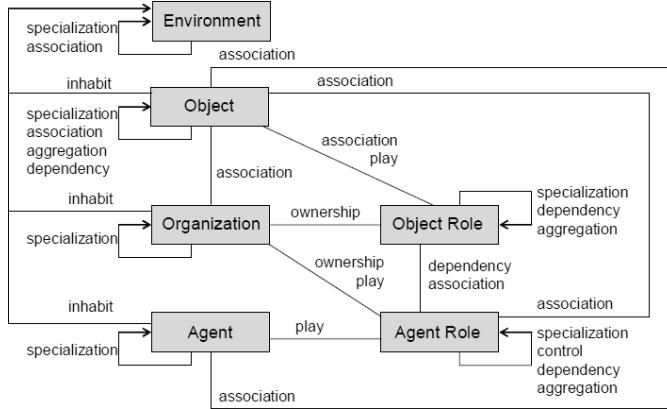


Figure 1. The abstractions and relationships of TAO [Silva et al., 2003].

- Organization: It is an element that groups agents, which play roles and have common goals. An organization hides intra-characteristics, properties and behaviors represented by agents inside it. It may restrict the behavior of their agents and their sub-organizations through the concept of axiom, which define the actions that must be performed.
- Object Role: It is an element that guides and restricts the behavior of an object in the organization. An object role can add information, behavior and relationships the object instance that executes.
- Agent Role: It is an element that guides and restricts the behavior of an agent in the organization. An agent role defines (i) duties that define an action that must be performed by an agent, (ii) rights that define an action that can be performed by an agent and (iii) protocol that defines an interaction between an agent role and other elements.
- Environment: It is an element that is the habitat for agents, objects and organizations. An environment can be heterogeneous, dynamic, open, distributed and unpredictable [Omicini, 2001].

Additionally, Silva et al. (2003) defined the following relationships in TAO: Inhabit, Ownership, Play, Specialization/Inheritance, Control, Dependency, Association and Aggregation/Composition.

2.2. Norms for Multi-Agent Systems

The norms are used to restrict the behavior of agents, organizations and sub-organizations during a period of time, and set sanctions applied if violated or fulfilled [Silva, Braga and Figueiredo, 2010]. Thus, the norms should be associated with an environment, an organization, a sub-organization or an agent role. The following are the main elements that compose the norm based on a survey conducted by [Figueiredo, 2011].

- Deontic concepts: deontic logic refers to the logic of requests, commands, rules, laws, moral principles and judgments [Meyer and Wieringa, 1993]. In MAS, such concepts have been used to describe the constraints for the behavior of agents in the form of obligations (what the agent must execute), permissions (what the agent can execute) and prohibitions (what the agent cannot execute).

- Involved Entities: provided that the norms are always set to restrict the behavior of entities, identification of affected entities is essential. The norm may regulate the behavior of individuals (for example, a particular agent, or an agent, while playing a particular role), or the behavior of a group of individuals (for example, all agents playing a particular role, groups of agents, groups of agents playing roles or all agents in the system).
- Actions: Once a norm is used to restrict the execution of the entities, it is important that the action being regulated is clearly specified. Such actions may be communication, usually represented by sending and receiving a message, or non-communicative actions (such as access and modify a resource, get in an organization, move to another environment, etc.).
- Activation Constraints: The norms have a period in which its restrictions must be fulfilled, but only when they, the norms, are active. Norms may be activated by a constraint or a set of constraints that can be: the execution of actions, specifying time intervals (before, after or in between), the realization of system states or temporal aspects (such as dates) and also the activation / deactivation of other norm and fulfillment / violation of a norm.
- Sanctions: When a norm is violated the entity may suffer a punishment, and when a norm is fulfilled, the involved entity may receive a reward. Rewards and punishments are called for sanctions and should be reported to the specification of the norm.
- Context: the norms are usually defined in a determined context that determines the application area. The norm may, for example, be described in the context of a particular environment and must be filled only by agents in execute in the environment. Similarly, in the context of an organization.

3. Related Work

Some conceptual frameworks and organization models have been proposed for MAS, however they provide limited support to the norm concepts. Our aim is analyze the conceptual frameworks and organization models considering the provided support to the modeling of the typical entities of the MAS along their properties and their relationships. Also, the support given to the modeling of the norm concepts (Section 2.2) will be analyzed. We analyze three conceptual frameworks and two organization models.

The proposed framework by d'Inverno and Luck (2001) defines a hierarchy composed by four layers having entities, objects, agents and autonomous agents. However, it presents the following limitations: (i) in this framework the environment entity has only structural features without transactions, (ii) no dynamic aspect associated with the proposed entities is defined and (iii) it does not provide elements to define correctly the agent norms.

Yu and Schmid (2001) propose a conceptual framework for the definition of role-based agent-oriented MAS. Agents are showed as an entity playing roles within any organization. As weak points we highlight the following aspects: (i) Although agents are defined as an entity playing roles, this conceptual framework does not define the agent properties and relationships between agents and roles, (ii) Although the authors confirm that roles are played in organizations, the proposal does not define the organization properties and the relationship among them and roles, (iii) Neither it defines the environment entity in which

they contain agents and organizations, (iv) it does not allow to describe the elements of norms and (v) it only restricts the agent's behaviour in context of a role.

According to [Dardenne, Lamsweerde and Fickas, 1993], KAoS is a conceptual framework that defines abstractions, such as entities, relationships and agents, as object extensions. An entity is an autonomous object independently of other objects. A relationship is a subordinated object. An agent is an object that has a choice and behavior, and defines beliefs, goals and actions. We can mention some weak points for this framework: (i) KAoS does not consider organizations, roles and environments; (ii) it does not explain in a satisfactory way, the distinction between an entity and an agent; (iii) it does not describe the object features or explains how it is extended by other abstractions; (iv) it does not define any dynamic aspect associated with the described entities and (v) it describes only policies to restrict the access to properties of its abstractions.

The organization model Moise+ [Hübner, Sichman and Olivier, 2002] is based in model Moise [Hannoun, 2002] that presents an organization-centred view considering three forms to represent the organizational restricts (roles, plans and norms). Moise+ has two central notions to represent the organization: (i) organizational specification, that agent group adopts to create an (ii) organizational entity, that its action is designed to achieve a goal. Moise+ allows the description of permission and prohibition norms for roles in context of an organization. However, only non-communicative actions can be restricted by the norm. This model presents the following weak points: (i) it does not allow the environment specification. Thus, the modelling agents moving from one environment to another is not possible; (ii) it does not define the agent properties; (iii) it does not allow the norm specification for agents and environments and (iv) it does not support the definition of sanctions.

The organization model OperA [Dignum, 2004] is a framework that allows the specification of MAS through of the distinction between the characteristics (structure and behaviour) of the organization model and the behaviour of the agents in this model. This framework has the models: organizational, social and interaction for the modelling of organizations and their components. The model describes the organizational structure of the society along with the roles and interactions defined by stakeholders of the organization. OperA allows the description of norms of obligation, permission and prohibition for agents, agent roles and agent groups in context of organization. Additionally, it allows the definitions of restrictions for norm activation. However, this organizational model: (i) it does not allow the modeling the structural aspects of agents and environments, (ii) it does not support the definition of reward, only punishment and (iii) it does not restrict the agent behaviour in context of an environment.

4. Extending Framework TAO

The extension of TAO is based in the inclusion of the elements that compose the norms. These elements are presented in the section 2.2. Thus, in TAO are defined the entity Norm and four relationships: Context, Restrict, SanctionReward and SanctionPunishment.

The creation of the abstraction Norm is necessary because it has a state, behavioral properties and specific relationships. Additionally, the following relationships are defined for allow the association of norms with the entities defined in TAO: (i) Context, that is responsible for identification of the context that the norm will be applied; (ii) Restrict, that identifies the entities that will be restricted by the norm; (iii) SanctionReward, that

identifies the rewards of the determined norm and (iv) SanctionPunishment, that identifies the punishments of the determined norm. The Figure 2 shows the extension of TAO along with the new abstraction Norm and its relationships. In the next Sections, this abstraction and its relationships are described and their representations are done through templates.

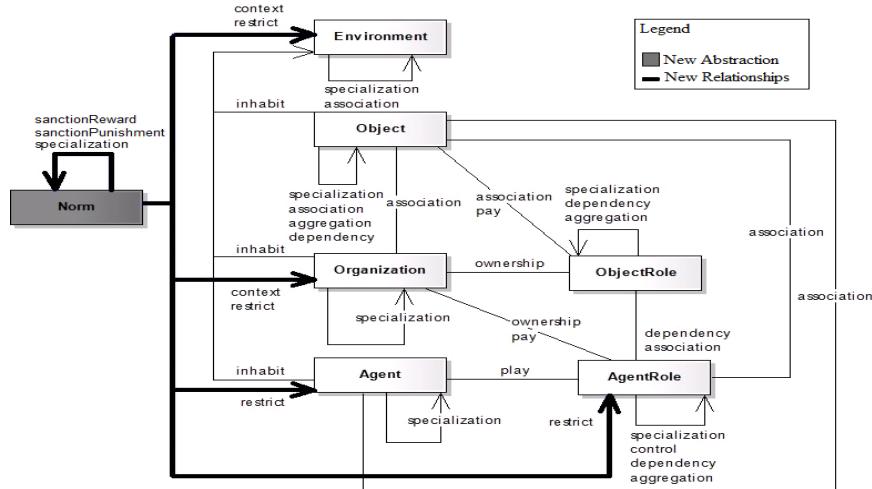


Figure 2. New abstractions and relationships in TAO.

4.1. Norm: A New Abstraction

Norm is an element that restricts the behavior of agents, agent roles, organizations and sub-organizations. A norm restricts the behavior of entities during a period of time and it applies sanctions when violated or fulfilled [Silva, Braga and Figueiredo, 2010].

A norm is an element with state and behavior properties, and its relationships. The state of a norm stores the resource to be restricted. According to [Silva, Braga and Figueiredo, 2010], the resource may be restricted by the norms can be an entity or a property of an entity. The entities that can be governed by norms include: (i) an agent, (ii) an agent role, (iii) an organization, or (iv) an environment. In adding, the restricted properties include: (i) a goal, (ii) a belief, (iii) an attribute, (iv) a method, (v) an action, (vi) a plan, (vii) a protocol, (viii) an association or (ix) a message.

The behavior of a norm is defined based on its characteristics. The characteristics of norms are based in deontic concepts and activation constraints. The deontic concepts define the restriction type of the norm. A norm may be obligation (what the agent must execute), permission (what the agent can execute) and prohibition (what the agent cannot execute). The activation constraints define a period in which the norms are active. A norm may be activated by a constraint or a set of constraints that can be: (i) the execution of actions, (ii) specifying time intervals (before, after or in between), (iii) the realization of system states or temporal aspects (such as dates) and also (iv) the activation / deactivation of other norm and fulfillment / violation of a norm.

The relationships of a norm describe (i) the context that determines the application of norm, (ii) the entity that has its behavior restricted and (iii) the reward or punishment that may be received by the entity that has fulfilled or has violated the norm. These relationships are extensively described in Section 4.2. The norm template presents a norm class. A norm class

defines its state as the resource to be restricted, the behavior of its instances as a set of its properties and a set of relationships that are common to all norm instances.

Norm

Norm_Class Norm_Class_Name

Restriction_Type Deontic_Concept_Name
 Resource <Element_Class_Name.property>
 Activation_Constraint setOf{Constraint_Type Constraint_Type_Name :
 (<Element_Class_Name_First> and/or <Element_Class_Name_Second>) or <date> or
 <Element_Class_Name.property : Operator = (Element_Class_Name.property) or
 value>}Relationships setOf {Relationship_Name}

End Norm Class

4.2. Norms Relationships

This Section presents the relationships between the elements of the conceptual framework with new abstraction Norm. Three new relationships are introduced to indicates others properties of norms.

Let A be a set of agents, $a \in A$, E be a set of environments, $e \in E$, N be a set of norms, $n \in N$. and O be a set of objects, $o \in O$. Let Org be a set of organizations, org , $subOrg \in Org$ and $subOrg$ always represents a sub-organization. Let R be a set of roles, $R = RObj \cup RAg$ where $RObj$ is a set of object roles and RAg is a set of agent roles, $r \in R$, $ro \in RObj$ and $ra \in RAg$. For each one relationship presented below, we present its definition, its classification and the elements that are linked through it.

- Context (C): $C(context, norm)$: $C(e, n)$, $C(org, n)$, $C(subOrg, n)$ – Norm requires the definition of the application context. The context relationship defines that the environment, organization or sub-organization are application context of norm. The behavior of all the elements related to the environment, organization and sub-organization will be governed by the norms.
- Restrict (R): $R(element, norm)$: $R(a, n)$, $R(e, n)$, $R(org, n)$, $R(subOrg, n)$, $R(ra, n)$ – The restrict relationship defines which entity will have their behavior constrained by the norm. If the entity fulfills or violates the norms, the sanction will be applied.
- SanctionReward (SR): $SR(reward, norm)$: $S(n, n)$ – The sanctionReward relationship specifies the reward that can be received by the entity that has fulfilled the norm.
- SanctionPunishment (SP): $SP(punishment, norm)$: $S(n, n)$ – The sanctionPunishment relationship specifies the punishment that can be received by the entity that has violated the norm.

The relationship template is used to define the links between the elements. For each relationship type, the template identifies the elements and its roles in the relationship.

Relationship

Relationship Relationship_Name

CONTEXT : context, norm
 | RESTRICT : element, norm
 | SANCTION_REWARD : reward, norm
 | SANCTION_PUNISHMENT : punishment, norm

End Relationship

4.3. Inclusion of Norm in TAO's Abstraction

In addition to setting the new element Norm and their relationships, adaptations on the already existent abstractions are necessary. According to [Silva et al., 2003], an agent role guides and restricts the behavior of an agent since the goals, beliefs, duties, rights, protocols and commitments associated with the role characterize the agent while playing the role. In the original conception, the concept of right and duty are used to define the actions that can and must be executed by agents. These concepts can be considered semantically equivalent to the deontic concepts of permission and obligation defined in the norms. Thus, the concepts of right and duty have been replaced by norm concepts in the Agent Role template.

<p>Agent_Role</p> <p>Agent_Role_Class Agent_Role_Class_Name</p> <p>Goals setOf{Goal_Name} Beliefs setOf{Belief_Name} Actions setOf{Action_Name}, Protocols setOf{Interaction_Class_Name} U setOf{Rule_Name} Commitments setOf{Action_Name} Relationships setOf{Relationship_Name}</p> <p>End Agent Role Class</p>	<p>Agent_Role</p>
---	--------------------------

An organization defines a set of rules and laws that agents and sub-organizations must obey. The rules and laws are used to characterize the global constraints of the organization [Silva et al., 2003]. Considering that the concept of norm includes rules and laws features, in the new configuration, these concepts were removed and substituted by norm in organization.

<p>Organization</p> <p>Organization_Class Organization_Class_Name</p> <p>Norms setOf{Norm_Name} Actions setOf{Action_Name} Relationships setOf{Relationship_Name}</p> <p>end Organization Class</p>	<p>Organization</p>
---	----------------------------

In TAO, the environment abstraction incorporates the access restrictions associated with their services and resources [Silva et al., 2003]. In other hand, the norm concept considers the definition of access restrictions related with services and resources in the environment. Therefore, the concepts of services and resources have been removed and replaced by the norm concept in the environment.

<p>Environment</p> <p>Environment_Class Environment_Class_Name</p> <p>Norms setOf{Norm_Name} Behavior setOf{Properties} Relationships setOf{Relationship_Name} Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}</p> <p>end Environment Class</p>	<p>Environment</p>
---	---------------------------

5. Case Study

This section presents the modeling of the virtual marketplace [Silva, 2004] using the proposed extension of the TAO addressing the norm concepts related to the entities in MAS.

5.1. Virtual Marketplace

Virtual markets are markets located on the Web useful to the users buy and sell items. Each one consists of a main market where the users can negotiate any type of item. In addition, the market defines two main types: (i) markets for special products that negotiate expensive and high quality items, and (ii) markets for used product that negotiate low quality and low price items. The users can: (i) buy items on the main market, in markets with special products and used products, and (ii) sell its items in markets for used product. In the main market and markets with special products, the users buy the items available on the market. The main market calculates the profits, so the markets with special products and used product must send the information relating to sales to the main market.

5.2. Identification of Virtual MarketPlace Entities

In Virtual Marketplace environment is possible to identify the main organization General Store and its two sub-organizations, Imported Bookstore and Second-hand Bookstore, performing the roles, Market of Special Goods and Market of Used Goods, respectively. The modeling of the environment Virtual Marketplace is shown below.

Virtual_Marketplace

Environment_Class Virtual_Marketplace

Norms setOf{N1}
Behavior setOf{Open, Heterogeneous}
Relationships setOf{Inhabit_VirtualMarketplace_GeneralStore,
Inhabit_VirtualMarketplace_Imported Bookstore,
Inhabit_VirtualMarketplace_user agent, ... }
Events perceived: setOf{Group_Forming}

end Environment Class

The modeling of the organization General Store is shown below.

General_Store

Organization_Class General_Store

Norms setOf{N2, N3, N4, N5}
Relationships setOf{Ownership_GeneralStore_ImportedBookstore,
Ownership_GeneralStore_SecondhandBookstore, ... }

end Organization Class

Moreover, in this system two types of agents are identified: user agent, which can play the buyer role and the store agent, which can play the manager and seller roles. These roles were defined by the main organization, along with the object roles desire and offer. Instances of these roles are played by the instances of the Book class, which inherits from the Item superclass and has two subclasses, SecondHandBook and ImportedBook. The modeling of the agent role Buyer is shown below.

Buyer

Agent_Role_Class Buyer

Goals setOf{ buy_products }, Beliefs setOf{ Offer, Product }
Actions setOf{ payGood, buyGood, finderSeller, reportStatusOffer }
Protocols {FIPA_Protocol}, Commitments {pay_for_Product}
Relationships {Association_Buyer_Seller, Association_Buyer_Manager, ... }

End Agent Role Class

5.3. Definition of Norms to Virtual MarketPlace

For the defined virtual Market in [Figueiredo, 2011], were established the following norms along with their respective modeling:

- N1: All Virtual Market's seller has permission to update stock of good.
- N2: General Store organization's buyers are required to pay for items they bought.
- N3 (Punishment): Buyers that have violated N2 norm are forbidden to buy items.

N1

Norm_Class N1

*Restriction_Type Permission
Resource Good.updateQuantity
Relationships setOf{Context_VirtualMarketPlace_N1, Restrict_Seller_N1}*

End Norm Class

N2

Norm_Class N2

*Restriction_Type Obligation
Resource Buyer.payGood
Activation_Constraint {after : Buyer.buyGood}
Relationships setOf{Context_GeneralStore_N2, Restrict_Buyer_N2,
Sanction_N3_N2}
End Norm Class*

N3

Norm_Class N3

*Restriction_Type Prohibition
Resource Buyer.buyGood
Relationships setOf{Context_GeneralStore_N3, Restrict_Buyer_N3}*

End Norm Class

Additionally, for the proposed definition and modeling in [Silva, Braga and Figueiredo, 2010], the entities Market of Special Goods, General Store, Imported Bookstore, Second-hand Bookstore, Buyer and Seller define constraints on the agent and sub-organization behaviors. This role determines that the agents playing the role have an obligation (duty) to search for buyer agents and the right to accept or reject a proposal submitted by a selling agent. Now, these concepts are modeled by two new norms:

- N4: Buyer agents linked to Buyer roles must look for seller agents.
- N5: Buyer agents linked to Buyer roles can send messages about the proposal situation to a seller agent proposal.

N4

Norm_Class N4

*Restriction_Type Obligation
Resource Buyer.finderSeller
Relationships setOf{Context_GeneralStore_N4, Restrict_Buyer_N4}*

End Norm Class

N5

Norm_Class N5

*Restriction_Type Permission
Resource Buyer.reportStatusOffer
Relationships setOf{Context_GeneralStore_N5, Restrict_Buyer_N5}*

End Norm Class

5. Conclusion and Future Works

The TAO represents a basic ontology defining the core set of abstractions for large-scale MAS. However, the support related to norms is restricted since the deontic concepts are partially supported in TAO. This paper presents the extension of the TAO in order to enable the integration between their entities with the concepts related to norms, along with deontic concepts, entities involved, actions, activation restrictions, sanctions and context.

The extension proposed involves the creation of the Norm abstraction and three relationships: (i) context, (ii) restrict and (iii) sanction. These relationships allow to indicate others properties of norms. The properties of each abstraction and relationships are specified through templates. Additionally, adjustments in the templates of already existent abstractions are proposed in order to include the concepts related to norms. In (i) agent role abstraction, were removed from the concepts related to duty and right, (ii) organization abstraction, was removed the concept related to axiom and (iii) environment abstraction, were removed from the concepts related to the restriction of services and resources. Thus, the extended TAO allows the modeling of all the concepts present in Section 2.2. The case study is centered on the modeling of a Virtual Market through the use of templates, aiming to illustrate the adequacy of the proposed extension.

As future works we consider the following aspects: (i) the formalization of the proposed templates, (ii) the integration of the agents' internal architectures defined in [Russell and Norvig, 2004] and norms elements and (iii) a new TAO extension considering the setting made in (ii).

References

- Dieste, O., Juristo, N., Moreno, A., Pazos, J. (2001). Conceptual Modeling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends. In: Chang, S.K. (eds.): Handbook of Software Engineering and Knowledge Engineering Fundamentals. World Scientific Publishing Co., Vol.1.
- Silva, V.; Garcia, A.; Brandao, A.; Chavez, C.; Lucena, C.; Alencar, P. (2003). Taming Agents and Objects in Software Engineering. In: Garcia, A.; Lucena, C.; Zamboneli, F.; Omicini, A; Castro, J. (Eds.), Software Engineering for Large-Scale Multi-Agent Systems, Springer-Verlag, LNCS 2603, pp. 1-26, 2003, ISBN 978-3-540-08772-4.
- Silva, V.; Choren R.; Lucena, C. (2008). MAS-ML: A Multi-Agent System Modelling Language, In International Journal of Agent-Oriented Software Engineering, Interscience Publishers, vol.2, no.4.
- Gonçalves, E. J. T., Farias, K., Cortés, M. I., Feijó, A. R., Oliveira, F. R. and Silva, V. T. (2011). MAS-ML TOOL - A Modeling Environment for Multi-agent Systems, In: The

International Conference on Enterprise Information Systems (ICEIS 2011), Beijing, China.

Figueiredo, K.; Silva, V. T. (2010). NormML: A Modeling Language to Model Norms. In: 1st Workshop on Autonomous Software Systems. Salvador, Brazil.

López y López, F. (2003). Social Power and Norms: Impact on agent behavior. PhD thesis, Univ. of Southampton, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science.

Omicini, A. (2001). SODA: Societies and Infrastructure in the Analysis and Design of Agent-based Systems. In: Ciancarini, P., Wooldridge, M. (eds.), Agent-Oriented Software Engineering, Springer-Verlag,: 185-194.

Silva, V. T.; Braga, C.; Figueiredo K. (2010). A Modeling Language to Model Norms. The International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2010), 9th. Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, Toronto, Canadá.

Figueiredo, K. (2011). Modeling and Validation Norms in Multi-Agents Systems". Master Thesis. Niterói: UFF, Instituto de Computação.

Meyer, J. J.; Wieringa, R. J. (1993). Deontic Logic in Computer Science: Normative System Specification, Deontic logic in computer science: normative system specification, John Wiley and Sons Ltd. Chichester, UK.

D'Inverno, M.; Luck, M. (2001). Understanding Agent Systems. New York: Springer.

Yu, L.; Schmid, B. (2001). A Conceptual Framework for Agent-Oriented and Role-Based Work on Modeling. In: WAGNER, G.; YU, E. (Eds.). Proceedings of the 1st International Workshop on Agent-Oriented Information Systems.

Dardenne, A.; Lamsweerde, A.; Fickas, S. (1993). Goal-directed Requirements Acquisition. Science of Computer Programming. v.20, p.3-50.

Hannoun, M. (2002). MOISE: un modèle organisationnel pour les systèmes multi-agents. Tese (Thèse(Doctorat)) – École Nationale Supérieure des Mines de Saint-Etienne.

Hübner, J. F., Sichman, J. S. and Olivier, B. (2002). A model for the structural, functional and deontic specification of organizations in multiagent systems. In: SBIA '02 Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence, Springer-Verlag London, UK.

Dignum, V. (2004). A model for organizational interaction: based on agents, founded in logic. PhD dissertation, Universiteit Utrecht, SIKS dissertation series 2004-1.

Silva, V. T. (2004). Uma linguagem de modelagem para sistemas multi-agentes baseada em um framework conceitual para agentes e objetos, Doctoral Thesis. Rio de Janeiro: PUC, Departamento de Informática.

Russell, S. and Norvig, P. (2003). Artificial Intelligence: A Modern Approach, 2nd Ed., Upper Saddle River, NJ: Prentice Hall, ISBN 0-13-790395-2.

Reinforcement learning for route choice in an abstract traffic scenario

Anderson Rocha Tavares¹, Ana Lucia Cetertich Bazzan¹

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{artavares,bazzan}@inf.ufrgs.br

Abstract. *Traffic movement in a commuting scenario is a phenomena that results from individual and uncoordinated route choice by drivers. Every driver wishes to achieve reasonable travel times from his origin to his destination and, from a global point of view, it is desirable that the load gets distributed proportionally to the roads capacity on the network. This work presents a reinforcement learning algorithm for route choice which relies solely on drivers experience to guide their decisions. Experimental results demonstrate that reasonable travel times can be achieved and vehicles distribute themselves over the road network avoiding congestion. The proposed algorithm makes use of no co-ordinated learning mechanism, making this work a case of use of independent learners concept.*

1. Introduction

The subject of traffic and mobility presents challenging issues to authorities, traffic engineers and researchers. To deal with the increasing demand, techniques and methods to optimize the existing road traffic network are attractive since they do not include expensive and environmental-impacting changes on infrastructure.

In a commuting scenario, it is reasonable to assume that drivers choose their routes independently and, most of the time, uninformed about real-time road traffic condition, thus relying on their own experience. Daily commuters usually have an expectation on the time needed to arrive on their destinations and, if a driver reaches its destination within expectation, his travel time can be considered reasonable. From a global point of view, it is desired that vehicles gets distributed on the road network proportionally to the capacity of each road. There is a challenge on finding a good trade-off between global (road usage) and individual (travel time) performance on traffic scenarios.

Traffic assignment deals with route choice between origin-destination pairs in transportation networks. In this work, traffic assignment will be modeled as a reinforcement learning problem. This approach uses no communication among drivers and makes no unrealistic assumptions such as the drivers having complete knowledge on real-time road traffic condition. In reinforcement learning problems, agents make decisions using only their own experience which is gained through interaction with the environment.

The scenario studied in this work abstracts some real-world characteristics such as vehicle movement along the roads, allowing us to focus on the main subject which is the choice of one route among the several available for each driver.

The remainder of this document is organized as follows: Section 2 presents basic traffic engineering, single and multiagent reinforcement learning concepts that will be used throughout this paper. Section 3 presents and discusses related work done in this field. Section 4 presents the reinforcement learning for route choice algorithm whose results are discussed in Section 5. Finally, Section 6 concludes the paper and presents opportunities for further study.

2. Concepts

2.1. Commuting and traffic flow

In traffic engineering, a road network can be modeled as a set of nodes, representing the intersections, and links among these nodes, representing the roads. The weight of a link represents a form of cost associated with the link. For instance, the cost can be the travel time, fuel spent or distance.

A subset of the nodes contains the origins of the road network, where drivers start their trips, and another subset contains the destinations, where drivers finish their trips. Usually, in a commuting scenario, a driver has to travel from an origin to a destination (an OD pair) on the same time of the day. A driver's trip consists on a set of links, forming a route between his OD pair among the available routes.

Traffic flow is defined by the number of entities that use a network link in a given period of time. Capacity is understood as the number of traffic units that a link support in a given instant of time. Load is understood as the demand generated on a link at a given moment. When demand reaches the link's maximum capacity, the congestion is formed.

2.2. Reinforcement Learning

Reinforcement learning (RL) deals with the problem of making an agent learn a behavior by interaction with the environment. The agent perceives the environment state, chooses an available action on that state and then receive a reinforcement signal from the environment. This signal is related to the new state reached by the agent. The agent's goal is to increase the long-run sum of the reinforcement signals received [Kaelbling et al. 1996].

Usually, a reinforcement learning problem is modeled as a Markov Decision Process (MDP) which consists on a discrete set of environment states (S), a discrete set of agent actions (A), a state transition function ($T : S \times A \rightarrow \Pi(S)$), where $\Pi(S)$ is a probability distribution over S) and a reward function ($R : S \times A \rightarrow \mathbb{R}$). $T(s, a, s')$ means the probability to go from state s to s' after performing action a in s .

The optimal value of a state, $V^*(s)$, is the expected infinite discounted sum of rewards that the agent gains by starting at state s and following the optimal policy. A policy (π) maps the current environment state $s \in S$ to an action $a \in A$ to be performed by the agent. The optimal policy (π^*) represents the mapping from states to actions which maximizes the future reward.

In order to converge to the optimal policy, value iteration and policy iteration algorithms can be used. In policy iteration, the value function is estimated (policy evaluation), then this estimation is used to change the policy, until the policy converge to optimal. To accelerate this process, value iteration is used: it truncates the policy evaluation phase after one step, thus changing the policy at each step.

Both value and policy iteration algorithms are model-based which means that they use prior estimations of R and T (which are the environment model). Model-free systems don't rely on R and T estimates in order to converge to the optimal policies. Q-learning [Watkins and Dayan 1992] is such an algorithm.

2.3. Multiagent Reinforcement Learning

A multiagent system can be understood as group of agents that interact with each other besides perceiving and acting in the environment they are situated. The behavior of these agents can be designed a priori. In some scenarios, this is a difficult task or this pre-programmed behavior is undesired, thus making the adoption of learning (or adapting) agents a feasible alternative [Buşoniu et al. 2008].

For the single-agent reinforcement learning task, well understood, consistent algorithms with good convergence exists. When it comes to multiagent systems, several challenges arise. Each agent must adapt itself to the environment and to the other agents behaviors. This adaptation demands other agents to adapt themselves, changing their behaviors, thus demanding the first to adapt again. This nonstationarity turns invalid the convergence properties of single-agent RL algorithms.

Single-agent RL tasks modeled as a MDP already have scalability issues on realistic problem sizes and it gets worse for multi agent reinforcement learning (MARL). For this reason, some MARL tasks are tackled by making each agent learn without considering other agents adaptation, knowing that convergence is not guaranteed. It is remarked by [Littman 1994] that training adaptive agents in this way is not mathematically justified and it's prone to reaching a local maximum where agents quickly stop learning. Even so, some researchers achieved amazing results with this approach.

3. Related work

In traffic engineering, the traditional method for route assignment works as follows: each driver has his route determined at the initial phase of traffic simulation, through econometric formalisms which find an equilibrium. This process is not self-adaptive, thus do not allows the use of learning methods. Nevertheless, this process does not consider individual decision-making, thus do not allows the modelling of heterogeneity.

Application of intelligent agent architectures to route choice is present on a number of publications. Agent-based approaches support dealing with dynamic environments. Next, some works based on this approach are reviewed.

Several of these works use abstract scenarios, most of the times inspired by congestion or minority games. On these scenarios, agents have to decide between two routes and receive a reward based on the occupancy of the chosen route. This process is repeated and there is a learning or adaptation mechanism which guides the next choice based on previous rewards.

With this process, a Pareto-efficient distribution or the Wardrop's equilibrium [Wardrop 1952] may be reached. In this condition, no agent can reduce its costs by switching routes without rising costs for other agents.

Two-route scenarios are studied in [Bazzan et al. 2000, Chmura and Pitz 2007, Klügl and Bazzan 2004]. The former analyses the effect of different strategies on mi-

nosity game for binary route choice. The second uses a reinforcement learning scheme to reproduce human decision-making in a corresponding experimental study. The third includes a forecast phase for letting agents know the decision of the others and then let them change their original decision or not. Each one of these works assessed relevant aspects of agents decision-making process, even though only binary route choice scenarios were studied. The interest on the present work is to evaluate a route choice algorithm in a complex scenario, with several available routes.

This kind of complex scenario was investigated by [Bazzan and Klügl 2008]. On their work, Bazzan and Klügl assessed the effect of real time information on drivers' route replanning, including studies with adaptive traffic lights. The authors assume that real time information on road occupation of the entire network is known by the drivers. This assumption was needed for assessing the effects of re-routing, but it is unrealistic.

More recently, the minority game algorithm was modified for use in a complex scenario with several available routes [Galib and Moser 2011]. Using the proposed algorithm, drivers achieve reasonable (within expectation) travel times and distribute themselves over the road network in a way that few roads get overused. The modified minority game algorithm uses historic usage data of all roads to choose the next one on the route. Having historical information of the roads used by the driver is a reasonable assumption, but having historic information of all roads on the network is unrealistic. The algorithm proposed on the present work will be compared with the modified minority game.

4. Algorithm and scenario

4.1. Reinforcement learning for route choice

In this study, one agent will consider the others as part of the environment. Thus, other agents learning and changing their behavior will be understood as a change of environment dynamics. For this fact, agents follow the concept of independent learners [Claus and Boutilier 1998]. Prior to the present work, independent learning agents were studied in cooperative repeated games [Claus and Boutilier 1998, Tan 1993, Sen et al. 1994]. In all these works, empirical policy convergence was achieved, though [Claus and Boutilier 1998] demonstrates that Q-learning is not as robust as it is in single-agent settings and studies whether the found equilibrium is optimal.

The present study is an application of the independent learners concept in a competitive multi-agent system as agents compete for a resource (the road network). Decisions on this route choice scenario are sequential, making this a more complex scenario, expanding the horizon achieved on prior works.

The MDP for this problem is modeled as follows: there are no states on the scenario. The set of actions comprises the selection of the outbound links from the nodes of the network. Not every link will be available for the agents to choose, as it depends on which node of the network it is. The reward function is presented on Section 4.3. There is no need of a transition function as there are no states on this problem.

4.2. The algorithm

The proposed algorithm is based on Q-learning. For a description of Q-learning, the reader may refer to [Watkins and Dayan 1992].

4.2.1. Initialization

At the beginning of execution, OD pairs are randomly distributed among drivers. Then, each driver calculate the shortest route P^* for his OD pair. As there are no different weights on network links, the shortest route is the one with less links between origin and destination. Then, for each driver, the expected travel time is calculated by the following equation:

$$et_{P^*} = \sum_{a \in P^*} t_a(ex) \quad (1)$$

Where et_{P^*} is the estimated travel time on route P^* , t_a is the travel time function. It is defined in Eq. (2) and is applied for each link a of the shortest route P^* . The term ex is the expected number of drivers on the same route. At the beginning of the simulation ex is given by the number of drivers with the same OD pair of the driver calculating its expected travel time plus a random number in the range [-50:50]. This means that each driver has an estimation of the number of commuters who live in the same neighborhood and uses the same roads to reach their workplaces.

4.2.2. Execution

Each iteration (or episode) of this reinforcement learning for route choice algorithm follows the steps shown in Figure 1.

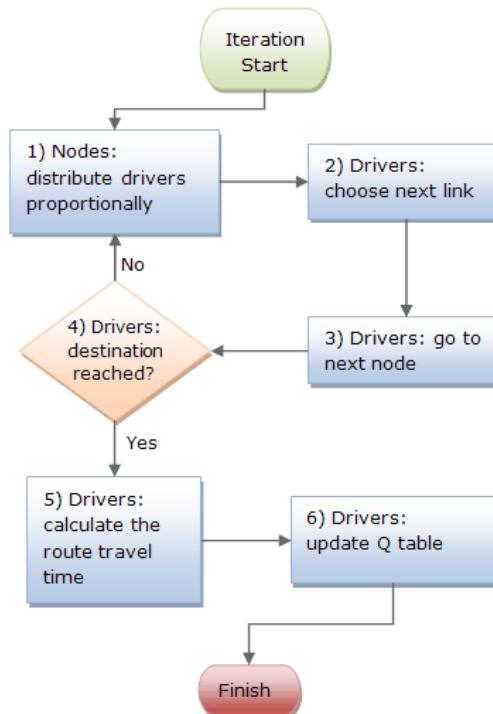


Figure 1. RL for route choice flowchart

At step 1, drivers are hypothetically distributed among the outbound links of the nodes containing vehicles. This hypothetical distribution is proportional to the capacity of

each link and will be compared to the actual distribution achieved by the drivers individual choices. At step 2, drivers choose an outbound link to traverse according to the ϵ -greedy strategy: choose an arbitrary link with probability ϵ , or choose the best link according to the Q-table with probability $1 - \epsilon$. At step 3, drivers reach the destination of the chosen link. If this node is the driver's final destination (step 4), the trip ends, otherwise steps 1 to 4 are repeated. At step 5, each driver i calculate the travel time of the chosen route P . Drivers traversing the link a of the road network experience the travel time (t_a) given by the following function [Ortúzar and Willumsen 2001]:

$$t_a(x) = f_a[1 + \alpha \left(\frac{x}{c_a}\right)^\beta] \quad (2)$$

Where x is the number of drivers on the link a . The constant f_a is the free-flow travel time, a parameter of link a which has capacity c_a . Then, for each driver, the travel time of the chosen route P is given by the following formula:

$$at_P = \sum_{a \in P} t_a(x) \quad (3)$$

Where at_P is the actual travel time experienced by the driver on route P , t_a is the travel time experienced on link a (calculated via Eq. (2)) with x being the number of drivers on the link a .

At step 6, drivers update the values on Q-tables with the entries corresponding to the links in route P according to the Q-learning update formula:

$$Q(a) = (1 - \alpha)Q(a) + \alpha(R + \gamma \max(Q(a'))) \quad (4)$$

Where $Q(a)$ is the Q-value for action 'choosing the link a ', α is the learning factor, γ is the discount factor and R is the reward received by the driver for traversing link a . The reward function is discussed in Section 4.3.

4.3. Reward function

The reward function was designed with the goal of fostering drivers to assume different behaviors. By traversing a road, a driver receive a reward R , defined as:

$$R = s(R_{tt}) + (1 - s)(R_{occ}) \quad (5)$$

Where R_{tt} is the reward component regarding the travel time, R_{occ} is the reward component regarding road occupation and 's' can be understood as a selfishness coefficient. Ranging from 0 to 1, it determines whether the driver will prioritize his own welfare, trying to minimize his travel time (higher values of s) or the social welfare, tending to choose roads with less occupation (lower values of s).

The component regarding the travel time (R_{tt}) is given by:

$$R_{tt} = -t_a(x) \times W \quad (6)$$

In this equation, $t_a(x)$ is the travel time function given by Eq. (2) with x being the number of drivers on link a . W is the route weight given by:

$$W = \frac{at_P}{et_{P^*}}$$

Where at_P is the actual travel time experienced by the driver (Eq. (3)) and et_{P^*} is the expected travel time for the driver (Eq. (1)). The purpose of the route weight is to make the driver avoid apparently good links which result in bad options in subsequent decisions.

In this component, the reward decreases as travel time increases. This is to foster drivers to choose routes that will result in smaller travel times. By using this component (higher values of s on Eq. (5)), it is expected that drivers try to minimize individual travel times, making selfish choices. That is, if a congested link or route leads to the final destination faster than an uncongested alternative, they are expected to choose the congested option.

The reward component regarding road occupation (R_{occ}) is given by:

$$R_{occ} = \left(\frac{c_a}{x_a} \right) - 1 \quad (7)$$

Where c_a is the capacity of link a and x_a is the number of vehicles on this link. This reward component will become positive if the driver chooses an uncongested link ($c_a > x_a$) and will become negative if the number of vehicles on the link becomes higher than its capacity. By using this component (smaller values of s on (5)), it is expected that drivers make choices taking into account the social welfare, that is, to avoid congestion and alleviate the traffic flow on the network, even if it results in higher individual travel times.

4.4. Evaluation Metrics

In order to assess the Q-learning based route choice algorithm in terms of drivers' travel time and distribution of vehicles over the road network, the following metrics will be used:

- Experiment average travel time (xATT): is the average of the mean travel time for all drivers along the 50 episodes. For this metric, lower values means better performance.
- Actual and expected travel time difference (AEDIFF): This metric is calculated for each OD pair. In one episode, it is given by the difference between the average of actual travel time and the average of expected travel time for all drivers on the same OD pair. For the experiment, the values obtained are averaged over the number of episodes. It is desirable that this metric reaches negative values. This means that actual travel times are lower than drivers' expectations.
- Actual and proportional distribution difference (APDIFF): this metric is relative to the roads. For one road, it is given as the absolute value of the difference between the actual and the proportional number of vehicles in it. For the road network, it is given as the sum of the values obtained for all roads. The closer this metric gets to zero the better, because this means that the distribution of vehicles in the network is close to the hypothetic proportional distribution.

4.5. Studied scenario

In this work, the abstract road network used in the experiments is the same used by [Galib and Moser 2011], for comparison purposes. It consists on 10 nodes and 24 links, as depicted in Figure 2. All nodes have 3 outbound links, except nodes 8, 9 and 10 which have 2, 1 and 0 outbound links, respectively. Nodes 1, 2 and 3 are the possible origins and nodes 8, 9 and 10 are the possible destinations, resulting in nine possible OD pairs. The network links have the same weight, representing no differences on their lengths.

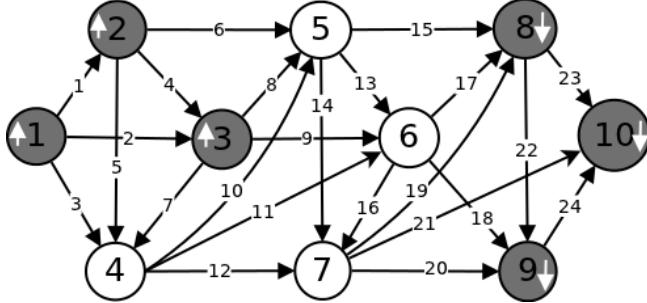


Figure 2. Road network, the same used by [Galib and Moser 2011]. Labels on links are identification numbers. Nodes with upward arrows are the origins and downward arrows represent the destinations

Each one of the 1001 drivers have a fixed OD pair through all the experiment which simulate a commuting scenario, like in a city with drivers living in different neighborhoods trying to reach their workplaces. Each iteration of the experiment represents this happening at the same time of the day.

5. Results and discussion

5.1. Reward function and drivers' behaviors

In these experiments, the objective is to test the effect of the selfishness coefficient (s in Eq. (5)) on drivers' behavior. Parameters' values are: $\alpha = 0.5$, $\gamma = 0.4$, $\epsilon = 0.1$ for the Q-learning based route choice algorithm. There are 1001 drivers on the road network and roads' capacities are randomly assigned in the range [130:250] at the beginning of the simulation. For the travel time, (Eq. (2)), $\alpha = 1$, $\beta = 2$. This means that, as the number of drivers on a road increases, the travel time increases quadratically. The constant f on Eq. (2) is set as 5 minutes for all links.

Figure 3 shows APDIFF metric increasing as the selfishness coefficient increases. This means that, when drivers strive to avoid congested roads (lower values of s), their distribution over the road network gets closer to the proportional. Figure 4(b) shows the road network usage for $s = 0$. It is possible to see that actual and proportional number of vehicles are very close for most roads. The biggest exception is road 19, that connects node 7 and 8. A detailed investigation showed that this happens because, in order to try to avoid congested roads, several drivers who must finish their trips at node 8 end up reaching node 7 and then they become out of alternatives but traverse road 19 to node 8. This does not happen when $s = 1$ as shown in Figure 4(a).

In Figure 4(a), despite the differences between actual and proportional distribution, only few roads were congested and no road got severely congested, as the maximum usage did not get higher than 120% of the capacity.

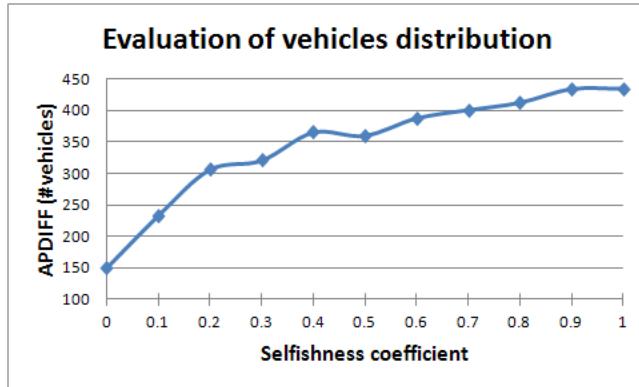


Figure 3. Quality of vehicles distribution on the network versus s

Figure 5 shows drivers' travel time decreasing as the selfishness coefficient increases. Travel time becomes higher than the expected only when drivers totally disregard travel times and strive to find uncongested roads ($s = 0$).

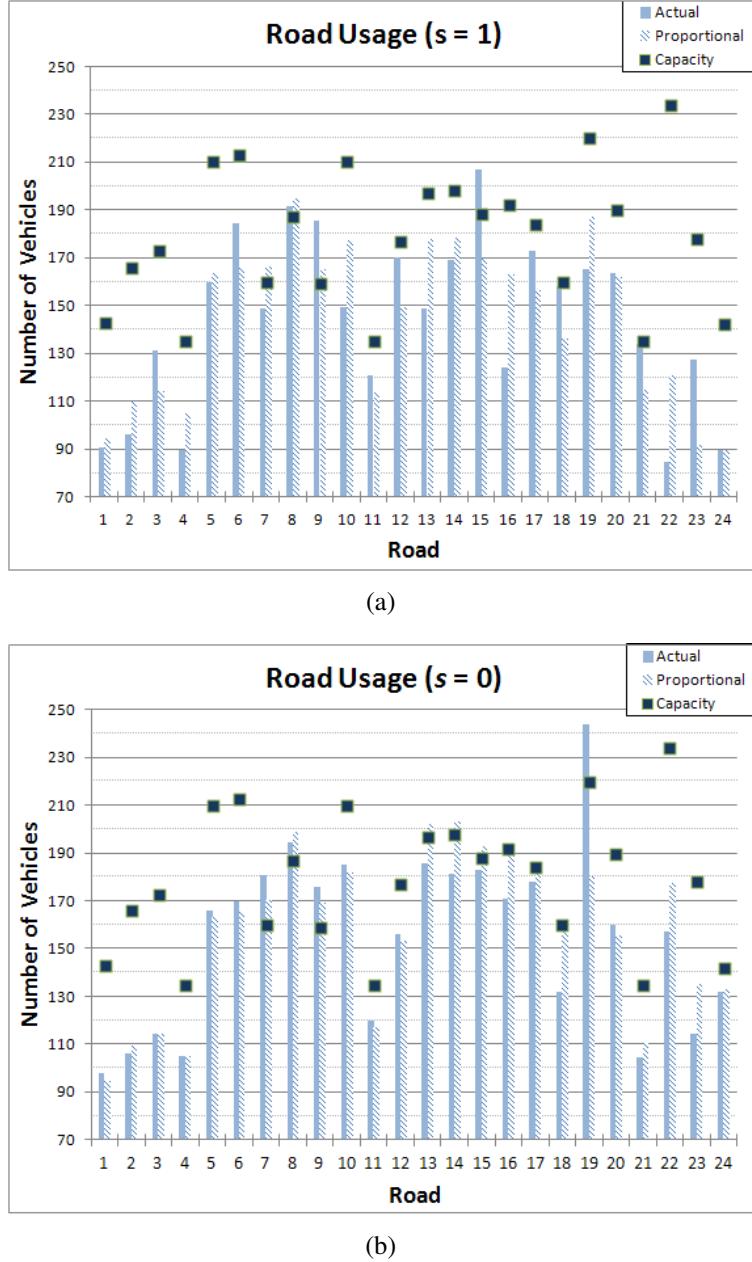
A more interesting investigation can be done on Figure 6, where the AEDIFF metric is plotted. This plot shows that travel time is more affected by s on two moments: when drivers start considering minimizing travel time (from 0 to 0.1) and when they stop considering road occupation (from 0.9 to 1). At this second moment, drivers from the OD pair 2-8 start having reasonable travel times. On average, drivers from OD pairs 1-9, 1-10, 3-8 and 3-9 do not experience reasonable travel times. In the worst case, travel time is 11.58 minutes above expectation (OD pair 2-8 and $s = 0$).

Comparing both road usage and travel times, we can see that, by adjusting the selfishness coefficient, it is possible to achieve either a more distributed road usage or smaller travel times. For these experiments, it turned out that using $s = 1$ is a good choice, as travel times are smaller, and a good distribution of vehicles on the road network can still be reached. By comparing with the extreme opposite ($s = 0$), travel times weren't reasonable anymore and even more roads were congested. This shows that, although drivers don't "care" about social welfare when $s = 1$, they still avoid congested roads as this improves their travel times. This is why drivers distribute themselves over the network, even when the goal is not to achieve a perfectly proportional distribution.

5.2. Comparison with evolutionary game theory

The objective of the following experiment is to compare the reinforcement learning for route choice algorithm with the one based on the Minority Game, proposed in [Galib and Moser 2011]. Comparison is made in terms of travel times per OD pair and distribution of drivers along the roads.

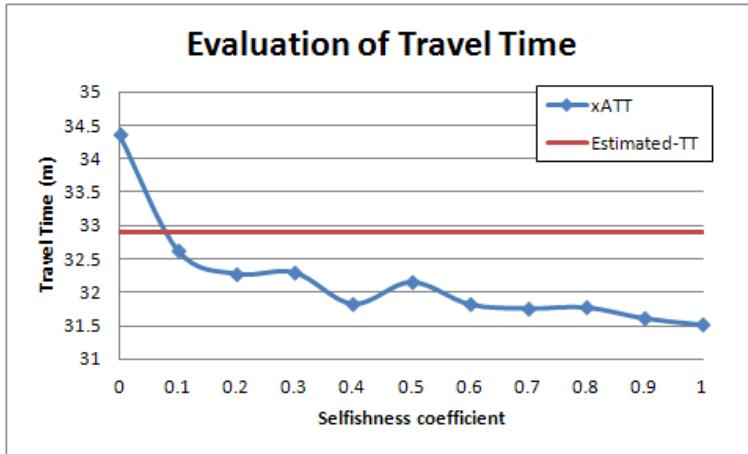
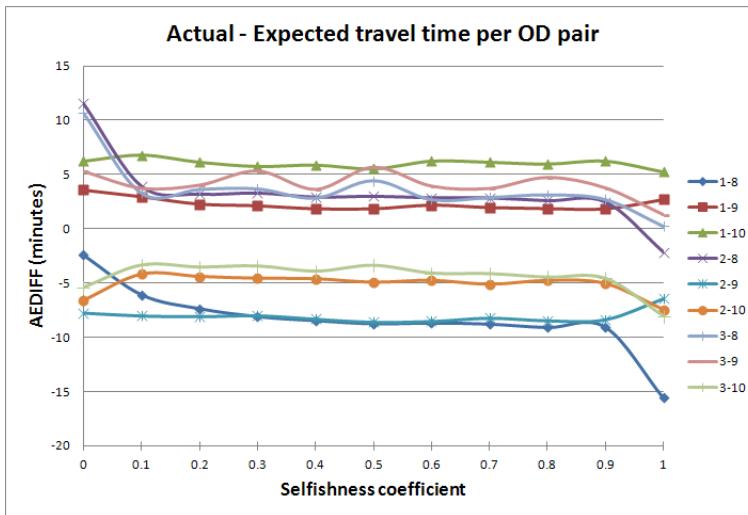
Figure 7(a) shows the travel time obtained by both algorithms. Figure 7(b) compares both algorithms regarding roads usage. The values shown are the average over 50 iterations. The algorithms have similar performances, although drivers using the minority game based algorithm achieve lower travel times. The highest travel time difference is 3.41 minutes for OD pair 3-10.

**Figure 4. Roads usage with $s = 1$ (a) and $s = 0$ (b)**

6. Conclusions and future work

In this work we have presented a new algorithm for route choice in an abstract traffic scenario using reinforcement learning. Our approach is helpful for either individual and global point of view, as drivers achieve reasonable travel times, on average, and only few roads are overloaded.

The proposed approach is based on realistic assumptions as the algorithm only relies on drivers own experience about the road network, dismissing the use of real-time information and historic data of roads. This makes our algorithm an attractive alternative to be used on existing navigation systems, as no new technologies are required.

**Figure 5. Evaluation of xATT metric****Figure 6. Evaluation of AEDIFF metric**

This work is a successful application of the independent learners concept on a complex, competitive scenario. Agents learned how to choose routes to their destinations even considering other agents as part of the environment.

Further investigation can be conducted to assess how the algorithm performs in heterogeneous scenarios, that is, when there are drivers who use other decision processes or algorithms. Future works can also attempt to assess how good it would be for agents when they consider other agents on the environment, that is, how good it would be to learn joint actions in this competitive environment.

7. Acknowledgments

Authors would like to thank Mr. Syed Galib for clarifying questions on the minority game for route choice algorithm [Galib and Moser 2011] and for providing data for comparison. The authors also would like to thank the anonymous reviewers for their suggestions of paper improvements. Both authors are partially supported by CNPq and FAPERGS.

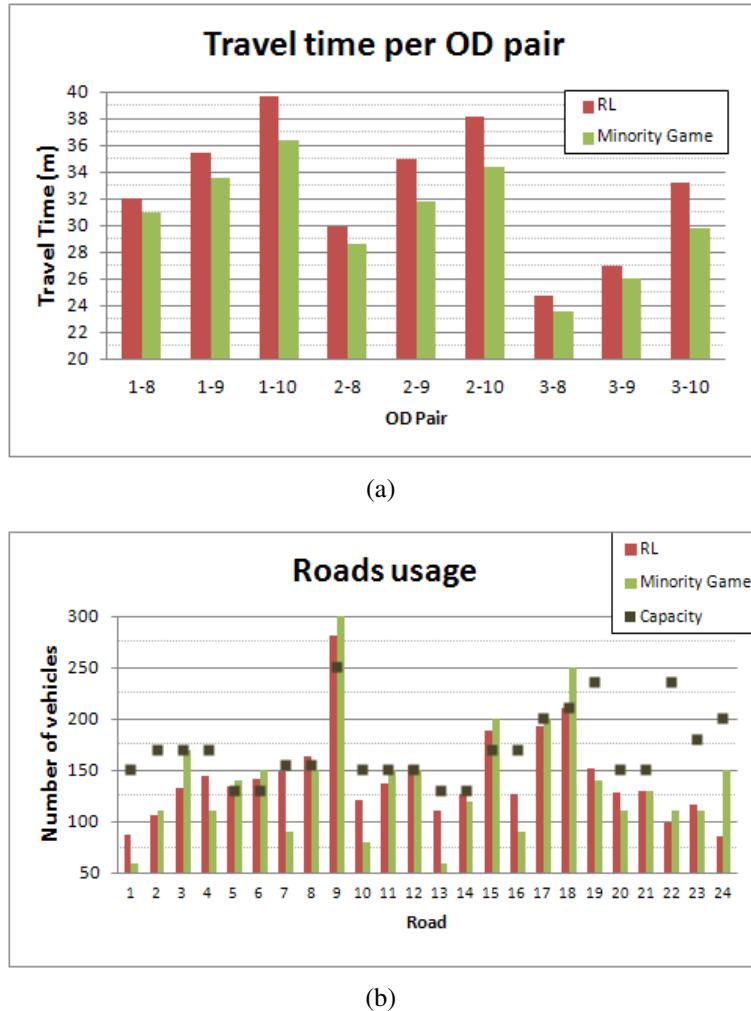


Figure 7. Comparison of algorithms regarding travel time (a) and road usage (b)

References

- Bazzan, A. L. C., Bordini, R. H., Andriotti, G. K., Viccari, R., and Wahle, J. (2000). Wayward agents in a commuting scenario (personalities in the minority game). In *Proc. of the Int. Conf. on Multi-Agent Systems (ICMAS)*, pages 55–62. IEEE Computer Science.
- Bazzan, A. L. C. and Klügl, F. (2008). Re-routing agents in an abstract traffic scenario. In Zaverucha, G. and da Costa, A. L., editors, *Advances in artificial intelligence*, number 5249 in Lecture Notes in Artificial Intelligence, pages 63–72, Berlin. Springer-Verlag.
- Buşoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172.
- Chmura, T. and Pitz, T. (2007). An extended reinforcement algorithm for estimation of human behavior in congestion games. *Journal of Artificial Societies and Social Simulation*, 10(2).
- Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative

- multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752.
- Galib, S. M. and Moser, I. (2011). Road traffic optimisation using an evolutionary game. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, GECCO ’11, pages 519–526, New York, NY, USA. ACM.
- Kaelbling, L. P., Littman, M., and Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.
- Klügl, F. and Bazzan, A. L. C. (2004). Simulated route decision behaviour: Simple heuristics and adaptation. In Selten, R. and Schreckenberg, M., editors, *Human Behaviour and Traffic Networks*, pages 285–304. Springer.
- Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, ML, pages 157–163, New Brunswick, NJ. Morgan Kaufmann.
- Ortúzar, J. and Willumsen, L. G. (2001). *Modelling Transport*. John Wiley & Sons, 3rd edition.
- Sen, S., Sekaran, M., and Hale, J. (1994). Learning to coordinate without sharing information. In *Proceedings of the National Conference on Artificial Intelligence*, pages 426–426. JOHN WILEY & SONS LTD.
- Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning (ICML 1993)*, pages 330–337. Morgan Kaufmann.
- Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. In *Proceedings of the Institute of Civil Engineers*, volume 2, pages 325–378.
- Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

A Model for Opinion Ranking

Allan D. S. Lima, Jaime S. Sichman

Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica da USP
Avenida Prof. Luciano Gualberto, travessa 3 nº 158 - CEP - 05508-970 - São Paulo – SP
adsl@usp.br, jaime.sichman@poli.usp.br

Abstract. Day after day we have been noticing the growth in the number of opinions about any subject over the Internet. Consequently it's becoming hard for people to know what others think about a subject. In order to proper handle this problem, a research field called opinion mining was created. One of its objectives is to classify the relevance of an opinion. Focusing in this point, we present in this paper a model to rank opinions. Our model is based in the composition of a ranking function by combining many concepts. Some of them are related to social aspects of the user who is searching for opinions. Also we discuss how to find the best combination of our parameters and how our model is different for projects where opinions were ranked before.

1. Introduction

When someone wants to find opinions about a subject (product, theme, person, etc.) how can he have access to third party opinions? Before the Internet, when someone wanted to discover opinions about a subject he could ask for people in his social circle like his parents, relatives, friends, etc. Furthermore, other sources like television, magazines and newspapers were frequently consulted as well. However nowadays, it's possible to search over the Internet for opinions about basically every subject. By using tools like websites, forums, blogs and search engines it is possible to everyone to express or to access information, as well as opinions. Ironically, as a side effect, this also brought an information overload, making hard to find the opinions that are really relevant among all those available.

For example, how someone interested in buying an electronic game called Just Dance can find opinions about this product? One of the most common ways is to access a search engine (like Google) and search for “just dance review”. This query will give the user more than 60 million pages and probably each of them has many opinions in its content! Hence, the problem we will discuss in this paper can be described as: *among the data available at the Internet, how to find the opinions that are relevant to someone?* We believe that the best way to solve this problem is by proposing a model to rank opinions capable of give personalized results, i.e., respecting the fact that every user can have its own way to define an opinion as relevant. In other words, our main goal is to build a model that can generate a personalized opinion list depending on who is searching the opinion. In order to deal with this problem, we are going to introduce a unique way to use Opinion Mining [Liu 2012] together with other concepts like Social Search [Morris et al 2010a], a concept that even Google have been applying on its document ranking functions [Google 2012]. In order to achieve our objective, we have been devel-

oping an opinion ranking model composed by: (1) a set of concepts that can evaluate the relevance of an opinion; (2) an abstract (domain independent) ranking function for opinions able of use those concepts as parameters; (3) a way to discover how to proper combine those concepts in a scoring function.

So far we have defined seven concepts as the parameters of our ranking function: (1) Information Retrieval [Manning et al at 2008]; (2) Memes [Dawkins 2006]; (3) author's experience; (4) how a user sees an author as relevant or not; (5) how all users see an author as relevant or not; (6) the similarities between a user and an author; (7) the distance between an user and an author in a relationship network (like a graph). Also it's important to highlight that parameters like (4) allow our model to give different scores to the same opinion depending of the user is searching for it. We made this possible because the same opinion can relevant for a user while it can irrelevant to other.

This paper describes the current state of our work. We start from an introduction to opinion mining and other basic concepts necessary to proper understand our model, in section 2. Then we review the previous work related to our objective in section 3. In section 4 we describe our model, while in section 5 show how our model handles opinions differently for others projects. Finally, in section 6 we made our final considerations and present our plans for future work.

2. Basic Concepts

2.1. Opinion Mining

Textual information can be categorized basically in two types: (1) facts and (2) opinions. Facts are objective declarations about entities or events while opinions are declarations reflecting subjective fillings of people or perceptions about entities or events [Liu 2012]. Since the popularization of the Internet, especially regarding to tools like blogs, the way people express and have access to opinions have been changing. For example, when someone wants to make his mind about a polemic subject, he can use not only the traditional ways (friends, relatives, television, newspapers or magazines) but also the whole information present at the Internet [Liu 2010] as well.

Information Retrieval [Manning et al 2008] is the computer science research field responsible of mining information from the Internet. However, research in this field in most cases does not handle facts and opinions differently, both are considered as information. Consequently those techniques aren't able to proper help users demanding for opinions. This is one of the reasons why researchers created a new research field called Opinion Mining that can be defined as follow [Liu 2012]:

Definition 1 – Opinion Mining: *Given a set of documents D containing opinions (or sentiments) about a subject, opinion mining is considered to be a set of tasks of extraction and identification of features and components from the subject addressed by each document d in D, and then defining if these comments are positive, negative or neutral.*

As an example to differentiate opinion mining from information retrieval, let us analyze an hypothetical piece information about a camera: “it has a good image resolution, but its zoom isn't as good as other cameras already in the market”. It's possible to notice the existence of two different opinions in the text, the first one is positive and the second one is negative. Moreover the first opinion is directly related to the image resolu-

tion. Meanwhile the second refers to the zoom feature by a comparison with other cameras. While information retrieval will handle that sentence as a text document piece, opinion mining will try to identify and eventually index opinion aspects presents on it. This simple example was responsible to intuitively introduce some of the opinion mining targets. But there are many other research objectives related to this research field, among them we can highlight [Liu 2010]:

- **Problem formalization:** mathematically handle problems related to opinion mining through formalizations of its definitions, limitations, constraints or objectives;
- **Sentiment and subjectivity analysis:** given a document, identify if it has opinions, and if so, classify them as positive, negative or neutral;
- **Characteristic oriented sentiment analysis:** discover the features that were referred in each opinion. For example, for a computer it's possible to have opinions about features like its *processor* or its *hard drive*.
- **Sentiment analysis in comparative sentences:** usually an author expresses his opinion about a theme by comparisons or metaphors. For example, “the notebook battery lasts as long as every ordinary battery”;
- **Search engine and information retrieval:** systems where the user is able to make queries about a theme and receives as result a list of opinions or documents (containing opinions);
- **Spam detection and opinion utility:** When it comes to opinion mining spam is also a problem; for example, it's possible that an author paid by the product manufacturer could make product reviews. Also the utility of an opinion is a score that represents its importance degree so it can be used to create ranking documents functions.

2.2. Information Retrieval

Information Retrieval (IR) is seen like the task of search for data (usually documents) in non-structured documents (usually texts) in order to satisfy a specific information need in a big collection of data (usually stored in computers) [Manning et al 2008]. In order to properly understand this definition it's necessary to understand the difference among structured and non-structured data before. A dataset is structured when its information is well delimited by fields that have a semantic value that makes it easy to parse by an algorithm. For example, table lines in a relational database are considered as structured data. However, a dataset is considered as non-structured when there is no distinction among its different parts, which makes it hard to parse by an algorithm. For example, a text in an Internet blog is considered to be a set of non-structured data.

Because of the big collections of data, the results of a query in an IR system can have millions of documents, what exceeds the human capacity of reading them in a short time interval. Hence, IR provides many strategies in order to score documents and consequently generate an ordered list of them. One of the most simple and popular ways to rank documents uses the frequency of terms in the documents. The *tf-idf* score function is composed by two concepts the (1) *tf*: term frequency and the (2) *idf*: inverse document frequency [Manning et al 2008]. The term frequency is the number of occurrences of a term in a given document. But the inverse document frequency is computed according to Equation 1. The *idf* score is logarithm of the number of documents (N) divided by the number occurrences of that a term in all documents in a collection (*df*).

$$idf_t = \log \frac{N}{df_t}$$

Equation 1. Inverse document frequency.

The final value of the *tf-idf* score is archived by the multiplication of the both (*tf* x *idf*). But since this is value of a single term, we need to sum the *tf-idf* score for each term in a query in order to compute the score of a document.

Ranks generated by score functions like *tf-idf* have to be evaluated, this why in information retrieval there are many metrics to measure how an information retrieval system is good. In this section we are going to cover four of them [Manning et al 2008]: (1) precision; (2) coverage; (3) F Measure; (4) Means Average Precision;

Precision, Equation 2, is computed by the division of the number of relevant items retrieved by the number of retrieved items. In other words precision represents the conditional probability of a relevant item, given that it was retrieved. An important variation of this metric is the K-Precision (also known as P@K) where only the first K retrieved items are considered.

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{items retrieved})} = P(\text{relevant}|\text{retrieved})$$

Equation 2. Precision.

Coverage, Equation 3, is computed by the division between the number of relevant items retrieved and the number of items retrieved. In other words, it's the conditional probability that an item be retrieved given that it is relevant.

$$\text{Coverage} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$

Equation 3. Coverage.

In order to combine both (precision and coverage) there is a metric called F Measure. It unites both by the harmonic mean, Equation 4, where α represent the weight of the combination. For example, if $\alpha > 0.5$ then it means that precision is more important than coverage.

$$F\text{Measure} = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{C}}$$

Equation 4. F Measure.

Another important metric used to evaluate information retrieval systems is the Means Average Precision. Given that during the evaluation of a system many query are submitted to it, then this concept is computed by the average precision for the first K documents in each query during the tests. Hence, given $Q = \{q_0, \dots, q_j, \dots, q_n\}$ as the set of all test queries; given $D = \{d_1, \dots, d_{mj}\}$ as the relevant documents for a query q_j ; and given R_{jk} as the set retrieved documents at the first k documents; Equation 5 shows how to compute the Means Average Precision.

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk})$$

Equation 5. MPA.

2.3. Social Search

Social Search is been expected to be the future of search engines, researchers [Evans and Chi 2010], information retrieval systems [Horowitz and Kamvar 2010; ChaCha 2011], and companies like Google [Sherrets 2008] and Microsoft [Morris et al 2010a; Morris et al 2010b] have been studying about how social components can help a user to fulfill its information needs. Experiments developed by Microsoft researchers showed that people usually have more confidence in answers from its friends. Also they have discovered that people frequently believe that social networks are better than search engines when answering subjective questions like products recommendations [Morris et al. 2010a]. Moreover, the experience of information searching could be improved by the integration of search engines and social resources [Morris et al 2010b].

It's possible to understand the meaning of social search by two complementary definitions:

Definition 2 – Social Search by Morris et al: *The term social search means the process of finding information on-line by the help of social resources, like, for example, asking to friends or unknown people at the Internet for answers [Morris et al 2010b].*

Definition 3 – Social Search by Evans & Chi: *Social Search is term used to describe search acts that make use of social interactions with other individuals. These interactions may be explicit or implicit, local or remote, synchronous or asynchronous [Evans & Chi 2010].*

Both definitions complement themselves. The definition 2 talks about the interactivity of social search while definition 3 discusses interaction types in social search. Following the concepts of social search projects like Aardvark [Horowitz & Kamvar 2010] and ChaCha [ChaCha 2012] were developed. The first one can search the users with the best knowledge to answer a question, while the second one offers a search service for answers made by specialists (its employees).

2.3. Memes

The term “meme” was introduced in 1967 by Richard Dawkins [Dawkins 2006]. It represents a cultural transmission unit that propagates itself in a society through the replication by imitation of things like ideas, clothes, styles, quotes, etc. For example, when a teacher reads something good, it's expected that he is going to spreads it among its students or colleagues. If what it has spread is something interesting eventually people will start to spread it too. Every time a meme reaches someone new, a popular meme (or fertile) can make that person a vehicle to propagate itself like a gene among individuals of the same specimen. Another analogy frequently used to explain what is a meme is comparing it to a virus. For example, a virus uses a host cells to propagate itself, like a meme uses the mind of its host to propagate itself.

While many memes can spread themselves many times, others can't do that. According to Dawkins [Dawkins 2006] it happens because of 3 memes characteristics (or dimensions): (1) fidelity; (2) fecundity; and (3) longevity. Fidelity is defined as the ability that a meme has to not change during the time even when it happens to be replicated many times. Fecundity is rate were a meme is replicated, the faster it is replicated, the more are the chances that it has to capture a large audience. Finally, longevity refers to the amount of time in which a meme can continue infecting minds.

3 Related Work

Despite the lack of projects directly related to our objective it's still possible to find in the literature some works using opinions in the development ranking functions. One of the first works in this field was published in 2006 [Mishne 2006]. It describes a ranking function associated with aspects (parameters) like sentiment analysis, spam detection, and link based authority estimation in order to rank blog posts. The function presented in this project is composed by a linear combination of those aspects. Moreover they have applied MAP (Mean Average Precision) and R-Precision metrics as the methodology to evaluate the weight of proposed parameters in the ranking function. At the end of the experiments they have concluded that the link based authority estimation value wasn't relevant improve to the ranking function while the other aspects worked well to improve results.

Another interesting project [Zhang 2009] proposes a ranking function to documents composed by two main values: opinion relevance and topic relevance. Both values are combined by a quadratic combination because the researchers believed that this type of combination is superior to linear interpolations. Based on that, they have developed a raking function and conducted experiments with TREC Blog06 and Blog07 corpus [Access to Web/Blog Research Collections 2012]. In those experiments they have applied as evaluation metrics MAP, R-Precision, and precision at top 10 (P@10) results in order to evaluate different variations of their ranking function. As a result they have concluded that the quadratic combination together with logarithm normalization was the best way to implement their ranking functions.

A third project [Huang and Croft 2009] presents a probabilistic ranking function for documents. The ranking function is based on the number of occurrence of opinion words. It's used together with others technics like query expansion based on a synonymous dictionary to build a score function for documents. The values were combined by a linear interpolation. Then they performed experiments in the TREC Blog06 and COEAE08 (a document dataset for the Chinese language compiled by the Chinese Academy of Sciences) databases to discover the weight of each component in their scoring function that maximizes the results. The main metric used is this project was MAP, however other metrics like R-Precision and P@10 where computed and presented in the paper.

The last project of our state of art review [Attardi and Simi 2006] presents a ranking function that uses as one of its parameters a relevance estimation function for opinions in documents. To achieve that, subjective words considered to be carrying an opinion bias were tagged in the documents. Then they build a system that are able receive as one of its inputs the subject that the user is searching for opinions. To interpret

those queries their system finds for words carrying an opinion bias near to the words representing the subject. In order to evaluate the results they made experiments in the Blog06 using the P@5 (precision at top 5) as the main evaluation metric.

We believe that to develop a good ranking function exclusively for opinions (not for documents like blog posts) we can apply similar evaluation metric but use also concepts from research fields other than only information retrieval, like social search. How they are going to do that shall be described in the next section.

4. Model for Opinion Ranking

In order to create an abstract model to rank opinions we need to formally define the most important concept of our model, in particular what we consider an opinion.

Definition 4 – An Opinion is a 7-tuple $\langle \text{author}, \text{date}, \text{site}, \text{sentence}, \text{subject}, \text{orientation}, \text{feature} \rangle$ where *author* is a string containing the name of the opinion's author, *date* is a date where the opinion was published, *site* is the URL (Universal Resource Locator) of the site where the opinion was published, *sentence* is a string with the sentence containing the opinion, *subject* is a string representing the opinion's subject, *orientation* is a string with the opinion's orientation ("positive", "negative" or "neutral"), *feature* is a string representing the feature which the opinion is about.

Consider the following sentence about the electronic game Just Dance 2 that was published on 14 November 2010 by Alex St-Amour at vgchartz.com: "*The audio for this game is remarkably well done.*" [St-Amour 2009]. In our model this opinion would be represented by the following 7-tuple: <"Alex St-Amour", "14 November 2010", <http://gamrreview.vgchartz.com/review/45671/just-dance-2/>, "The audio for this game is remarkably well done.", "Just Dance 2" "positive", "audio">

We have been developing a model to rank opinions composed by a set of scores: (1) Information Retrieval Score; (2) Memetic Score; (3) Author's Experience; (4) Author's Image Score; (5) Author's Reputation Score; (5) Similarity Score; (6) Network Distance Score. Those concepts are described in the following sub-sections.

4.1. Parameters of our Model

The **Information Retrieval Score** is based on the basics IR concepts: term frequency and document frequency (tf-idf). Given a feature f and an opinion o it can be computed according to Equation 6.

$$\text{IR Score}(f, o) = \sum_{h \in \text{Hyponymies}(f)} \text{tf}(h, \text{Sentence}(o)).\text{idf}(h)$$

Equation 6. IR Score.

Equation 6 sums the tf-idf score of every hyponym of the feature f returned by the function *Hyponymies*. We consider a hyponym as a non-empty set of words semantically linked to the feature. For example, in the context of electronic games, if $f = \text{"multiplayer"}$ then $\text{Hyponymies}(f) = \{\text{"local multiplayer"}, \text{"on-line multiplayer"}, \text{"co-op multiplayer"}, \text{"cooperative multiplayer"}, \text{"competitive multiplayer"}\}$.

The **Memetic Score** is a social value computed by an estimation of the opinion's longevity. For example, if a positive opinion about a product feature has been said for many years, it could mean that positive opinions about that feature are relevant. Given an orientation *or*, Equation 7 shows how to estimate this concept. In Equation 7 $P(or|t)$ represents conditional probability of a opinion with the orientation *or* given the time *t*. The concept of time in the equation can represent a month, a semester, a year, etc., it depends on the context where the model shall be applied. For example, if we take time a year and its initial value as 2000, Longevity will use $P(or|t=2000)$, $P(or|t=2001)$, $P(or|2002)$, etc. So, given an orientation, for each year the equation will compute and sum the variation of the probability that an opinion with that orientation occurs, compared to the previous year.

$$\text{Longevity}(or) = \sum_{t=0}^{n-1} -|P(or|t) - P(or|t+1)|$$

Equation 7. Longevity of an orientation.

The **Author's Experience** is a score reflecting the number of times that an author has expressed his opinion about a subject. Given an author *a*, a feature *f* and *Opinions* a function that returns a set of opinions containing only the opinions expressed by *a*, Equation 8 shows how to compute the author's experience.

$$\text{Experience}(a, f) = \sum_{op \in \text{Opinions}(a)} \begin{cases} 1, & f = \text{feature}(op) \\ 0, & f \neq \text{feature}(op) \end{cases}$$

Equation 8 – Author's Experience.

The **Author's Image Score** is a social value representing how a user sees an opinion's author as relevant or not. It can be estimated by user feedback while an opinion search engine is running or by asking users about the authors before they start to use the system. Formally, given an author *a* and a user *u*, we will define the Image of *a* as a real number *v*, shown in Equation 9.

$$\text{Image}(a, u) = v, \text{where } v \in R \text{ and } 0 \ll v \ll 1$$

Equation 9. Author's Image.

The **Author's Reputation Score** is a social value defined by how users at the Internet evaluate an author relevant or not. It's basically the sum of all reputations of an author. Formally, given an author *a* and a set of all users *U*, Equation 10 shows how to compute the Image of *a*.

$$\text{Reputation}(a) = \frac{\sum_{u \in U} \text{Image}(a, u)}{|U|}$$

Equation 10. Author's reputation.

The **Similarity Score** is social concept defined by an estimation of preferences and interests that an author and a user have in common. For example, books, electronic games, hobbies, etc. Given an author a , a user u and $Interests$, a function that returns the set of preferences and interests that a person has interest, Equation 11 shows how to estimate the Similarity between a and u .

$$\text{Similarity}(a, u) = \frac{\text{Interests}(a) \cap \text{Interests}(u)}{\text{Interests}(u)}$$

Equation 11. Similarity score.

Finally, in a relationship graph what we call the **Network Distance Score** is the smallest number of nodes between the opinion's author and the user which made a query, it's also a social metric. Formally, given a user u , an author a and g , a relationship graph where users and authors are the nodes, Equation 12 defines the distance among a and u .

$$\begin{aligned} \text{Distance}(u, a, g) = n, \text{ where } n \in N \text{ and} \\ n \text{ is the number of edges in a shortest path} \\ \text{connecting } u \text{ and } a \text{ in } g \end{aligned}$$

Equation 12. Network Distance Score.

4.2. An Opinion Raking Function

Formally it's possible to define our ranking function as follows:

Definition 5 – Opinion Ranking Function:

Inputs – An opinion; information retrieval score; memetic score; author's reputation score; author's image score; similarity between author and user/reader; network distance score; author's experience score.

Output – A score s (where s belongs to the set of real numbers) representing the opinion's score.

How those parameters will be combined is something that we expect to be different according to the domain where the model shall be applied. This is why we propose a way to discover a good combination of them through the use of an Artificial Intelligence technique called Genetic Algorithms [Russell and Norvig 2009]. This technique mimics the process of natural evolution by the simulation of concepts like: selection, inheritance, mutation, and crossover. We believe that it fits very well to solve our problem because genetic algorithms make it possible to test many combinations of our parameters in the heuristic way, without testing most of the possible permutations. So from an initial random set (initial population) of possible combinations for our parameters, applying selection, inheritance, mutation, and crossover it's possible to evolve them. This evolution should be made to achieve the best performance on the evaluation metrics usually applied to evaluate ranking functions (MAP, R-Precision, and precision at top 10).

Given the Longevity and Similarity functions (parameters of our model) and a , b , c , d (weights), in this first version of our model, we propose the following ways to combine our parameters:

- Linear combination: $a.\text{Longevity} + b.\text{Similarity}$
- Multiplication: $a.\text{Longevity}.\text{Similarity}$
- Logarithm combination: $a.\log_c(\text{Longevity}) + b.\log_d(\text{Similarity})$
- Logarithm multiplication: $a.\log_c(\text{Longevity}).\log_d(\text{Similarity})$

5. Model Discussion

In order to compare our approach to rank opinions with the ones applied by the other projects presented in the section 3, let us discuss how our function to rank opinions compares to them.

The first relevant difference is about the objectives: we want to rank opinions while the models that we could find in our literature review want to rank documents (sets of sentences that can eventually contain opinions). Thereby, let's take as example the Just Dance 2 review published at vgchartz.com [St-Amour 2010] once again. Our model will consider every single opinion as entity and then generates a score for each one of them. However, other models will handle the whole review as a single entity and generate a score for it based on the number of opinions on it. The objective of our model is to rank opinions, so all parameters that we consider are focused on the estimation of the opinions relevance. On the other hand, other projects while aiming to rank documents can usually apply other concepts like topic relevance. So the score computed by our model is exclusively linked to opinion's relevance, but the scores of other projects use opinion's relevance estimation as a parameter of a document relevance estimation function.

Regarding the concepts applied to rank an opinion, our model uses: tf-idf score, author's image, author's reputation, author's experience, orientation's longevity, similarity among user and author and finally network distance between user and author. While other models have score functions to evaluate an opinion based on the number words carrying opinion bias. This contrast is a consequence of our model's objective, which leads its ranking function to explore many ways to rank an opinion. For example, given the sentence that we have analyzed before "*The audio for this game is remarkably well done.*" [St-Amour 2010], for each sentence like these our model shall compute the score of all its parameters and then combine them. But the other models will just judge its relevance by the fact that it's an opinion or not since what they are ranking are documents.

Another important difference between our model and the previous models in the literature is the possibility to customize results. Social concepts [Morris et al 2010a] present in our project were developed to proper handle different kinds of users. When it comes to opinions it's expected that the same opinion can be relevant for a user A while it can be irrelevant for a user B. In the models that we could find it was impossible to provide different results for the same query. However in our model some of its concepts are inspired in Social Search (author's image, similarity and network distance) where the concepts defined allow customs results according to the user profile. For example, a sentence written by an author A can be on the top 10 best scored opinions for a user U1, because A has a good image for U1. However, for a user U2 the same sentence can't achieve the top 10 because U2 thinks that A doesn't have a good image.

There is a model that combines the scores of their secondary ranking functions in a final score by linear combination [Mishne 2006] while another uses a quadratic combination [Zhang & Ye 2009]. However our model only suggests possible ways to combine its concepts. We have made this decision because we believe that the best way to handle this issue is to combine changes according to the domain where the model shall be applied. Hence we also discuss how a way to find out a good combination of our parameters through Genetic Algorithms.

The different projects found in the literature developed their experiments using corpus composed mainly be blog posts, without the selection of a specific domain (for example, blogs about books). Since we believe that final result of a score function can vary according to its domain, we chose a specific type of opinions to evaluate: opinions about electronic games. Furthermore there are many other fields like books or music where opinions can be retrieved and ranked. However, when it comes to the evaluation metrics the related work have an important contribution to our work because they showed to us that it's possible to use information retrieval techniques in order to evaluate opinion mining systems, so we are able to apply traditional metrics like MAP to measure the results of our model.

6. Conclusion

Queries for opinions in traditional search engines are handled the same way as queries for facts. In order to fill this gap, developers have created a research area called Opinion Mining, whose goal is to handle the subset of queries designed to retrieve opinions. In this field, this report presented the current state of a model for scores of opinions by relevance, by combining several concepts that weren't present in models that we could find in our state of art review.

So far we have discussed mainly how to rank opinions. However, we have been applying our model to create a function in a specific domain: reviews for electronic games. But there are some tasks that we need to do in order to have a function ready to rank that kind of opinions. Among them it's possible to name: (1) to develop a hyponym thesaurus of electronic game features; (2) to train an opinion classifier; (3) to collect the relevance data from users; (4) to apply genetic algorithms for the best combination of the parameters; (5) and finally to make the proper adjustments in the model based on the experience acquired in the development of a ranking function for opinions about games. Currently, we have developed a crawler based on a collection of game reviews and we are also starting to develop an opinion classifier based on the work of [Li et al 2011], in parallel with the compilation of a hyponym thesaurus for games features..

References

- "Access to Web/Blog Research Collections" (2012),
http://ir.dcs.gla.ac.uk/test_collections/access_to_data.html. March
- Attardi, G., Simi , M. (2006) "Blog Mining through Opinionated Words". Fifteenth Text RE-trieval Conference.
- ChaCha. (2012) "Questions & Answers | ChaCha", <http://www.chacha.com/>. March.

- Dawkins, R. (2006) *The Selfish Gene*. Oxford University Press, USA, This 30th anniversary edition.
- Evans, B. M., Chi, E. H. (2010) “An elaborated model of social search”. *Information Processing & Management* Volume 46, Issue 6, November, Pages 656-678. Collaborative Information Seeking.
- Google. (2012) “What you'll find with Search plus Your World”, <http://www.google.com/insidesearch/plus.html>. March.
- Horowitz, D., Kamvar, D. K. (2010) “The Anatomy of a Large-Scale Social Search Engine”. 19th international conference on World Wide Web, pp. 431-440.
- Huang, X., Croft, B. (2009) “A Unified Relevance Model for Opinion Retrieval”. 18th ACM conference on Information and knowledge management.
- Li, S., Wang, Z., Zhou, G., Lee, S. (2011) “Semi-Supervised Learning for Imbalanced Sentiment Classification”. International Joint Conferences on Artificial Intelligence.
- Liu, B. (2012) “Opinion Mining. Technical Report”, <http://www.cs.uic.edu/~liub/FBS/opinion-mining.pdf>. March.
- Liu, B. (2010) “Sentiment Analysis and Subjectivity”. Workshop of Social Theory and Social Computing: Steps to Integration, March 22-2.
- Manning, C. D., Raghavan, P., Schütze, H. (2008) *Introduction to Information Retrieval*. Cambridge University Press. Cambridge University Press.
- Mishne, G. (2006) “Multiple Ranking Strategies for Opinion Retrieval in Blogs”. The Fifteenth Text REtrieval Conference.
- Morris, M. R., Teevan, J., Panovich, K. (2010a) “A Comparison of Information Seeking Using Search Engines and Social Networks”. 4th International AAAI Conference on Weblogs and Social Media, pp. 291-294.
- Morris, M. R., Teevan, J., Panovich, K. (2010b) “What Do People Ask Their Social Networks, and Why? A Survey Study of Status Message Q&A Behavior”. 28th international conference on Human factors in computing systems.
- Russell, S. Norvig, P. (2009) *Artificial Intelligence: A Modern Approach*. Prentice Hall. 3rd Edition.
- St-Amour, A. (2010) “Just Dance 2 Review”. <http://gamrreview.vgchartz.com/review/45671/just-dance-2/>. March.
- Sherrets, D. (2008) “Google’s Marissa Mayer: Social search is the future”. VentureBeat, Jan 31, 2008. <http://venturebeat.com/2008/01/31/googles-marissa-mayer-social-search-is-the-future/>.
- Zhang , M., Ye, X. (2009) “A Generation Model to Unify Topic Relevance”. 31st annual international ACM SIGIR conference on Research and development in information retrieval.

Um estudo sobre alinhamento de ontologias no domínio de reputação de agentes

Marcos Inky Tae e Anarosa A F Brandão

Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica - Universidade de São Paulo (USP)
São Paulo – SP – Brasil

Abstract. Nowadays, the role of open and distributed systems is gaining an increasing importance within the society. Such systems may consist of dynamic environments populated by organized and cognitive entities, also known as open multiagent systems. In order to achieve the social control in such systems, some strategies that are adopted may raise some issues related to interoperability among agents that interact about reputation. The SOARI architecture deals with the semantic interoperability during interaction about reputation, but it adopts a semi-automatic approach to align the ontologies that describe heterogeneous reputation models. In this work we describe results about the viability analysis of adopting an automatic approach to align ontologies in the SOARI architecture.

Resumo. Atualmente observamos a crescente dependência da sociedade em relação aos sistemas abertos e distribuídos, que podem ser representados por ambientes físicos e dinâmicos populados por redes de entidades cognitivas auto-organizadas, os sistemas multiagentes (SMAs) abertos. Estratégias adotadas para o controle social em SMAs abertos levantam questões relacionadas a interoperabilidade no domínio de reputação de agentes. A arquitetura SOARI tratou este problema adotando uma abordagem semi-automática para o alinhamento de ontologias sobre reputação de agentes. Neste trabalho descrevemos os resultados da análise de viabilidade de automatização completa do mapeamento de ontologias para a arquitetura SOARI.

1. Introdução

Atualmente observamos a crescente dependência da sociedade em relação aos sistemas abertos e distribuídos, os quais podem ser definidos como sistemas computacionais que representam ambientes físicos e dinâmicos populados por redes de entidades cognitivas auto-organizadas (Fredriksson and Gustavsson, 2003). Por suas características, pode-se dizer que sistemas multiagentes (SMAs) abertos são soluções naturais para o desenvolvimento deste tipo de sistemas (Luck et al, 2005).

Como decorrência natural surgem questões relacionadas à forma de auto-organização em SMAs abertos, assim como ao seu controle social, tendo em vista o trânsito dos agentes nestes sistemas. Algumas delas estão fortemente ligadas ao conceito de interoperabilidade, que pode ser definida como a habilidade de troca e compartilhamento de informações entre dois ou mais sistemas ou componentes (IEEE, 1991). Para haver tal troca e compartilhamento, estes sistemas ou componentes devem ser capazes de acessar, processar e interpretar a informação. Isto significa que assuntos relacionados à heterogeneidade entre estes sistemas ou componentes podem comprometer as atividades que devem ser executadas para atingir interoperabilidade. Tais

atividades são atividades de integração e podem ser classificadas relativamente a três dimensões: estrutural, sintática e semântica. Neste trabalho o foco recai sobre a integração semântica, que se refere à resolução de conflitos semânticos entre fontes de informações heterogêneas. Mais especificamente, nosso interesse está na análise de viabilidade da adoção de uma solução totalmente automatizada para tratar a interoperabilidade semântica durante interações entre agentes cognitivos no domínio de reputação.

Em sistemas abertos, onde os agentes podem entrar e sair a qualquer tempo, a possibilidade de interação expõem tais sistemas a riscos como, por exemplo, a tomada de decisão baseada em informações fornecidas por agentes desconhecidos ou maldosos.

Para tratar este tipo de problema em SMAs, foram criados vários modelos computacionais para avaliar a reputação de agentes (Huynh et al, 2004; Muller and Vercouter, 2008; Mui et al, 2002; Sabater and Sierra, 2002; Sabater et al, 2006) como critério para a tomada de decisão sobre a participação ou não dos agentes nas interações sociais. Esta proliferação de modelos de reputação de agentes e a heterogeneidade inerente a isso trouxe consigo o problema da interoperabilidade entre modelos de reputação de agentes. Neste caso, a questão é focada na interação entre agentes cognitivos cujos modelos internos de reputação são distintos, quando seus interesses recaem na avaliação da reputação de outros agentes.

Uma solução possível para este problema foi proposta através da arquitetura SOARI (Nardin et al, 2008a; Nardin et al, 2008b; Nardin et al, 2008c; Nardin, 2009). Trata-se de uma arquitetura orientada a serviços baseada no uso de uma ontologia comum de domínio (FORe) (Casare and Sichman, 2005a; Casare and Sichman, 2005b) como interlíngua, que permite a interação sobre reputação entre agentes que utilizam modelos de reputação distintos. Porém, agentes SOARI usam uma abordagem semi-automática para o mapeamento dos conceitos oriundos de ontologias que descrevem outros modelos de reputação e a FORe.

Neste trabalho apresentamos uma análise de viabilidade da adoção de uma abordagem completamente automática para o mapeamento entre conceitos de ontologias que descrevam modelos de reputação e a FORe em agentes SOARI. Nele foram estendidas e consolidados os resultados apresentados em (Tae and Brandão, 2011).

O artigo está estruturado da seguinte forma: na seção 2 descrevemos algumas soluções para tratar o problema de interoperabilidade semântica no domínio de reputação de agentes; na seção 3 descrevemos algumas ferramentas para alinhamento de ontologias, Alignment API (Euzenat et al, 2011), Lily (Wang and Xu, 2009) e RiMOM (Wang et al, 2010). Na seção 4 apresentamos as ontologias descrevendo dois modelos de reputação, Repage (Sabater et al, 2006) e Typology of Reputation (Mui et al, 2002), desenvolvidas para realizar o mapeamento automático, seguidos, na seção 5, pela apresentação dos resultados destes mapeamentos. Finalmente, na seção 6 apresentamos a partir de uma análise qualitativa, as conclusões sobre a possibilidade de adoção de abordagem automática de mapeamento em agentes SOARI.

2. Interoperabilidade semântica no domínio de reputação de agentes

Um dos primeiros trabalhos a tratar do problema de interoperabilidade entre modelos de reputação de agentes foi a proposição de uma ontologia funcional de reputação (FORe) (Casare and Sichman, 2005a; Casare and Sichman, 2005b). FORe "subsumia" grande parte das ontologias que representavam os modelos de reputação existentes, podendo então ser adotada como um

vocabulário comum no domínio de reputação de agentes, seguindo a abordagem híbrida proposta em (Visser et al, 2000). Tal abordagem, esquematizada na Figura 1, pressupõe que cada agente tenha sua própria ontologia, porém o vocabulário usado pelas ontologias dos agentes sejam concordantes com o vocabulário da ontologia comum.

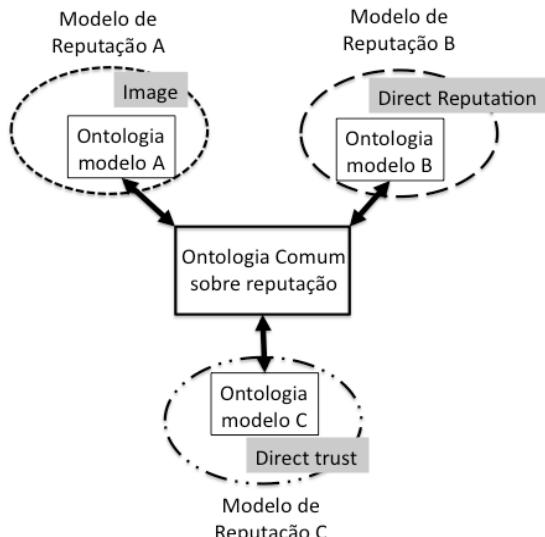


Figura 1: Modelo Híbrido para tratar interoperabilidade semântica

A arquitetura SOARI foi definida a partir da extensão da proposta de uma arquitetura geral para interação semântica sobre reputação (Vercouter et al, 2007, Brandão et al, 2007a,Brandão et al, 2007b), onde era previsto o uso da FORe como vocabulário comum para tratar o problema de interoperabilidade semântica no domínio de reputação de agentes. SOARI é adota uma abordagem orientada a serviços e é esquematizada na Figura 2. Nela observamos que o mapeamento entre conceitos das ontologias é feito por um serviço externo ao agente, evitando sobrecarga de processamento internamente ao agente e mantendo o sigilo sobre os modelos internos de reputação dos agentes envolvidos nas interações. O mapeamento, ou alinhamento de ontologias identifica, entre ontologias distintas, quais são os conceitos equivalentes ou similares, dependendo do grau de confiança associado.

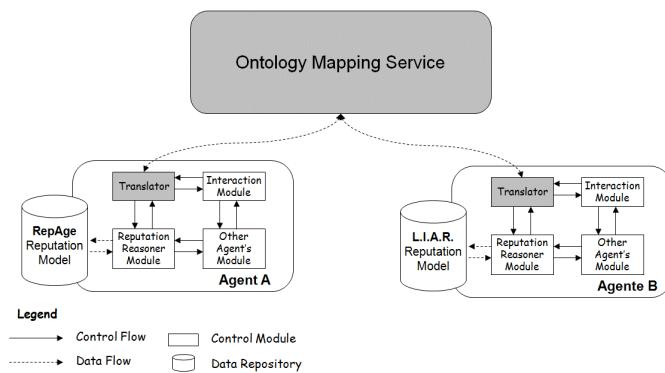


Figura 2: Arquitetura SOARI (fonte Nardin et al, 2008c)

Experimentos demonstraram que agentes SOARI identificam mais rapidamente agentes com má reputação, durante interações sobre reputação, quando comparados a agentes que não fazem uso

de semântica durante as interações (Nardin et al, 2011). Assim, a extensão da arquitetura para que o serviço de mapeamento seja completamente automático facilitaria sobremaneira sua adoção pelo mercado.

3. Ferramentas para alinhamento de ontologias

Tendo em vista a automação do alinhamento de ontologias na arquitetura SOARI, foram estudadas algumas ferramentas para alinhamento de ontologias, sendo dada uma breve descrição de cada uma delas.

A ferramenta Alignment API (Euzenat et al, 2011) é um conjunto de ferramentas que oferece funcionalidades de alinhamento, acesso, descrição e compartilhamento de alinhamentos de ontologias. É implementada em Java, onde cada classe API oferece implementações básicas para armazenamento, manipulação e comparação de alinhamentos, inclusive com possibilidade de escolha de algoritmos de mapeamento e geração pré-processadas de saídas. Provê suporte abstrato para acessar uniformemente parte da ontologia que pode ser útil para os algoritmos de mapeamento, através do OntoWrap e métricas para o cálculo de similaridades entre conceitos de ontologias distintas, através do OntoSim. Além disso, também disponibiliza uma linguagem para descrição de alinhamentos, chamada EDOAL (Expressive and Declarative Ontology Alignment Language), elaborada para capturar mais precisamente correspondências entre entidades ontológicas.

Lily (Wang and Xu, 2009) é um sistema de alinhamento que adota estratégias híbridas para executar o alinhamento de ontologias. Possui quatro funções principais: (i) método genérico para alinhamento de ontologias, usado para ontologias de pequeno porte; (ii) método de alinhamento para ontologias de grande porte; (iii) método de alinhamento semântico de ontologias, usado para descobrir relações semânticas entre ontologias; e (iv) depuração de alinhamentos de ontologias, usado para melhorar os resultados dos alinhamentos. Sua interface gráfica permite ao usuário selecionar as ontologias a serem alinhadas, assim como o alinhamento de referência a ser adotado. A partir destas informações é feito o alinhamento e calculada a qualidade do mesmo, conceito a conceito.

RiMOM (Wang et al, 2010) é um sistema para alinhamento dinâmico de ontologias usando várias estratégias de alinhamento. O sistema pode combinar várias estratégias dinamicamente para gerar alinhamentos cujos resultados se beneficiem da combinação de estratégias. Consiste de três camadas: (i) a camada de interface, que implementa a interface gráfica que permite a interação do usuário com o sistema e personalização do procedimento de alinhamento; (ii) a camada de tarefa, que armazena os parâmetros para as tarefas de alinhamento e controla a execução do processo; e (iii) camada de componentes, que engloba cinco grupos de componentes executáveis (pré-processador, alinhador, agregador, pós-processador e avaliador), que são instanciados e executados pelo usuário.

4. Modelos de reputação de agentes e respectivas ontologias

Os modelos de reputação de agentes usados neste trabalho foram Repage (Sabater et al, 2006) e Typology of Reputation (Mui et al, 2002), pois os mesmos já dispunham de ontologias descritas seguindo o vocabulário comum proposto pela FOrE. Como não foram encontradas outras descrições usando ontologias para os modelos adotados, a estratégia adotada neste trabalho foi construir ontologias para descrever os modelos partindo do zero, sem conhecimento prévio do

vocabulário comum da FORe. Em (Tae and Brandão, 2011) apresentamos versões iniciais das ontologias bem como os resultados dos alinhamentos obtidos entre cada ontologia e a FORe.

O modelo Repage é baseado em dois conceitos principais: Imagem e Reputação. Imagem é definida como sendo uma avaliação de crença, que é formada a partir da informação adquirida pelo agente por experiência própria ou por imagens propagadas por terceiros. Reputação é definida como sendo uma crença sobre a existência de uma avaliação transmitida, que pode não representar, necessariamente, a verdade. Trata-se de uma meta-crença, baseada no valor de reputação transmitido anonimamente numa rede social da qual o agente que se avalia a reputação faz parte.

O modelo Typology foi construído tendo como base uma tipologia de vários modelos de reputação classificando-a, inicialmente como individual ou global. São dez conceitos distintos organizados hierarquicamente, como pode ser visualizado na Figura 3.

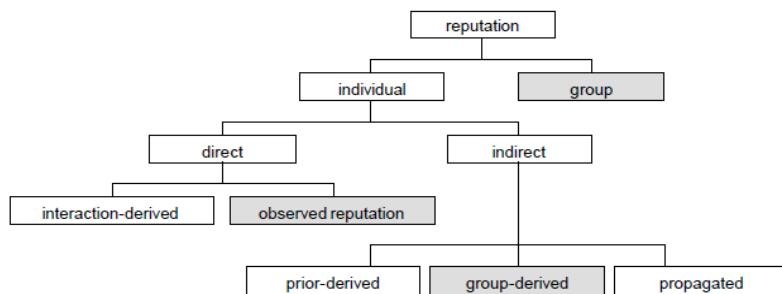


Figura 3: Typology of Reputation (fonte (Mui et al, 2002))

As ontologias desenvolvidas para os modelos e apresentadas em (Tae and Brandão, 2011) foram refinadas tendo em vista o melhor entendimento de ambos os modelos pelo desenvolvedor e os resultados desencorajadores dos alinhamentos produzidos. Assim, introduziu-se alguns conceitos e relacionamentos em ambas ontologias. Na Figura 4 observa-se a taxonomia e relações das ontologias usadas em (Tae and Brandão, 2011).

A Figura 5 mostra um *snapshot* da ontologia Repage apos o refinamento. Sua estrutura foi revista, sendo os conceitos *Reference*, *Identified* e *Unidentified* suprimidos e os conceitos *Image*, *Reputation*, *SharedEvaluation* e *SharedVoice* passaram a ser subconceitos do novo conceito *Evaluation*. Novos relacionamentos também foram incluídos com o intuito de prover a ontologia com mais detalhes que pudessem, eventualmente, auxiliarem os alinhadores.

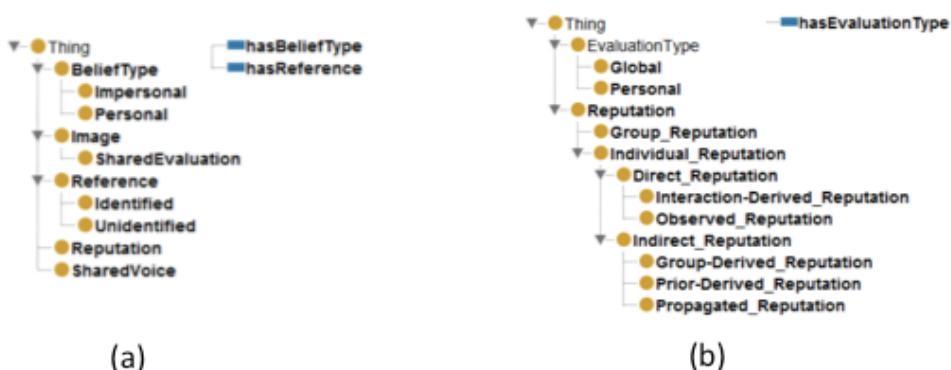


Figura 4: Ontologias antes do refinamento: Repage (a) e Typology of Reputation (b)

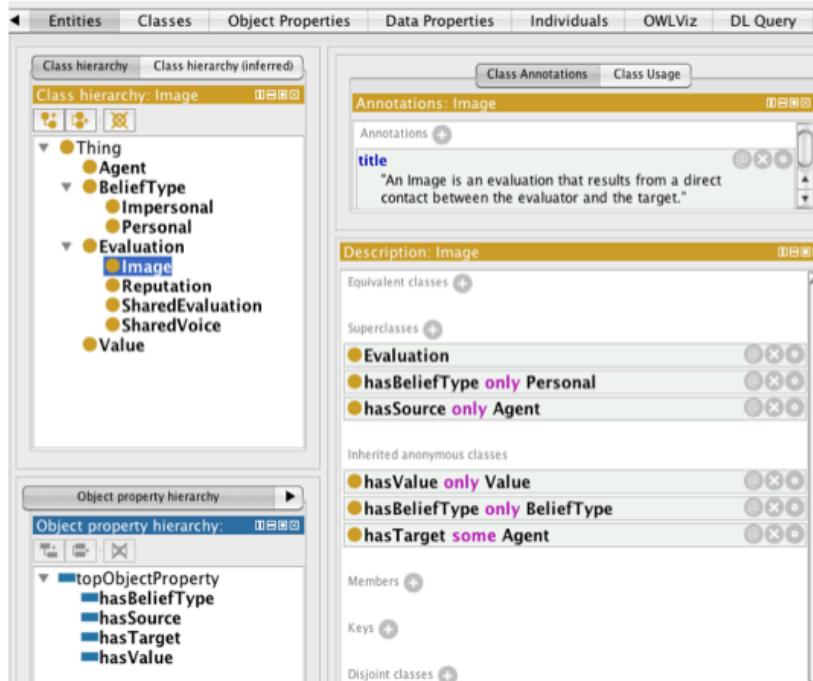


Figura 5: Ontologia Repage refinada

A Figura 6 mostra um snapshot da ontologia Typology, onde também pode ser observada a inclusão de novos conceitos e relacionamentos, como *InformationSource*, *InteractionType* e respectivos subconceitos.

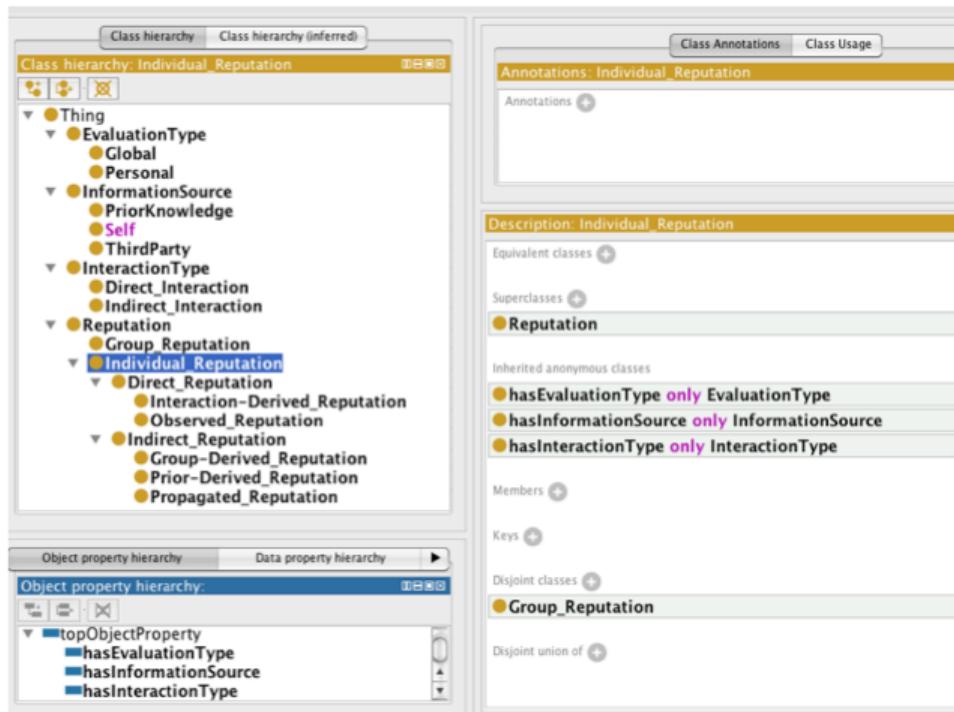


Figura 6: Ontologia Typology refinada

5. Alinhamentos obtidos

Por motivos de espaço, são apresentados apenas os alinhamentos obtidos usando a ferramenta Lily nas tabelas 1 e 2. Das tabelas constam, na primeira coluna da esquerda, os conceitos da ontologia que se pretende alinhar, na segunda coluna, os conceitos correspondentes na ontologia FOrE após o alinhamento. A terceira coluna indica a confiança avaliada pelo sistema de alinhamento e atribuída ao alinhamento entre entidades correspondentes de cada linha e a última coluna apresenta a opinião de um especialista do domínio sobre a exatidão ou não do alinhamento.

Tabela 1: Alinhamento realizado entre Repage e FOrE usando Lily

Repage	FOrE	Confiança	Especialista do domínio
BeliefType	Agent	0.5	Não
Impersonal	GroupOfReputation	0.4715	Não
Personal	Individual	0.4971	Não
Entity	NonAgent	0.4643	Não
Agent	IndividualReputation	0.2114	Não
SetOfAgents	TransmitterReputation	0.0676	Não
Evaluation	ReputationEvaluationProcess	0.1372	Não
Image	ActivityReputation	0.0354	Não
Reputation	GroupReputation	0.2097	Não
SharedEvaluation	EvaluatorRole	0.069	Não
SharedVoice	ReputationType	0.0187	em termos
Value	ValuePartition	0.1037	Não
hasBeliefType	isReputationOf	0.5482	Não
hasSource	isInformationSourceOf	0.3333	OK
hasTarget	hasResponsabilityAssigmentBy	0.5912	Não
hasValue	hasOutput	0.1799	em termos
isBeliefTypeOf	hasReputation	0.5938	Não
isSourceOf	hasInformationSource	0.372	OK
isTargetOf	isResponsabilityAssignerFor	0.6086	Não
isValueOf	isOutputOf	0.166	em termos

Pela tabela 1 apresentada observa-se que a confiança atribuída aos alinhamentos obtidos entre as ontologias Repage e FOrE usando Lily foi baixa, na maioria, considerando-se que ela varia no intervalo [0,1]. Além disso, de acordo com o especialista do domínio, apenas dois alinhamentos foram considerados corretos e três aceitáveis, num universo de vinte possíveis.

Tabela 2: Alinhamento realizado entre Typology e FORe usando Lily

Typology	FORe	Confiança	Especialista do domínio
EvaluationType	ReputationEvaluationProcess	0.0476	Não
Global	Observation	0.1619	OK
Personal	DirectExperience	0.1514	OK
InformationSource	InformationSource	0.1874	OK
Self			
PriorKnowledge	Prejudice	0.1172	OK
ThirdParty	SecondHandInformation	0.0925	OK
InteractionType	ReputationType	0.2751	Não
Direct_Interaction			
Indirect_Interaction	GroupInheritance	0.0327	OK
Reputation	TargetRole	0.0916	Não
Group_Reputation	GroupReputation	0.1692	em termos
Individual_Reputation	IndividualReputation	0.2416	OK
Direct_Reputation	PrimaryReputation	0.4223	OK
Interaction-Derived_Reputation	DirectReputation	0.5258	OK
Observed_Reputation	ObservedReputation	0.5859	OK
Indirect_Reputation	SecondaryReputation	0.3122	OK
Group-Derived_Reputation	StereotypedReputation	0.1514	em termos
Prior-Derived_Reputation	CollectiveReputation	0.3337	em termos
Propagated_Reputation	PropagatedReputation	0.4223	OK
Target	TargetBehavior	0.1081	Não
Value	ValuePartition	0.0941	Não
hasEvaluationType	isReputationOf	0.2584	Não
hasInformationSource	isInformationSourceOf	0.7054	OK
hasInteractionType			
hasTarget	hasResponsibilityAssignmentBy	0.3778	Não
hasValue	hasOutput	0.1427	em termos
isEvaluationTypeOf	hasReputation	0.4351	Não
isInformationSourceOf	hasInformationSource	0.6914	OK

isInteractionTypeOf			
isTargetOf	isReponsabilityAssignerFor	0.4636	Não
isValueOf	isOutputOf	0.1429	em termos

Novamente observou-se que a confiança atribuída aos alinhamentos obtidos entre as ontologias Typology e FORe usando Lily foi baixa, na maioria, tendo inclusive media menor que a obtida pelo alinhamento (Repage, FORe). Além disso, algumas entidades simplesmente não obtiveram correspondentes na FORe. Apesar dos resultados, a opinião do especialista do domínio mostra que o alinhamento foi bem sucedido pois foi correto em 42% dos casos (14/33) e aceitável em 15% dos casos (5/33).

Foram realizados alinhamentos com a ferramenta Alingment API, com resultados cuja qualidade avaliada pelo especialista do domínio foi classificada como inferior à apresentada pelos alinhamentos realizados usando Lily.

6. Conclusões e trabalhos futuros

Os alinhamentos realizados indicam que as ferramentas para alinhamento de ontologias usadas mostraram-se pouco eficientes para realizar o mapeamento de conceitos de ontologias no domínio de reputação de agentes.

Acreditamos que este resultado se deve, principalmente, ao fato de que as ontologias são usadas em sua essência durante as interações entre os agentes, uma vez que não é feito uso de instâncias de conceitos durante a troca de mensagens envolvendo reputação entre agentes SOARI. Assim, a falta de instâncias de conceitos que poderiam melhor alimentar os algoritmos de alinhamento pode ter sido determinante para a baixa confiança nos alinhamentos obtidos.

Outras ferramentas de alinhamento podem ser analisadas, como por exemplo AgreementMaker (), porém uma análise preliminar sobre a forma como foi implementado o mapeamento na ferramenta indica que a falta de instâncias dos conceitos das ontologias que descrevem os modelos de reputação deve levar a soluções parecidas com as apresentadas neste trabalho.

Até o momento podemos afirmar que a solução semi-automática proposta pela arquitetura SOARI provê melhores resultados que as soluções automáticas pesquisadas.

Agradecimentos

Este trabalho é resultado de projeto apoiado pela FAPESP, processo 2010/20620-5 e pelo CNPq, através do programa PIBIC-USP, processo 146339/2011-8.

7. Referencias

(Brandão et al, 2007a) Brandão, A. A. F. ; Vercouter, L. ; Casare, S. J. ; Sichman, J.S. . Extending the ART Testbed to Deal with Heterogeneous Agent Reputation Models. In: Trust n Agent Societies -TRUST Workshop at Sixth Intl. Joint Conf. on Autonomous Agents and Multiagent Systems, 2007, Honolulu. W20: trust in Agent Societies (TRUST) - May 15th 2007, 2007. v. 1. p. 07-13.

(Brandão et al, 2007b) Brandão, A. A. F. ; Vercouter, L. ; Casare, S. J. ; Sichman, J.S. . Exchanging Reputation Values among Heterogeneous Agent Reputation Models: An Experience on ART Testbed. In: Sixth Intl. Joint Conf. on Autonomous Agents and Multiagent Systems, 2007, Honolulu. Proceedings of

the Sixth Intl. Joint Conf. on Autonomous Agents and Multiagent Systems. Nagar : IFAAMAS, 2007. v. 1. p. 1307-1309.

(Casare and Sichman, 2005a) Casare, S. and Sichman, J. Towards a functional ontology of reputation. In Proc. 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05), Utrecht – The Netherlands, July 2005a.

(Casare and Sichman, 2005b) Casare, S. and Sichman, J. Using a functional ontology of reputation to interoperate different agent reputation models. Journal of the Brazilian Computer Society, 11(2):81–94, 2005b. (Euzenat et al, 2011) David, J., Euzenat, J. Scharffe, F., dos Santos, C.T. (2011) The Alignment API 4.0 Semantic Web Vol 2, 1, IOS Press, <http://dx.doi.org/10.3233/SW-2011-0028>

(Fredriksson and Gustavsson, 2003) Fredriksson, M and Gustavsson, R. Quality of service in network-centric warfare and challenges in open computational systems engineering. In Proceedings of 20th International Workshop on Enabling technologies: Infrastructure for collaborative enterprises (WETICE), 2003.

(Huynh et al, 2004) Huynh, T. D., Jennings, N. R., Shadbolt, N., 2004. Fire: An integrated trust and reputation model for open multi- agent systems. In: 16th European Conference on Artificial Intelligence. pp. 18–22. URL <http://eprints.ecs.soton.ac.uk/9559/>

(IEEE, 1991). IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries. IEEE Computer Society Press, New York, USA.

(Luck et al, 2005) Luck, M., McBurney, P., Shehory, O. and Willmott, S. Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink, 2005.

(Mui et al, 2002) Mui L.; A. Halberstadt; M. Mohtashemi. Notions of Reputation in Multi-Agents Systems: A Review. In: Proceedings of 1st International Joint Conference on Autonomous Agents and Multi-agent Systems.AAMAS'02, July 15-19, 2002, Bologna, Italy.

(Muller and Vercouter, 2008) Muller, G., Vercouter, L., 2008. L.i.a.r. achieving social control in open and decentralised multi-agent systems. Tech. rep. , Ecole Nationale Supérieure des Mines de Saint-Etienne, Saint-Etienne, France.

(Nardin et al, 2008a) Nardin, L.G., Brandão, A.A.F., Sichman, J. and Vercouter, L. An ontology mapping service to support agent reputation models interoperability, In Proceedings of the TRUST in Agent Societies Workshop at AAMAS'2008, Estoril, Portugal, 2008.

(Nardin et al, 2008b) Nardin, L.G. ; Brandão, A. A. F. ; Sichman, J.S. ; Vercouter, L. A Service-Oriented Architecture to Support Agent Reputation Models Interoperability. In: 3rd Workshop on Ontologies and their Applications (WONTO 2008), 2008, Salvador. Proceedings of the 3rd Workshop on ontologies and their applications (WONTO 2008), 2008. v. 427.

(Nardin et al, 2008c) Nardin, L.G. ; Brandão, Anarosa A.F. ; Sichman, J.S. ; Vercouter, L. . SOARI: A Service Oriented Architecture to Support Agent Reputation Models Interoperability. In: Rino Falcone; Suzanne K. Barber; Jordi Sabater-Mir; Munindar P. Singh. (Org.). Trust in Agent Societies 11th

International Workshop, TRUST 2008, Estoril, Portugal, May 12-13, 2008 - Revised Selected and Invited Papers. Berlin: Springer, 2008, v. 5396, p. 292-307.

(Nardin et al, 2009) Nardin, L.G. ; MULLER, G. ; Brandão, Anarosa A.F. ; Vercouter, L.; Sichman, J.S. . Effects of expressiveness and heterogeneity of reputation models in the ART-Testbed: Some preliminary experiments using the SOARI architecture. In: Workshop on Trust in Agent Societies, 2009, Budapest. Autonomous Agents and Multi Agent Systems 2009 (AAMAS-09) Workshop on, 2009. v. 1. p. 1-12.

(Nardin, 2009) Nardin, L.G. Uma arquitetura de apoio à interoperabilidade de modelos de reputação de agentes. 2009. Dissertação de Mestrado (Engenharia Elétrica) - Universidade de São Paulo.

(Nardin et al, 2011) Nardin, L.G., Brandão, A.A.F., Sichman, J.S.. Experiments on semantic interoperability of agent reputation models using the SOARI architecture. Engineering Applications of Artificial Intelligence (2011),doi:10.1016/j.engappai.2011.05.004

(Sabater et al, 2006) Sabater-Mir, J., Paolucci, M., Conte, R., 2006. Repage: Reputation and image among limited autonomous partners. Journal of Artificial Societies and Social Simulation 9 (2). URL <http://jasss.soc.surrey.ac.uk/9/2/3.html>

(Sabater and Sierra, 2002) Sabater-Mir, J., Sierra, C., 2002. Social regret, a reputation model based on social relations. SIGecom Exch. 3 (1), 44–56

(Tae and Brandão, 2011) Tae, M.I. and Brandão, A.A.F. Towards automating ontologies alignment to support autonomous agent's interaction about reputation. In: Anais do II CBSoft: Congresso Brasileiro de Software: teoria e Pratica, II Workshop on Autonomous Software Systems – AutoSoft 2011, vol 10, pp 90-100.

(Vercouter et al, 2007) Vercouter, L.; Casare, S. J. ; Sichman, J.S. ; BRANDÃO, A. A. F. . An Experience on Reputation Models Interoperability based on a Functional Ontology. In: Twentieth International Joint Conference on Artificial Intelligence, 2007, Hyderabad. Proceedings of the 20th International Joint Conference on Artificial Intelligence - IJCAI 2007, 2007. p. 617-622.

(Visser et al, 2000) Visser, U., Stuckenschmidt, H., Wache, H., Vogele, T., 2000. Enabling technologies for interoperability. In: Visser, U., Pundt, H. (Eds.), Proceedings of the Workshop on the 14th International Symposium of Computer Science for Environmental Protection. TZI, Bonn, pp. 35–46.

(Wang and Xu, 2009) Wang, P., Xu, B., 2009. Lily: Ontology alignment results for oaei 2009. In: Proceedings of the ISWC 2009 Workshop on Ontology Matching. Vol. 551 of CEUR Workshop Proceedings. CEUR-WS.org, Linkping, Sweden, pp. 186–192.

(Wang et al, 2010) Wang, Z., Zhang, X, Hou, L., Zhao, Y., Li, J., Qi, Y. and Tang, J. (2010) RiMOM results for OAEI 2010, in Pavet at al (Eds) Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) collocated with the 9th International Semantic Web Conference (ISWC-2010), Shanghai, China, November 7, 2010., CEUR Workshop Proceedings, vol. 689 <http://ceur-ws.org/Vol-689/>

ADAM: An Autonomous Agent for High-Frequency Currency Trading in the Brazilian Market

Vicente Matheus M. Zuffo, Paulo André L. de Castro

Divisão de Ciência da Computação (IEC) – Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos – SP – Brazil

vicentematheus@gmail.com, pauloac@ita.br

Abstract. This paper presents an autonomous system called ADAM (Autonomous Dollar futures Agent Model) that implements a strategy based in the “Follow The Leader” concept to trade liquid futures contracts in the Brazilian exchange BM&F, taking in consideration peculiarities of the respective market. The agent operates at high frequency, being one of the first of its kind in this market.

1. Introduction

Nowadays, the so called high-frequency trading (trades made in short time intervals between each other) are responsible for about seventy percent of the American financial market. Although it is growing rapidly, in Brazil this figure is yet around five percent. They are characterized as being predominantly performed by computer algorithms. Basically, the set of factors that determine the investor decisions are compressed into strategies that allow the development of programs which, in theory, simulate the characteristics of the investor.

Many hedge funds, banks and brokers with a strong operation in the United States are opening offices in Brazil to explore the robustness of the local market (one of the largest and safest stock exchanges in the world). Within this scenario, it is happening a lot of investment in technology for the creation of infrastructure and software, aiming the market opportunities that can be leveraged trading at high frequency.

In this study, we implemented a high-frequency strategy based on the “Follow-The-Leader” concept, through an algorithm called ADAM (Autonomous Dollar futures Agent Model). It is an autonomous system for trading dollar futures contracts in the “Bolsa de Mercadorias e Futuros” (BM&F). This type of algorithm is free of human interference in its decisions, since it negotiates according to a pre-established model.

It was chosen the dollar futures market at BM&F as the tool of study due to its high liquidity (measured in volume of trades) and the price of the contract is approximately continuous (the minimum interval between orders is fifty “centavos” per thousand dollars and it has a satisfactory intraday volatility).

The motivation to implement the proposed strategy is to use the concept of high frequency as an advantage over traditional trading agents and to explore opportunities to gain in the dollar futures market in particular. Differently from the usual literature about quantitative investment strategies (mostly based on Technical Analysis), this paper introduces an idea of approaching the market players based on the concept of a leader

agent. In addition, it explores a technology branch still very incipient in Brazil, which is the high-frequency trading algorithm.

For testing the algorithm, it was created a simple market simulation platform, which allows the use of real data (historical) from FIX protocol.

2. Dollar Futures Market in the BM&F

Currency futures are contracts in which is negotiated a standard amount of one currency against another. Basically, at the time of the deal, it is fixed a conversion rate of a certain amount of one currency for another at a future date. In the BM&F, the most negotiated currency futures contract is the dollar future. Commonly, each contract defines an amount of US\$ 50.000 and maturity date as the first business day of each month.

The trade is always performed between two parts (a buyer and a seller) and, at the end of every day, according to the variation of the dollar futures rate, the difference of the contract value is paid or charged to the respective part. Thus at the maturity date, the total contract value is not transferred from one part to another, but only the difference between the initial and the final contract values (in Brazilian currency, real), according to the exchange rate.

For instance, a dollar future for June 2012 is a contract, which amount value is US\$ 50.000, and it expires in the first working day of June. Assuming that the rate negotiated between the buyer A and the seller B for this contract is 1.6 reais per dollar, then in the first working day of august, assuming that the exchange rate is 1.7 reais per dollar, then the buyer A will receive a positive difference resulted from the rate variation, which in this case is $US\$ 50.000 \times (1,7 - 1,6) R\$ / US\$ = R\$ 5.000$, while the seller B should pay only this difference and not the entire value of the contract (US\$ 50.000).

In the BM&F, the quotes for these contracts are given as reais per thousand dollars with one decimal place and the minimum price fluctuation is fifty centavos per thousand dollars. That is to say the least negotiable price greater than R\$ 1.600 per US\$ 1.000 will be R\$ 1.600,5 per US\$1.000.

The most traded contract among the dollar futures contracts is the one that expires on the first day of the next month of the trade date, because it is the closest to a spot trade. In addition, the quantities negotiated are always multiple of five, i.e., an amount equivalent to US\$ 250.000.

Naturally, the Supply and Demand Law applies to the pricing of any asset traded on a stock exchange. Basically, the agents willing to buy the asset publish buy orders for a specified price on a certain number of contracts. Similarly, the agents willing to sell it publish sell orders with higher prices. Thus are formed the so called bid and ask books, which line up all the buy and sell orders available in the market.

A trade occurs when an agent decides to buy contracts at a price greater than or equal to the first order of the ask book or when an agent decides to sell a contract at a price less than or equal to the first order of the bid book.

The only institutions authorized to buy and sell these futures contracts at BM&F are the securities brokers. Thus, every investor must have an account at a brokerage

house in order to participate in the market. That is, brokers act as intermediary elements between the investors and the exchange, where the trades occur. Particularly in Brazil, the information published by BM&F to the market, regarding to the bid and ask books contain not only the prices and quantities of each order, but also the name (or code) of the brokerage house involved, information that is not present in the United States where the markets are called “blind”.

Tables 1 and 2 illustrate examples of bid and ask books of the Brazilian dollar futures market:

Table 1. Example of the Bid Book

Brokerage House	Quantity	Price (R\$ / US\$ 1.000)
BM000188	10	1.700,0
BM000072	50	1.700,0
BM000213	25	1.699,5
BM000012	5	1.699,5
BM000001	5	1.699,0

Table 2. Example of the Ask Book

Brokerage House	Quantity	Price (R\$ / US\$1.000)
BM000174	20	1.700,5
BM000030	30	1.701,0
BM000255	15	1.701,0
BM000008	10	1.701,0
BM000009	15	1.701,5
BM000008	5	1.701,5

Naturally, the first price of the bid book is always lower than the first price of the ask book. In addition, orders with the same price are classified chronologically according to their arrival instant in the BM&F servers. It is possible to change parameters of an order previously sent, since it is still available in the market. In the case of a change in the price or an increase in the number of contracts of the order, it goes to the end of the line between the orders of the desired price. In the case in which only the number of contracts is decreased, the order does not lose its position in the order queue.

3. Strategy Description

The financial market is composed of a multitude of agents. In particular, the dollar futures market on the BM&F is one of the most liquid which means it is extremely accessible, where there are a huge number of agents trading a high financial volume.

Within this context, it is assumed that there is at least one agent (or set of agents, as discussed below) on the market that somehow manages to obtain positive results consistently over various periods of time. This agent is then called leader. The strategy is based on a “follow the leader” tactics.

There are several possibilities for the consistent gain of the leader. Among those options, one can highlight the following:

- The leader has a considerable amount of resources so that his investment decisions (his price expectations) could alter significantly the prices of certain assets, even very liquid ones in a way that even others actors feel compelled to reassess their own decision because of considerable movements in the asset price;
- The leader is able to perform an analysis that is consistent with market movements, be it technical or fundamental. This capability can exist for skill or might have been developed with experience;
- The leader has some insider information that gives him extreme advantage over other market players.

Finally, the leader is an agent capable of obtaining considerable gains in certain movement in the price of an asset. Roughly, the strategy consists in, after a certain period of time, identify an agent as the leader and try to perform the same trades that he does.

The first problem identified in the development of ADAM was the time interval considered for the assessment of the leader, because this time interval must necessarily be less than the duration of the price movement. Otherwise, the result obtained by the movement does not happen.

For the agent proposed, the leader is defined as one that has the largest result function within the time interval evaluated. This definition does not guarantee that the agent chosen as leader in fact presents a consistent gain in the coming periods, but given the information publicly available by the market, this heuristic becomes the most coherent. To determine the result of an agent in the financial market (and the method is very similar for all types of assets) is necessary to monitor all trades made by this agent.

A trade in which the agent is one of the two parts involved can be represented by the pair $O = (Q, P)$, being Q the quantity negotiated and P the price (or quote).

Suppose that a particular agent A bought Q contracts of an asset at a price P. For the derivatives market, where the agent does not necessarily have the physical asset, as the settlement is given only by paying or receiving the financial adjustment of the price change, given such operation, it is Said that A is long and B is short in Q contracts, both at the price P.

Thus, if there is a movement in the price of the asset from P to P', then the respective result functions of the agents will be given by:

$$Res(A) = Q \times T \times (P' - P) \quad (1)$$

$$Res(B) = -Q \times T \times (P' - P) \quad (2)$$

Where T is the size (financially) of the contract.

If the agent made several trades, his generalized result function is given by:

$$Res(A) = T \times [P_{\text{MARKET}} \times Position(A) - \sum Q_{C,i} P_{C,i} + \sum Q_{V,i} P_{V,i}] \quad (3)$$

Where:

- Q_C and P_C correspond, respectively, to the Quantity and to the Price of an buy order traded by agent A;
- Q_V and P_V correspond, respectively, to the Quantity and to the Price of an sell order traded by agent A;
- P_{MARKET} is the last price negotiated in the market;
- $Position(A) = \sum Q_{C,i} - \sum Q_{V,i}$ (4)

As previously mentioned, in the Brazilian financial market, the buying and selling brokerage houses of each order are available information, published by the Exchange, both for orders available in the books and for the past trades.

Therefore, for every new trade, it is possible to evaluate the result function for all the brokers in the market. One cannot know which agents negotiate in each brokerage house and neither which agents trade through different brokers. Thus, it is not possible to determine exactly which agent, but only the set of agents (brokerage house) that represents the market leader at the moment.

In the dollar futures market, on average, there are new trades every second (even tens per second). Hence the need to create an algorithm that can do the analysis of market data at high frequency.

To determine the leader to be followed, there have been created two types of definition:

- Static Definition of Leader: agent (broker) with the biggest result function in a given period P (in hours) which, once evaluated, remains the leader throughout the day. That is, every day, only one assessment is made to determine the leader, during the first P hours of the trading time.
- Dynamic Definition of Leader: agent (broker) with the biggest result function after each period of P hours. Thus, it is possible, during the day, that there are many different leaders. The result function for each agent takes into account all the trades made in the day.

It was defined as the operating period of the strategy a whole Day, because it is the longest period in which the market operates in quasi-continuous mode. Therefore, it was decided that the strategy always ends the day with no position, i.e., the sumo f the contracts bought must equal the sumo f the contracts sold.

The algorithm then can be divided into two phases. During the first phase, it is made only the monitoring of the orders in the market. After a period P, the leader is determined, according to the result functions of all agents. Figure 1 shows a simple flowchart of this phase, in which the start is given by each new event that hits the market:

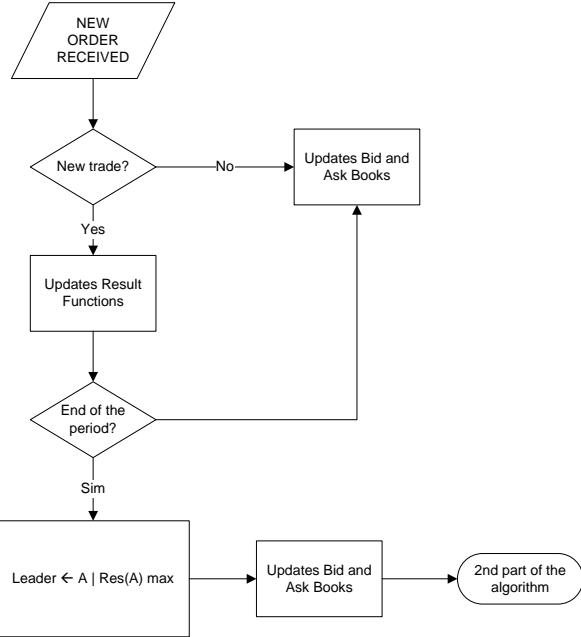


Figure 1. Flowchart of the first phase of the algorithm

Once the leader is determined, the algorithm buys or sells contracts in order to match its position to the leader's position. From this moment, for any new trade, it will check if the leader was one of the parts involved (and only one, so that there is a change in position). If the leader has bought X contracts, it will buy X contracts in the market. If the leader has sold X contracts, it will sell X contracts in the market, so that the positions of the agent and the leader will be the same.

Eventually, the agent will not be able to execute trades at the same price as the leader did, but slightly worse (higher when to buy and lower when to sell, according to market conditions). This difference is meaningful only if the period of analysis for the determination of the leader is not compatible with the average period of price movements for this asset, which generates the leader's advantage. For this reason, we used different periods in the experiments, as described in section V.

So the second part of the algorithm is represented by the flowchart in Figure 2:

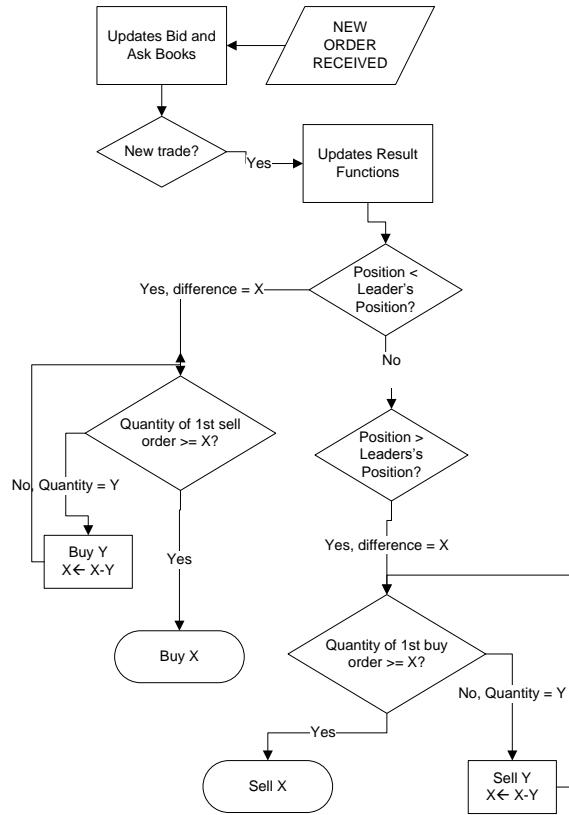


Figure 2. Flowchart of the second phase of the algorithm

For the model that uses dynamic definition of a leader, the re-evaluation of the leader after a new period P happens in a manner analogous to that of the first flowchart, but simultaneously to the second flowchart processing.

In the last minutes of trading, the algorithm stops following the leader and only performs the necessary trades to make its position null, as explained above.

4. Implementation

For testing the model it was developed a trading platform with three operating modes: manual, simulated and historical (with actual market data, from a database).

This platform runs on a server and can be remotely accessed by different processes, each one representing a market agent. It has a simple graphical interface for a better visualization of the bid and ask books. The client and the server send and receive messages via UDP protocol through a simplified model, created only for this purpose, based on FIX protocol (Financial Information Exchange), which is the standard among financial markets worldwide.

Both processes were developed in JAVA language. The platform has a main class called Market which processes the orders received by participants and matches them when a trade is supposed to happen. The different operating modes of the market run on separate threads, simulating the different participants that send orders (messages) to the market.

For the experiments, we used the historical operating mode because it allows the use of real market data, obtained from the BM&F for the period of January 2009. All messages (in FIX protocol) of the dollar futures market of that period have been translated and stored in a database, according to the entity in Figure 3:

Ordem	
PK	id_uniq
	id
	qtde
	preco
	buyer
	seller
	updateaction
	position
	side
	time

Figure 3. Order from the database that stores the historical data

On the Server, there is a separate thread to handle messages received by the new agent, to avoid concurrence with the messages received by the historical mode (database). In this mode, the system is designed in a way the presence of the new agent does not interfere with the order book, so that the simulation accurately represents the same events happened in the actual market, what is reasonable to assume for the dollar futures market due to its high liquidity. In practice, any new agent will interfere with market dynamics, the more the higher quantities are traded.

In Figure 4, there is a snapshot of the market platform with the algorithm in operation:

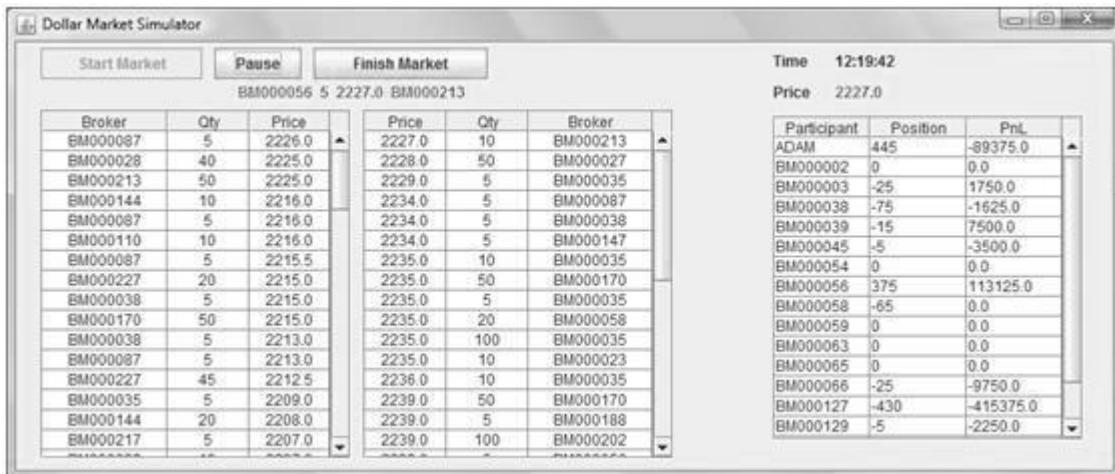


Figure 4. Simulator in execution

5. Experiments

The experiments were divided into two groups. The first group aimed to test the performance of the algorithm using the model with static definition of leader, while the second group used the model of dynamic definition of leader.

As mentioned in section III, when the definition of the strategy was proposed, a central question in the development of the strategy is the length of the period to be considered for the evaluation of the leader. There is no convincing theoretical model that justifies the choice of a certain period over another. In this context, three distinct periods were chosen for both experiments with static and dynamic definitions of leader.

For the first group of experiments (with static definition of leader), the periods chosen for the tests were one hour, two hours and four hours. One entire running market day takes about nine hours. Therefore, the periods were chosen so as to consider about 10%, 20% and 40% of the session duration. A period longer than four hours apparently is not consistent with the proposal because the lower the time that the algorithm works after setting the leader, in theory, the less will be exploited the advantage that the leader provides. In addition, it was considered that different periods would not have an effect significantly different from one of the three chosen periods, since the price movements over a period of one hour are relatively small. If the participant identified as the leader is actually advantageous, so in theory, he should continue to be the leader at least for a short period of time, as one hour.

For the second group of experiments, in which the leader is chosen dynamically, the periods considered were half an hour, one hour and two hours. In this case, it would make no sense to use the four-hour period, since the second definition of the leader would occur only by the end of the trading session. This group of tests aimed to verify whether the algorithm could profit from the advantage of the leader in the short term. By changing the leader, the algorithm theoretically believes that the advantage of the previous leader is over.

Tests were conducted for all weekdays from the 5th to the 23rd of January 2009, using the three different periods for each test group (with static and with dynamic definition of leader). These dates were chosen due to the available data provided by BM&F, being the most liquid and recent between the available data set.

The criterion for evaluating the performance of the ADAM was the relative result function compared to the result function of the other market participants in the same period of analysis. The result does not consider the transaction costs for any of the participants (although in practice, they are not equal). Because of the using of real data, the simulation considers all the market participants, what accounts for about 60 brokerages (we take each set of clients as one participant).

6. Results

In Figure 5, it is possible to observe the results (by the end of the session) of the ADAM in the first group of tests (with static leader), using periods of one hour (white), two hours (grey) and four hours (black).

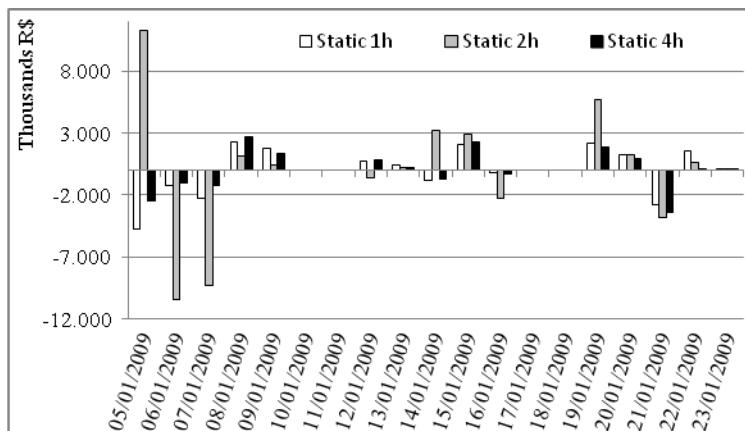


Figure 5. Results from the tests with static definition of leader

With the tests made, it was possible to observe that the results using the two-hour period were more volatile than the others. In other words, the standard deviation of the results in this period was higher than the standard deviations of the results with periods of one and four hours. This means that for the study period, the price movements were more concentrated near two hours intervals.

Moreover, it is possible to observe that, on most days, the results using different periods were all positive or all negative and they seldom happened to have opposite signs. That is, in general, the results of the leaders of each day kept the same trend throughout the day.

As a criterion for measuring the performance of the algorithm, we compared its results with the other market participants during the study period, as a ranking. Tables 3, 4 and 5 show the comparison between the first group of tests' results:

Table 3. Performance for One-Hour Period with Static Leader

Ranking	Agent	Result
1	BM000188	+ 33.401.100
26	ADAM	+ 624.375
65	BM000072	- 34.093.725

Table 4. Performance for Two-Hour Period with Static Leader

Ranking	Agent	Result
1	BM000188	+ 33.401.100
29	ADAM	+ 454.400
65	BM000072	- 34.093.725

Table 5. Performance for Four-Hour Period with Static Leader

Ranking	Agent	Result
1	BM000188	+ 33.401.100
26	ADAM	+ 1.128.850
65	BM000072	- 34.093.725

Similarly, Figure 6 shows the chart with the results of the second group of tests (with dynamic leader) using half-hour (white), one-hour (grey) and two-hour (black) periods:

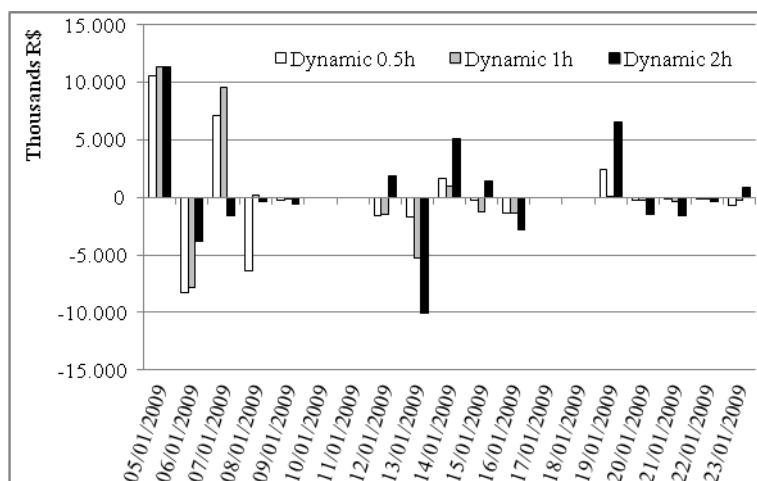


Figure 6. Results from the tests with dynamic definition of leader

The results obtained from the tests with different periods are not as dissimilar as those obtained with static definition of leader. That is, using the dynamic definition, we obtained more consistent results and relatively less volatile than the ones from the first group. In the same line, tables 6, 7 and 8 represent the performance of the algorithm for this group of tests:

Table 6. Performance for Half-Hour Period with Dynamic Leader

Ranking	Agent	Result
1	BM000188	+ 33.401.100
26	ADAM	+ 736.875
65	BM000072	- 34.093.725

Table 7. Performance for One-Hour Period with Dynamic Leader

Ranking	Agent	Result
1	BM000188	+ 33.401.100
17	ADAM	+ 3.544.375
65	BM000072	- 34.093.725

Table 8. Performance for Two-Hour Period with Dynamic Leader

Ranking	Agent	Result
1	BM000188	+ 33.401.100
15	ADAM	+ 4.499.550
65	BM000072	- 34.093.725

7. Conclusion

Given the content exposed, we conclude that using the static definition of leader, the algorithm performance was reasonable compared to the other market participants, with positive results in all three sets of tests. However, we can see that those results do not even have a significant order of magnitude. Although it performed well, it would be convenient to make an analysis with more extensive samples of data in order to reach a more reliable conclusion, because such results may be biased by the historical data used. If we had obtained more consistent results, we could affirm categorically that the algorithm's performance is satisfactory.

In contrast, the use of dynamic definition of leader showed much higher performance. The evaluation periods with the best results were, respectively, of two and one hour.

Because it is very likely that, in each trading day, the leader is not the same as the previous day, ideally the strategy would take advantage of some of the gains of the leader of the day. Therefore, in the long term, ADAM's performance would probably be among the best between all the market participants, if not the best one.

The implementation happened to be a very useful tool because it allows the test of any algorithm with real market data, since the latter communicates with the system through well-defined messages. It also allows in a simple way to convert real FIX messages of the market in registers in a database to be used for testing.

In a more sophisticated analysis of the algorithm's details, we notice that it can be improved in several aspects. For instance, in general, the price paid by the algorithm to match its position to the leader's is always equal to or slightly worse than the price paid by the leader in a trade. This kind of detail observed in high frequency eventually becomes relevant when many operations are performed on the same day. In this case, it is appropriate to refine the model to minimize the influence of this issue.

Possible studies that may be performed include the introduction of rules for entering orders into the market. One example of rule could be to consider the sums of the quantities of orders on the bid and ask books to try to establish a greater probability of the next trades be done in the bid price or in the ask price

Moreover, as the leader in most cases is a brokerage house with a large trade volume, it would be interesting to create filters to define the leader. For example, excluding the brokers with the greatest volume traded, as well the brokers whose result is more volatile, so the algorithm will be more secure, given the limitations of the investment capacity.

In practice, ADAM appears to be relatively simple to be built with different technologies, given a platform (or framework) capable of handling events at high frequency. Today, there are several companies that offer this type of product, such as frameworks, in which the algorithms can be customized according to the operating agent that one wants to create. This strategy, in particular the way it was conceived, is very restrictive to use for trading in the Brazilian market due to the large financial volume involved. For small investors, it would be ideal to customize the strategy for lower volumes to make it viable, according to their investment power, as well considering transaction costs when carrying out back-tests.

References

- De Castro, P. A. L. (2009) "Uma Arquitetura para Administração Automatizada de Ativos Baseada em Agentes Competitivos", Tese de Doutorado, Escola Politécnica da Universidade de São Paulo.
- Dacorogna, M. M., Gençay, R., Müller, U. A., Olsen, R. B. and Pictet, O. V. (2001) "Introduction to High Frequency Finance", Academic Press.
- Deitel, H. M. and Deitel, P. J. (2005), "Java: How To Program", Pearson Prentice Hall, 6th edition.
- FIX Protocol. (2010). "Technical Resources / Specifications", <http://www.fixprotocol.org/specifications>.
- Bolsa de Mercadorias e Futuros, "BELL – BM&FBOVESPA Electronic Link. Regras de Uso FIX (Derivativos, Câmbio, Renda Fixa)", Department of Trading Systems and Market Data, BM&F.

Gerenciamento de Agentes Remotos no Projeto Subverse via Scripts embutidos em Linguagem LUA

Saulo Popov Zambiasi¹

¹Sistemas de Informação / Ciência da Computação
UNISUL – Universidade do Sul de Santa Catarina – Florianópolis – SC – Brasil
{saulopz@gmail.com}

Abstract. *Subverse Project is a learning platform in constantly evolution. It is focused to Computer Science courses and it is to work with a set of theories presented in university disciplines. The Subverse is presented as a virtual world to implementation of agents based in paradigm of games multi-players. However, when it is necessary some update in behavior of an agent, a recompilation and a new connection with Subverse virtual world is obligate. An alternative way is the utilization of scripts that can be used by the agents. These scripts has the advantage that can be updated in execution time. A script can be read by an agent and executed in conformity with its specifications. In that context, this paper present a propose for management of behaviors of remote agents in the Subverse Project via scripts, more specifically in the programming language Lua.*

Resumo. *O Projeto Subverse é uma plataforma de aprendizado em constante evolução, para que alunos de cursos de computação possam trabalhar na prática diversas teorias apresentadas nas disciplinas. Este se apresenta como um mundo virtual para a implementação de agentes baseados no paradigma de jogos multijogador. Contudo, quando há a necessidade de alteração da forma como um Agente se comporta no mundo virtual, é necessário desconectá-lo do mundo virtual, de uma recompilação e uma nova conexão ao mundo virtual. Uma forma alternativa é a utilização de scripts que possam ser alterados em tempo de execução, lidos pelo Agente e executados em conformidade com suas especificações. Neste contexto, este artigo apresenta uma proposta de gerenciamento dos comportamentos dos Agentes remotos no Projeto Subverse via scripts, mais especificamente escritos em linguagem de programação Lua.*

1. Introdução

O Projeto Subverse, ou simplesmente Subverse, é um mundo virtual para a implementação de agentes baseado no paradigma de jogos multijogador. Este tem como objetivo servir como plataforma para o ensino universitário em cursos de computação [Benin, 2007], [Silva, 2008]. Por ser um ambiente que permite a multidisciplinaridade, este ambiente de aprendizado se apropria de várias disciplinas, tais como: computação gráfica, para a visualização do mundo virtual e dos agentes inseridos neste; técnicas de desenvolvimentos de jogos de computador; sistemas distribuídos com múltiplos agentes

conectados ao ambiente remotamente; Inteligência Artificial com a implementação de algoritmos inteligentes nos agentes que devem interagir no mundo virtual; entre outras [Zambiasi et al, 2012].

A participação no processo para o aprendizado proporciona ao aluno desafios reais. Este passa a não apenas adquirir conhecimentos teóricos e de utilização de softwares prontos, mas também passa ter o entendimento da estrutura, organização e demais elementos pelo processo da prática de desenvolvimento e da evolução do sistema em si. Assim, o Projeto Subverse se mostra como uma maneira de apresentar ao aluno esta possibilidade de participarem de todo o processo do desenvolvimento do sistema. Além disso, o sistema pronto objetiva oferecer também uma plataforma de desenvolvimento de algoritmos para aprendizado de diversas disciplinas.

Um Agente no mundo virtual do Subverse pode executar seus comportamentos como forma de atingir seus objetivos. Contudo, se algo deve ser alterado nesse comportamento, para que o agente possa ser melhorado ou modificado, é necessário parar sua execução, executar uma nova compilação no código fonte e então novamente carregá-lo no mundo virtual. Uma forma que poderia ser interessante para melhorar esse procedimento, seria a modificação da forma como o Agente age em tempo de execução. A utilização de *scripts*, conforme citado por Celes et al. (2011), poderia ser uma forma de se resolver este problema, alterando os *scripts* sem a necessidade de parar a execução do agente ou de nova recompilação.

Dessa forma, este artigo propõe uma forma alternativa de gerenciamento dos agentes, conectados remotamente ao Subverse, via utilização de *scripts*. No caso específico dessa proposta, é utilizada a linguagem de programação Lua.

O presente artigo está organizado da seguinte forma: Na seção 2 é apresentado o Projeto Subverse e os assuntos envolvidos em sua estrutura de execução atual. A seção 3 apresenta a proposta do gerenciamento via *scripts*. Na seção 4 são apresentadas as considerações finais.

2. Projeto Subverse

O Projeto Subverse se caracteriza por um conjunto de implementações que se mantêm em constante evolução. Módulos não desenvolvidos se mostram como fonte de pesquisa para a implementação de tecnologias atuais e módulos já prontos se mostram como estruturas que podem ser estudadas, melhoradas e adaptadas. Ainda, na forma de uma implementação de um mundo virtual que segue a filosofia dos jogos multijogadores, o Projeto Subverse fornece uma estrutura para conexões de múltiplos agentes para interagirem entre eles. Estes agentes são, por si, estruturas algorítmicas que podem ser implementadas como forma de aplicação de teorias de diversas disciplinas da computação. Contudo, para tal, o Projeto Subverse segue a filosofia dos sistemas *opensource*, tão conhecidos e disseminados na Internet hoje em dia. Ainda, para a implementação, sugere-se que sejam utilizadas também tecnologias *opensource*, gratuitas e atuais disponíveis.

No Subverse, diversos agentes podem ser implementados utilizando teorias de inteligência artificial, sistemas multiagentes, redes neurais artificiais, lógica, lógica difusa, sistemas especialistas e outras, a fim de estudo e aperfeiçoamento dos comportamentos destes [Benin, 2007]. Os agentes, especificamente, se caracterizam como algo que, provido de sensores, recebe informações do ambiente ao seu redor e toma ações, interagindo com este ambiente por meio de mecanismos denominados de atuadores.

Além disso, os agentes devem poder se comunicar com outros para poderem alcançar seus objetivos [Russel e Norvig, 2004]. No caso dos agentes do Subverse, estes são agentes de software e seus sensores e atuadores estão no contexto de um mundo virtual criado para a interação entre os agentes.

Um mundo virtual pode ser visto como uma arquitetura que suporta interações em tempo real em um mundo virtual. Esta arquitetura deve ser flexível o suficiente para suportar as tarefas que se deseja executar [Calvin et al., 1993]. Uma outra perspectiva de um mundo virtual é a de um ambiente artificial criado em um computador que contém especificidades mínimas de um mundo real. Estes sistemas são multiusuários, permitindo que muitas pessoas interajam simultaneamente. Os mundos virtuais são usados hoje em dia para diversas aplicações, tais como salas de conversa na Internet, vídeo conferência, jogos multijogadores, etc. Essas aplicações requerem que a interação seja feita em tempo real e intentam ser o mais realista possível [Pulo e Houle, 2001].

O Subverse pode ser comparado, em um certo ponto de vista, a jogos multijogadores. Estes geralmente utilizam uma arquitetura de comunicação cliente-servidor. Os clientes enviam mensagens para um servidor responsável pelo processamento do jogo e recebem mensagens processadas para mostrar em uma interface gráfica ao usuário. Para isso, um *middleware* pode promover certa modularidade de programação, facilitando o desenvolvimento de aplicações distribuídas, uma vez que o desenvolvedor não precisa se preocupar em escrever o código para gerenciar interações entre os processos. Entretanto, a comunicação entre o *middleware* e processos incorre em custos adicionais se comparado com a comunicação direta entre os processos [Dickey et al., 2004].

Em tempo, existem mundos virtuais *opensource* na internet, incluindo jogos multijogadores que possibilitam a conexão de agentes, ou *bots*. Porém, a documentação é em geral fraca e o tempo de aprendizado do sistema seria ampliado devido a necessidade da leitura de diversas linhas de código prontas. Isso, na maioria das vezes desestimula os novos desenvolvedores a se unirem ao projeto ou mesmo a alunos que estão em fase de aprendizado se arriscarem nesses sistemas. Existem também trabalhos, como o de Adobbati e Marshall (2001), já citado, que apresentam uma proposta bastante correlata com a apresentada aqui. Neste, é apresentada uma estrutura para que agentes possam se conectar à um mundo virtual 3d de um jogo multijogador chamado *Unreal Tournament* para interagirem na forma de um sistema multiagente. Porém, como já explicado, o objetivo do Projeto Subverse é possibilitar aos estudantes a participação em todo o processo de desenvolvimento e evolução do sistema como um todo.

2.1. Arquitetura

Algumas características são necessárias para que os elementos ativos, agentes, do mundo virtual possam interagir com os outros elementos e com o mundo virtual, propriamente dito. Essas compõem o modelo da arquitetura, composto por sistemas e módulos organizados conforme o paradigma cliente-servidor, conforme apresentada na Figura 1.

Os elementos da arquitetura são o Navegador Web, a Interface Visual e os Agentes, do lado do Cliente, e do lado do Servidor estão localizados um Banco de Dados, o Servidor Web, o Mundo Virtual e o Avatar [Zambiasi et al, 2009].

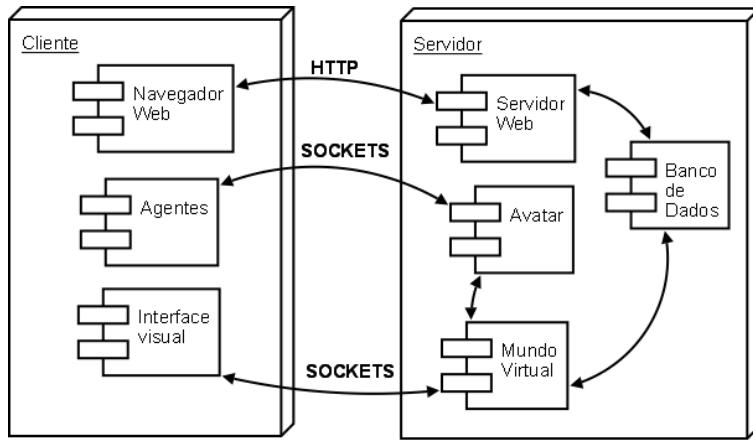


Figura 1: Arquitetura do Subverse.

O **Servidor Web** torna possível o acesso dos usuários às configurações e criação dos agentes que interagem no mundo virtual por meio de uma aplicação Web de forma bastante interativa. É necessário uma interface para permitir que o usuário acesse de qualquer lugar e a qualquer momento essas informações, tal como uma interface Web.

O **Banco de Dados** se apresenta como uma área para o armazenamento das informações de cada agente de forma persistente. Isto permite que o agente não perca suas informações, características e histórico quando ele está desconectado do mundo virtual ou adormecido.

O **Mundo Virtual**, propriamente dito, é a base de execução dos elementos ativos no mundo virtual. Este é responsável pela leitura das informações persistentes quando esses são instanciados, pela alocação de um avatar para cada agente que se conecta e pela interação entre os elementos do mundo virtual. Este é caracterizado como o núcleo de processamento e comunicação dos agentes.

O **Avatar** é o proxy do agente no mundo virtual, ou seja, um módulo de *software* instanciado pelo mundo virtual que serve para representar cada agente neste. Este faz basicamente o trabalho de comunicação entre o agente e o mundo virtual, representando as ações do agente no mundo virtual e enviando os retornos dessas ações ao agente.

O **Navegador Web** provê um meio de acesso para a criação de novos agentes e para a configuração das informações dos agentes via Web. Por definição deste projeto, apenas agentes cadastrados podem acessar o mundo virtual, como forma de segurança, maneira de manter as informações persistentes desses no sistema e para o registro de ações (*logs*) dos agentes para possíveis estatísticas e análises de execução, tanto para os desenvolvedores do sistema, como para os usuários criadores e implementadores de agentes.

A **Interface Visual** é um ambiente gráfico baseado em jogos de computadores para fornecer aos usuários uma forma de observar visualmente os agentes em execução no mundo virtual. Neste projeto, uma primeira versão deste módulo segue as seguintes formas de visualização: modo Avatar (é criado um agente para representar o usuário no mundo virtual, este pode interagir com os outros agentes como se fosse um agente); modo Deus (apenas navega no sistema, com as setas direcionais, podendo visualizar o ambiente enquanto este se encontra em funcionamento) e modo Agente (acompanhando um agente específico enquanto este interage, por sua própria programação, com o

mundo virtual e outros agentes).

Os **Agentes**, por fim, são os elementos de *software* que, via um meio de comunicação remoto com o seu representante no mundo virtual, o Avatar, interagem com o mundo virtual e com outros agentes. A programação das técnicas de Inteligência Artificial podem ser implementadas neste módulo. Deve ser possível que cada usuário do mundo virtual possa criar diversos agentes, mas para cada agente é necessário um cadastro próprio e um avatar para representá-lo no mundo virtual.

2.2. *Middleware de Comunicação*

O Projeto Subverse sugere uma estrutura distribuída para múltiplas conexões de agentes remotos para interação entre eles no mundo virtual do Subverse [Zambiasi et al. 2009], tal como um MMOG (*Massive Multiplayer Online Game*). Contudo, no lugar de jogadores controlando avatares remotamente, os elementos que devem controlar os avatares são agentes desenvolvidos pelos usuários. Os usuários iniciam seus agentes e visualizam seu ciclo de vida no jogo por meio de uma interface gráfica que também se conecta remotamente ao jogo.

Em jogos MMOG, os clientes são utilizados pelos jogadores para se conectar ao servidor do jogo. Dessa forma, o jogador pode enviar informações da ação que deseja realizar e recebe informações atualizadas das atividades que estão ocorrendo à volta do seu personagem (avatar) no jogo. O servidor possui o estado geral do mundo do jogo e coordena a atividade dos jogadores [Balan et al. 2005]. Muitos desses jogos trabalham baseados no paradigma cliente-servidor. Neste estilo os clientes trocam mensagens com um servidor, que é a parte responsável pelo processamento do jogo [Dickey et al. 2004]. O servidor, em execução, abre um soquete de comunicação e fica aguardando clientes se conectarem. Um cliente estabelece uma conexão com o servidor por meio de um endereço IP (*Internet Protocol*) e uma porta, solicitando a conexão [Silberschatz et al. 2001].

A utilização de soquetes é uma ótima escolha para se trabalhar com a comunicação de processos por ser um método rápido e simples. Contudo, se por um lado, a sua utilização agiliza o processo da comunicação entre os processos, por outro, dificulta pela necessidade de se tratar as mensagens em um baixo nível. Sendo assim, para facilitar a utilização de soquetes, é interessante a utilização de um *middleware* de comunicação, a fim de promover a modularidade e maior facilidade na programação de sistemas distribuídos. Por meio desse recurso, “o desenvolvedor não precisa se preocupar em escrever o código para gerenciar interações entre os processos” [Silva 2008].

Além de facilitar ao desenvolvedor, os *middlewares* de comunicação buscam fornecer portabilidade, transparência, interoperabilidade aos sistemas distribuídos e ocultam detalhes de implementações [Silva 2008]. Para Coulouris et al. (2001), tal camada busca mascarar a heterogeneidade dos sistemas, fornecendo um conjunto de funções para a troca de mensagens, dispensando-o da preocupação de como esta vai ser enviada ou recebida.

Pelo motivo dos agentes no Subverse trabalharem remotamente, há então a necessidade de um controle de troca de mensagens e uma forma de permitir múltiplas conexões. Desse modo, para facilitar o desenvolvimento desses agentes, sem que o desenvolvedor precise se preocupar com os detalhes da comunicação, este artigo apresenta uma proposta de *middleware* de comunicação para o Subverse. A proposta é

baseada no paradigma cliente-servidor, e conexões via soquetes, com troca de mensagens. As mensagens são as representações das chamadas de métodos com uma estrutura de variáveis, com os nomes, tipos e valores, além do nome do método que está sendo chamado.

De forma a facilitar o desenvolvimento do Subverse na comunicação entre o cliente e o servidor via soquetes, surgiu a necessidade do desenvolvimento de um *middleware* de comunicação. Assim, tendo como inspiração a forma de comunicação do RPC (*Remote Procedure Call*), visto em Couloris et all. (2001), foi feita uma implementação de um *middleware* que fornece ao desenvolvedor um conjunto de métodos que, para ele, é como se estivesse acessando diretamente os métodos no próprio servidor. Esses métodos, são implementados na forma de criar uma mensagem com as informações do tipo do método que está sendo chamado e o conjunto de variáveis que estão sendo enviadas.

Conforme visto na Figura 2, o agente acessa o MAPA (Métodos de Acesso ao *Proxy* do Agente). Por sua vez, o *Proxy* do Agente se utiliza do MAA (Métodos de Acesso ao Agente) para se comunicar com o Agente. Da mesma forma se dá no lado do Visualizador com o MAPV (Métodos de Acesso ao *Proxy* do Visualizador) e o MAV (Métodos de Acesso ao Visualizador).

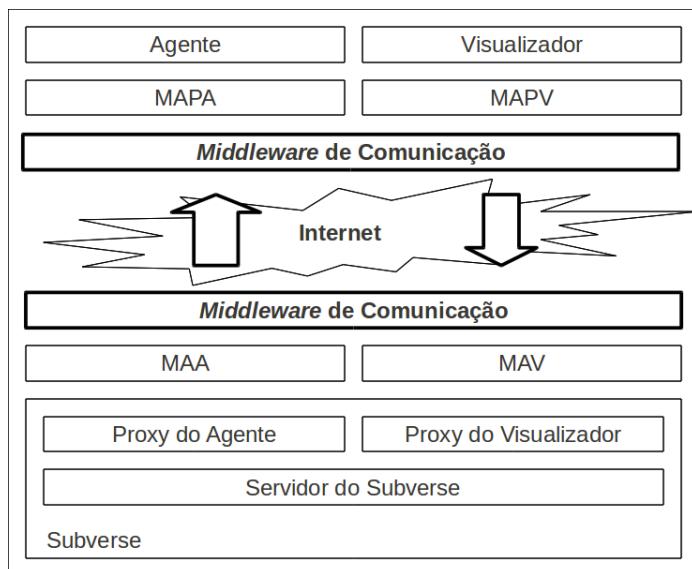


Figura 2: Middleware de comunicação

O conjunto de métodos MAPA, MAPV, MAA e MAV, utilizam o *middleware* de comunicação do Subverse diretamente, criando mensagens com o nome do método a ser chamado e as variáveis envolvidas na chamada do método, tanto de entrada como de saída (retorno).

As chamadas dos métodos, no Subverse, se dão na forma de comunicação assíncrona, ou seja, um método é chamado para efetuar uma determinada operação e outros métodos são utilizados para saber o status da chamada desse método. Por exemplo, se um agente quiser fazer uma movimentação para frente, um método “mover” é chamado com o valor “frente” para a variável “direção”. O desenvolvedor deve então utilizar o método “moveu” que retorna null, false ou true para saber se a operação foi executada com sucesso. Caso o retorno for null, então ainda não foi retornada uma resposta, true caso conseguiu mover e false caso a operação não foi possível.

Por sua vez, o *middleware* de comunicação do Subverse (Figura 3) é composto de alguns módulos principais. Os Métodos de Criação e Extração de Variáveis são responsáveis por criar uma variável com nome, tipo e valor, e métodos correspondentes para recuperar os nomes, tipos e valores das variáveis. Uma mensagem, que é a informação trocada entre o cliente e o servidor, é formada por uma ou mais variáveis, formando uma estrutura de variáveis. Quando o usuário termina a criação de uma estrutura, ele utiliza o método de empacotamento dessa estrutura, criando uma mensagem e enviando-a para a Fila de Mensagens para Envio. O Transmissor fica responsável por enviar essa mensagem ao servidor, quando possível. Cada mensagem possui um código, de forma a poder interpretar de qual requisição um determinado retorno se refere.

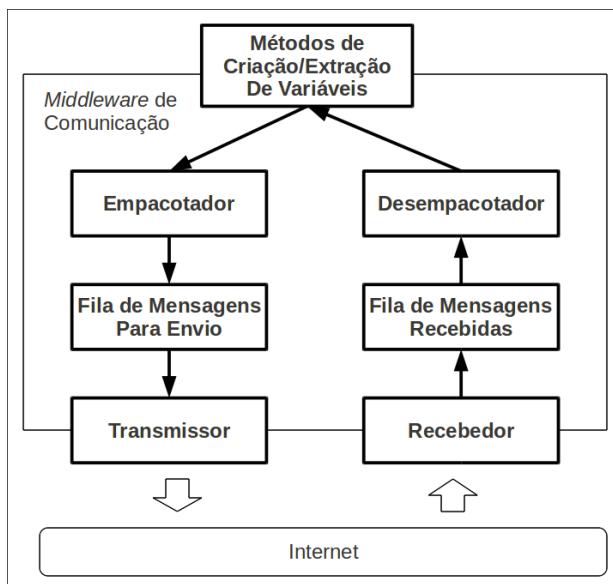


Figura 3: middleware de comunicação

O recebimento de uma mensagem inicia o processo contrário. Quando o recebedor pega uma mensagem, a coloca em uma Fila de Mensagens Recebidas, que pode ser desempacotada e interpretada, para efetuar a chamada da qual a mensagem é responsável para executar. Todo o processo de troca de mensagens funciona na forma de multitarefa, deixando tanto o cliente quanto o servidor livres para efetuarem as tarefas de processamento das quais são responsáveis, não necessitando o usuário desenvolvedor ficar preocupado com o processo de troca de mensagem. Além disso, o *middleware* do servidor aceita conexões de múltiplos clientes.

2.3. Implementação Embutida

Segundo Celes et al. (2011), a maioria dos jogos se utilizam do auxílio de alguma linguagem de script embutida no software. Essas linguagens normalmente são interpretadas, com tipagem dinâmica e gerenciamento automático de memória. A construção de estruturas de informações são bastante dinâmicas, fornecendo aos usuários uma forma interessante de ampliar as possibilidades do software em tempo de execução e sem a necessidade de uma nova compilação do código do software implementado.

O funcionamento dessa ideia se dá por meio da implementação de um script em um programa hospedeiro, desenvolvido em uma linguagem de programação não

necessariamente igual ao programa em si (por exemplo, o *script* em linguagem LUA e o programa em C++). Uma outra característica, que somada com a anterior torna a implementação na forma de *scripts* uma excelente ferramenta de desenvolvimento, é que essas linguagens também não devem poder acessar execuções não autorizadas no programa hospedeiro (Celes et al., 2011). Várias vantagens podem ser identificadas pela utilização de uma linguagem de *script* acoplada em um software (normalmente utilizadas em jogos de computador). O *script* pode ser utilizado para definir configurações, objetos e seus comportamentos, gerenciar algoritmos de inteligência artificial e tratar os eventos de entrada (Celes et al., 2011).

Em Celes et al. (2011), a linguagem Lua é apresentada como uma linguagem de *script* bastante atual e como uma das mais utilizadas no desenvolvimento de jogos. Isso devido “ao seu pequeno tamanho, bom desempenho, portabilidade e facilidade de integração”. Essa linguagem é extensível e projetada para oferecer metamecanismos que facilitam a adequação da linguagem às necessidades da aplicação.

3. Proposta

A proposta desse trabalho se firma na forma como ocorre a execução dos comportamentos do Agente. Em sua base, deseja-se evitar a recompilação do agente caso seja necessário a alteração do comportamento do mesmo, i.e. da maneira como este deve agir no mundo virtual. Em tempo, também é interessante que esses mesmos comportamentos possam ser modificados em tempo de execução do agente.

Um *Script* desenvolvido em linguagem de programação Lua se caracteriza como um arquivo texto contendo um conjunto de especificações algorítmicas para serem seguidas por um interpretador. No caso da proposta, cada um desses *scripts* é visto como um comportamento para o Agente. O agente possui um repositório desses *scripts* que podem ser acessados por um “Seletor de Comportamentos” que fica localizado no programa do Agente. Quando um comportamento é selecionado, este entra em execução para satisfazer as necessidades do Agente e para alcançar seus objetivos. Na Figura 4 pode ser vista uma representação gráfica da utilização de *scripts* em linguagem de programação Lua pelo Agente.

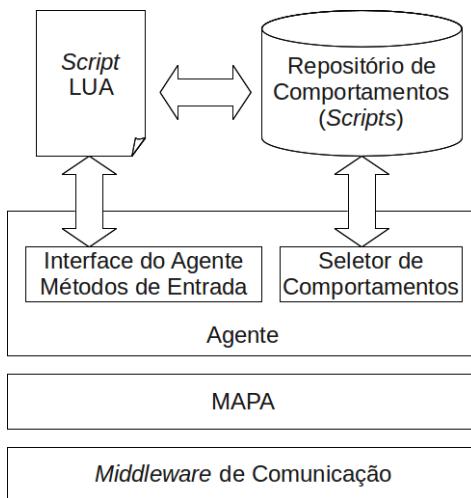


Figura 4: Comportamentos do agente via scripts.

O módulo de “Interface do Agente/Métodos de Entrada” da Figura 4 é classe do Agente com um conjunto de métodos públicos que o programador pode utilizar para

compor o comportamento do seu Agente. Contudo, essa classe está localizada no dispositivo computacional do cliente e o comportamento do Agente torna-se efetivo no mundo virtual apenas do lado do servidor. Essa classe, do lado do cliente, apenas faz o mapeamento dos métodos para um conjunto de mensagens que são enviadas para o servidor. Do lado do servidor, essas mensagens são recebidas e transformadas em chamada de métodos na classe do Avatar (sessão 2.1.), executadas e, se for o caso, um retorno é enviado de volta ao cliente. Todo esse processo é feito pelo MAPA e pelo *Middleware* de Comunicação visto na sessão 2.2..

Ainda, do lado do cliente, é feito o mapeamento para a linguagem Lua por meio dos recursos de biblioteca. Como exemplo, toma-se a classe de Agente da Figura 5 com parte dos métodos do Agente.

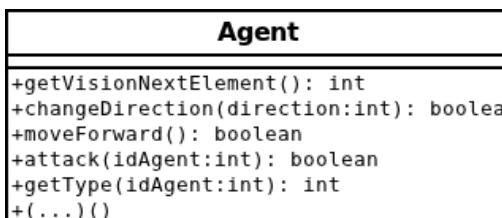


Figura 5: Representação da classe do Agente.

Para a execução do *script* criado pelo usuário, uma função (neste caso em linguagem de programação C++) é chamada passando como parâmetro o *script* na forma de uma *string* – pode ser um arquivo carregado em uma *string* ou informações texto de um banco de dados. Essa função (Figura 6) efetua o mapeamento dos métodos da classe do Agente para a linguagem Lua e executa o *script* do Agente criado pelo usuário.

```
1 #include "Agent.h"
2
3 void Lua_run(string script) {
4     lua_State *L = lua_open();
5     luaL_openlibs(L);
6     open(L);
7     module(L)[
8         class_<Agent>("Agent")
9             .def(constructor<>())
10            .def("getVisionNextElement", &Agent::getVisionNextElement)
11            .def("changeDirection", &Agent::changeDirection)
12            .def("moveForward", &Agent::moveForward)
13            .def("attack", &Agent::attack)
14            .def("getType", &Agent::getType)
15            // other methods of Agent
16    ];
17    if (luaL_dofile(L, script.c_str())) {
18        cerr << lua_tostring(L, -1) << endl;
19    }
20    lua_close(L);
21 }
```

Figura 6: Mapeamento da classe do Agente para o script Lua.

Na Figura 7 é possível visualizar um exemplo de um comportamento simples de um agente escrito em *script* Lua e baseado na estrutura parcial do Agente aqui apresentada. Este instancia um objeto no programa do Agente em execução com o nome “Agent” e pode efetuar chamada aos seus métodos para fazer o Agente agir no mundo virtual do Subverse.

```
1 -- Simple Agent Lua
2 require "globals"
3
4 a = Agent()
5 idElement = a:getVisionNextElement()
6 if idElement ~= 0 then
7     if a:getType(idElement) == MONSTER then
8         a:attack(idElement)
9     end
10 end
11 a:changeDirection(LEFT)
12 a:moveForward()
```

Figura 7: Exemplo de um comportamento simples.

No caso do exemplo de *script* (Figura 7), um arquivo de nome *globals.lua* é incluído. Este contém algumas constantes que são utilizadas pelo *script* (por exemplo: *MONSTER*, *LEFT*, *RIGHT*, etc.). Na linha 4, a classe do Agente, já instanciada no programa em C++ em execução é acessada por meio de uma variável chamada *a*. Na linha 5, o *script* utiliza um método da classe do Agente que retorna o próximo elemento que está na visão do Agente. Esse método retorna 0 se não houver mais nenhum elemento ou um inteiro positivo com o *id* do elemento visualizado. Seguindo o algoritmo do *script*, se o *id* do elemento do elemento seja diferente de 0, verifica se o elemento é do tipo *MONSTER*. Caso positivo, efetua um ataque a este elemento. Nas linhas 11 e 12 o Agente vira para a esquerda e dá um passo a frente.

A cada ciclo de execução de um *script*, o usuário pode executar apenas um conjunto de atividades. Por exemplo, um Agente não pode efetuar dois passos para frente num mesmo ciclo de execução. Comparativamente, é como pessoas em um jogo de cartas. Cada pessoa pode, em sua vez, efetuar apenas um conjunto de ações. A vez é passada para os próximos jogadores até que a pessoa possa jogar novamente.

4. Considerações Finais

Este artigo apresentou uma proposta para o gerenciamento dos Agentes que estão conectados ao mundo virtual do Projeto Subverse na forma de *scripts*. A linguagem de programação no contexto de *scripts* sugerida foi a linguagem Lua e a proposta se manteve nas bases de toda a estrutura e arquitetura do Projeto Subverse já existente.

Um protótipo foi implementado apenas para os testes da utilização de *scripts* em objetos/agentes. As chamadas dos métodos se deram de maneira correta com o que era esperado. Também foi possível alterar o comportamento do agente em tempo de execução pela alteração do arquivo de *script* escrito em linguagem de programação Lua. Contudo, ainda não foi implementada essa forma de gerenciamento dos comportamentos dos agentes no Projeto Subverse especificamente.

Por fim, este trabalho veio como uma contribuição na estrutura do Projeto Subverse, trazendo mais uma sugestão de implementação dos agentes e gerenciamento dos mesmos de forma alternativa, facilitando o controle, evolução e alteração do comportamento dos agentes em tempo de execução. Os próximos passos se dão em utilizar as especificações utilizadas para a criação do protótipo de testes dessa proposta como base de implementação dos agentes remotos no sistema do Projeto Subverse.

5. Agradecimentos

Este trabalho é parcialmente financiado pela PUIP – Programa Unisul de Incentivo à Pesquisa (<http://www.unisul.br>).

Referências

- Adobbati, R., Marshall, A. N., at all. *Gamebots: a 3d virtual world test-bed for multi-agent research*. Agentes'01. Montreal – Canada. May-June, 2001.
- Balan, R., Ebling, M., Castro, P. and Misra, A.. *Matrix: adaptative middleware for distributed multiplayer games*. Journal Middleware. Springer. 390-400. 2005.
- Benin, M.. **Evolução de NPC's e adversários em jogos de computador usando algoritmos genéticos**. Monografia de graduação. Faculdades Barra do Sul. Florianópolis, SC. 2007.
- Calvin, J., Dickens, A., Gaines, B. *The simnet virtual world architecture*. IEEE, 1993.
- Celes, W. de Figueiredo, L.H. and Ierusalimschy, R.. **A Linguagem Lua e suas Aplicações em Jogos**. Rio de Janeiro, 2004. Disponível em: <<http://www.tecgraf.puc-rio.br/~lhf/ftp/doc/wjogos04.pdf>> Acessado em Ago/2011.
- Coulouris, G., Dollimore, J. and Kindberg, T.. *Distributed system: concepts and design*. Addison-Wesley. 2001.
- Dickey, C.G, Zappala, D. and Lo, V., A *Distributed Architecture for massively multiplayer online games*. ACM NetGames Workshop. 2004.
- Pulo, K., Houle M. E., *Evolution of Virtual World System*. IEEE, 2001.
- Russel, S., Norvig, P., **Inteligência Artificial**. Tradução da segunda edição. Rio de Janeiro: Elsevier, 2004.
- Silberschatz, A., Galvin, P., Gagne, G.. **Sistemas operacionais: conceitos e aplicações**. Editora Campus. 6a.ed. 2001.
- Silva, F.C., 2008. **Middleware de comunicação para jogos multiplayer: um estudo de caso no Projeto Subverse**. Monografia de graduação. Faculdades Barra do Sul. Florianópolis, SC.
- Zambiasi, S.P., Benin, M. and Silva, F.C.. **Projeto Subverse, um mundo virtual baseado em jogos multijogadores como ferramenta de ensino multidisciplinar em cursos de tecnologia da informação**. SCGames. 2009.
- Zambiasi, S.P.; Oberderfer, L.P.Z.B.; Benin, M.R. *Asynchronous Remote Management of Agents for Subverse Project*. In IEEE Latin America Transactions, vol.10, no.1, pg.1289-1294. Jan, 2012.

Parte IV

Artigos aceitos como pôsteres

Um sistema multiagente para a formação e avaliação de grupos sócio-afetivos em ambientes CSCL

Alfredo Costa O. Junior¹, Cícero Costa Quarto¹, Rômulo Martins França², Luís Carlos Costa Fonseca³, Sofiane Labidi⁴

¹Centro de Ciências Tecnológicas – Universidade Estadual do Maranhão (UEMA)
CEP: 65055970 – São Luís – MA – Brasil.

²Núcleo de Educação a Distância – Universidade Federal do Maranhão (UFMA) – São Luís – MA – Brasil.

³Departamento de Informática – Universidade Federal do Maranhão (UFMA) – Codó – MA – Brasil.

⁴Laboratório de Sistemas Inteligentes / PPGEE – Universidade Federal do Maranhão (UFMA) – São Luís – MA – Brasil.

alfredo.coj@gmail.com, cicero@engcomp.uema.br, romulomf@gmail.com,
lccfonseca@gmail.com, sofiane.labidi@globo.com.

Abstract. *CSCL Environments (Computer Supported Collaborative Learning) to provide greater collaboration is a challenge for the design of such environments. In this context, the use of Multiagent System has provided significant advances in collaborative support. Thus, this paper aims to propose a architecture of a socio-affective Multiagent System in aid for training and evaluation of student groups to potentialize collaboration in CSCL environment. To do so, was defined the objectives, capabilities and properties of the SMA, as well as analysis of related work and mathematical inferences to determine the performance of groups.*

Resumo. *Ambientes CSCL (Computer Supported Collaborative Learning) que proporcionem maior colaboração é um desafio para o projeto de tais ambientes. Nesse contexto, o emprego de Sistema Multiagentes tem proporcionado avanços significativos no suporte colaborativo. Assim, esse trabalho tem como objetivo norteador propor uma arquitetura de um Sistema Multiagente sócio-afetivo no auxílio à formação e avaliação de grupos de alunos a fim de potencializar a colaboração em ambiente CSCL. Para isso, definiram-se os objetivos, capacidades e propriedades do SMA, além da análise de trabalhos relacionados e de inferências matemáticas para determinação de desempenho de grupos.*

1. Introdução

O advento da Internet e o avanço na utilização de Tecnologias de Informação e Comunicação (TICs) na educação, bem como *Sistemas Tutores Inteligentes* (STI) e teorias de *Inteligência Artificial* (IA) proporcionaram ao processo ensino e aprendizagem um grande salto no que tange a interação entre professores e alunos. Nesse contexto, surgiram as modalidades Educação a Distância (EaD) e a Aprendizagem Colaborativa Apoiada por Computador (tradução do inglês da sigla

CSCL – *Computer Supported Collaborative Learning*), que tem se tornado um grande enfoque e nova tendência do ensino a distância (Stahl et al., 2006).

Em Ambientes CSCL, Lima et al. (2005) afirma que fatores como *a afetividade, os laços de afinidade, a definição dos grupos e a interação entre os grupos, assim como o ambiente computacional* podem interferir no processo de ensino e aprendizagem colaborativa apoiada por computador.

Conforme Boff (2008), a implementação de ferramentas que viabilizem a formação de grupos a fim de potencializar a colaboração em ambientes CSCL não é tarefa fácil. Isso porque ainda segundo a autora é importante que esses sistemas considerem aspectos individuais de alunos, tais como: os estados sócio-afetivos dos mesmos.

Uma alternativa bastante relevante na solução de problemas de colaboração são os ambientes virtuais de aprendizagem associados à tecnologia orientada a agentes, a qual pode auxiliar a interação em grupos de trabalho, pois proporciona ao aluno a oportunidade de ter um ensino individualizado e adaptável (Boff, 2008). Desse modo, nota-se que propiciar um ambiente CSCL adequado é uma tarefa complexa e abrangente, entretanto, segundo Boff (2008) e Silveira (2006), as ferramentas de Inteligência Artificial orientadas a agentes se apresentam como soluções eficazes na construção desses ambientes.

Para tal, esse trabalho propõe a definição de uma arquitetura de Sistema Multiagente constituída pelos agentes FORMADOR DE GRUPOS e AVALIADOR DE GRUPOS, a fim de auxiliar a aprendizagem colaborativa apoiada por computador através da formação e aprimoramento de grupos sócio afetivos.

2. Fundamentação Teórica

Segundo Vygotsky (1998), em ambientes de aprendizagem em grupo é extremamente importante levar em consideração fatores sócio afetivos para que haja maior colaboração entre os aprendizes.

Assim, dentre os diversos ambientes de trabalho em grupos, destacam-se os ambientes CSCL (*Computer Supported Collaborative Learning* – Aprendizagem Colaborativa Apoiada por Computador), por proporcionar a união de suporte computacional e ambiente colaborativo (Stahl et al., 2006). De acordo com Lima et al. (2005), a representação dos aprendizes e seus grupos, a interação entre os aprendizes e professores e a distribuição dos aprendizes em grupos são questões que devem ser consideradas em projetos de ambientes CSCL.

Atualmente, existem muitas implementações que viabilizam a formação de grupos a fim de potencializar a colaboração em ambientes CSCL (cf. Silveira, 2006 e Boff, 2008). Entretanto, segundo Boff (2008), é importante que esses sistemas computacionais realizem inferência do estado sócio-afetivo do aluno e a reação do aluno ao seu estado sócio-afetivo.

De acordo com Boff (2008), a colaboração em ambientes CSCL pode ser intensificada com a implementação de técnicas de Inteligência Artificial orientadas a agentes. Para Jaques (1999), quando agentes cooperam uns com os outros em determinado ambiente para resolverem um problema caracterizam um Sistema Multiagentes. Ainda conforme Jaques (1999), esses ambientes de sociedade de agentes,

também conhecidos como SMA, são projetados para resolverem diversos tipos de problemas computacionais, inclusive problemas provenientes da interação de grupos em ambientes de EaD.

Silveira (2006) definiu e implementou uma arquitetura SMA, denominada AMIA para auxílio da aprendizagem colaborativa, cujos agentes foram modelados com algoritmos genéticos e seus principais objetivos são: *definir perfis de alunos* considerando características cognitivas e *formar grupos colaborativos*. Boff (2008) propôs um modelo probabilístico de conhecimento e raciocínio para um agente, chamado *Agente Social*, que faz parte do ambiente **AMPLIA** (Ambiente Multiagente Probabilístico Inteligente de Aprendizagem), cujos principais objetivos são: (a) *analisar o perfil dos alunos* considerando aspectos individuais, tais como *estado afetivo, questões psicológicas e cognitivas* e (b) *composição de grupos de trabalho*.

A Tabela 1 apresenta uma comparação entre os sistemas multiagentes desenvolvidos por Silveira (2006), Boff (2008) e a arquitetura multiagentes proposta nessa pesquisa, em que os critérios utilizados são baseados em características definidas por Silveira (2006). Assim, adotaram-se os seguintes critérios para comparar as arquiteturas multiagentes mencionadas: 1) O SMA considera feedback de professor e alunos em relação aos grupos formados. 2) Proporciona ambiente cooperativo/collaborativo. 3) Possui arquitetura multiagentes híbrida. 4) Possui agente pedagógico (com personagens animados). 5) Permite integração com ambientes AVA. 6) Possui modelo de aluno. 7) Implementa ferramentas de avaliação. 8) Agentes possuem emoções e/ou sensações. 9) Possibilita adaptabilidade. 10) Possibilidade de formação de grupos de forma automática pelo ambiente. 11) Critérios utilizados para definição de perfis de alunos.

Tabela 1. Critérios de comparação entre arquiteturas multiagentes.

Sistemas Multiagentes	CRITÉRIOS DE COMPARAÇÃO										
	1	2	3	4	5	6	7	8	9	10	11
AMIA	Sim	Sim	Sim	Não	Sim	Sim	Sim	Não	Sim	Sim	Estilos Cognitivos: <i>Convergência de pensamento, Divergência de pensamento, Holista, Serialista, Reflexivo e Impulsivo.</i>
AMPLIA	Não	Sim	Sim	Não	Sim	Sim	Não	Não	Sim	Sim	Critérios comportamentais: <i>Iniciativas de comunicação e Respostas a iniciativas de comunicação.</i>
Arquitetura Proposta	<u>Sim</u>	Sim	Sim	Não	Sim	Sim	Sim	Não	Sim	Sim	Fatores sócio-afetivos: <i>emotividade, atividade e repercussão</i>

Tanto no trabalho de Silveira (2006) quanto no de Boff (2008) foram abordados o uso de agentes sócio-afetivos no auxílio da aprendizagem colaborativa apoiada por computador. Abordagens como estas, motivaram a desenvolver este trabalho, que propõe considerar fatores, tais como: *personalidade/temperamentos, formação de grupos colaborativos e avaliação dos grupos* na modelagem da arquitetura de um Sistema Multiagente Sócio afetivo.

3. Descrição do SMA

O Sistema Multiagente (SMA) proposto é formado por dois agentes: *agente formador de grupos* e *o agente avaliador de grupos*. Esses agentes serão caracterizados pelas propriedades: *habilidade social, aprendizagem, bem como flexibilidade e autonomia*. *Habilidade social*, pois eles irão interagir com professor e alunos e entre eles mesmos para atingir suas metas. Possuem *Aprendizagem*, pois, à medida que os desempenhos de grupos não forem satisfatórios, os agentes irão reformular os grupos e aprimorá-los de acordo com os critérios de temperamento: *emotividade, atividade e repercussão*. *Flexibilidade*, pois, recebem influência de ações realizadas por agentes humanos, professores e alunos, e de software. E serão *Autônomos*, pois o aprimoramento de grupos depende mais das experiências dos agentes do que do conhecimento embutido no SMA pelo seu projetista.

Em relação à capacidade de resolver problemas, a arquitetura multiagente proposta é classificada como *híbrida*, pois têm características *reativas e proativas*. Ou seja, os agentes são reativos, pois a ação de cada agente depende das ações dos demais, e são proativos, pois as ações dos agentes são orientadas a um objetivo geral que é o de potencializar a aprendizagem colaborativa apoiada por computador, além de poderem prever ou evitar um determinado estado para atingir seus objetivos. O processo de interações entre os agentes de software e os agentes humanos (professor e alunos) é ilustrado na Figura 1.

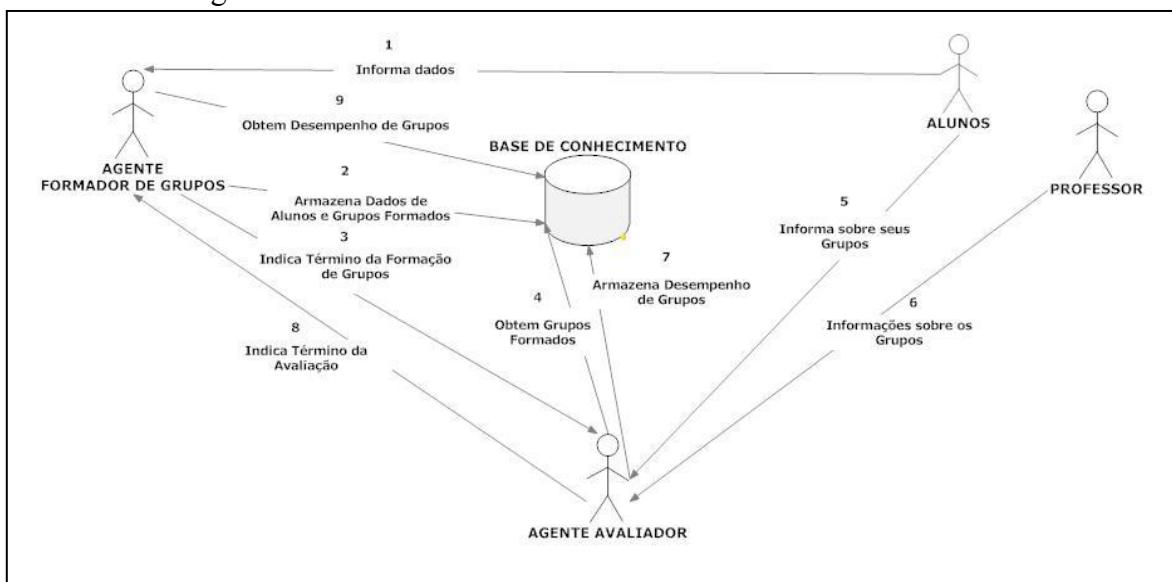


Figura 1. Arquitetura do SMA formado pelos Agentes FORMADOR DE GRUPOS e AVALIADOR DE GRUPOS.

No primeiro momento, os alunos devem responder a um questionário baseado no *Teste de caráter Roger Verdier* que é disponibilizado ao Agente FORMADOR DE GRUPOS, o qual a partir das respostas inferirá o perfil sócio afetivo de alunos, caracterizando-os segundo questões associadas aos temperamentos: *emotividade, atividade e repercussão*. Após isso, o Agente FORMADOR DE GRUPOS armazenará na Base de Conhecimento do SMA os perfis dos alunos e os grupos formados e informará ao Agente AVALIADOR DE GRUPOS o término de suas ações para essa primeira etapa.

O Agente AVALIADOR DE GRUPOS inicia a etapa de avaliação dos grupos de alunos e para isso obtém os clusters de alunos formados acessando a Base de Conhecimento. Para essa etapa, esse agente necessita do *feedback* de professor e alunos (cf. será descrito na seção 3.3), o qual é obtido por meio de um questionário de avaliação de grupo baseado em critérios propostos por Bonals (2003) e em *questionários de avaliação de grupos* propostos por Silveira (2006). De posse destas informações, o Agente AVALIADOR DE GRUPOS realiza o processamento das mesmas e obtém o desempenho de grupos, armazenando-o na Base de Conhecimento do SMA. Após isso, comunica ao Agente FORMADOR DE GRUPOS o término do processo de avaliação.

Essas informações sobre a avaliação são armazenadas também na Base de Conhecimento do SMA. Após saber o término do processo de avaliação, o Agente FORMADOR DE GRUPOS verifica se há algum grupo com desempenho insatisfatório (cf. descrito na seção 3.3). Se isso acontecer, por meio de informações armazenadas anteriormente na Base de Conhecimento do SMA, o Agente FORMADOR DE GRUPOS reformulará os clusters de alunos e então reiniciará o ciclo do SMA até que os grupos formados possuam desempenho satisfatório.

3.1 Agente FORMADOR DE GRUPOS

Segundo Lima et al. (2005), o modo como são distribuídos os aprendizes em grupos pode afetar diretamente os seus desempenhos. Ainda para Lima et al., essa relação, distribuição/desempenho, pode se dar, entre outros fatores, em função do perfil de cada aprendiz e/ou dos laços de afinidade existentes entre eles. Tendo em vista essa abordagem, o agente Formador de Grupos do SMA considerará o perfil/temperamento dos alunos na formação dos grupos. Conforme Stahl et al. (2006); Cunha (2002), o tamanho de grupo deve ser levado em consideração. Ainda de acordo com os autores, grupos pequenos favorecem a interação.

A atividade de formação de clusters do agente FORMADOR DE GRUPOS foi arquitetada de acordo com o algoritmo desenvolvido por Lopes et al. (2011) que se baseia no cubo dos temperamentos utilizado pela caracterologia de René Le Senne (Le Senne, 1963 e Justo, 1966), onde os fatores fundamentais do caráter ou temperamento humano são: *emotividade, atividade e repercussão*. Basicamente, o algoritmo de Lopes et al. (2011) forma grupos de alunos baseados tanto na existência de semelhanças (grupos afins) ou na falta delas (grupos antitéticos) entre suas características sócio afetivas notáveis. É importante destacar que o algoritmo de Lopes et al. (2011) possui resultados bem consolidados, os quais mostram que grupos afins apresentam melhor desempenho que grupos escolhidos de forma aleatória.

Outro ponto importante, na arquitetura SMA proposta é que o perfil/temperamento dos alunos, bem como o desempenho dos grupos, durante o ciclo de vida do grupo será persistido na Base de Conhecimento do SMA. O objetivo de persistir essas informações dos grupos é permitir que o Agente FORMADOR DE GRUPOS melhore as composições já testadas. Assim, o agente proposto nesta seção, estenderá o papel do algoritmo de Lopes et al. (2011) ao levar em consideração a avaliação (definição de desempenho de grupos) realizada pelo agente AVALIADOR DE GRUPOS, que será descrita na seção 3.2, para assim, caso necessário, reformular os clusters e aprender com experiências anteriores.

O Agente FORMADOR DE GRUPOS possui os seguintes papéis: (a) Obter informações dos alunos para definição das características sócio afetivas dos mesmos, conforme os temperamentos *emotividade, atividade e repercussão*. (b) Formar grupos sócio-afetivos. (c) Comunicar ao agente AVALIADOR DE GRUPOS o término do processo de formação de grupos. (d) Após o processo de avaliação, o agente obterá o desempenho de grupos do agente AVALIADOR DE GRUPOS e (e) Verificar se o desempenho será satisfatório ou não (a ser descrita na seção 3.3). Se o desempenho não for satisfatório, então esse agente reformulará os grupos para obtenção de uma melhor colaboração nesse ambiente CSCL. A Figura 2 apresenta o diagrama de Caso de uso do agente FORMADOR DE GRUPOS, em que podem ser observados os papéis desse agente.



Figura 2. Diagrama de Caso de Uso do Agente FORMADOR DE GRUPOS.

3.2 Agente AVALIADOR DE GRUPOS

Na Figura 3 é ilustrado o diagrama de Caso de uso do agente AVALIADOR DE GRUPOS, cujos principais papéis são: (a) obtenção do perfil de alunos e grupos formados através do Agente FORMADOR DE GRUPOS; (b) obtenção da avaliação dos alunos sobre seus grupos e do professor sobre os grupos formados, em que o questionário disponível ao professor e alunos, pelo agente AVALIADOR DE GRUPOS, considera critérios de Bonals (2003) e Silveira (2006). (c) processar os dados com as informações do professor, grupos e alunos; (d) avaliar a colaboração dos grupos; (e) determinar o desempenho para cada grupo; (f) indicar grupo mais colaborativo e (g) comunicar o término do processo de avaliação ao agente FORMADOR DE GRUPOS.

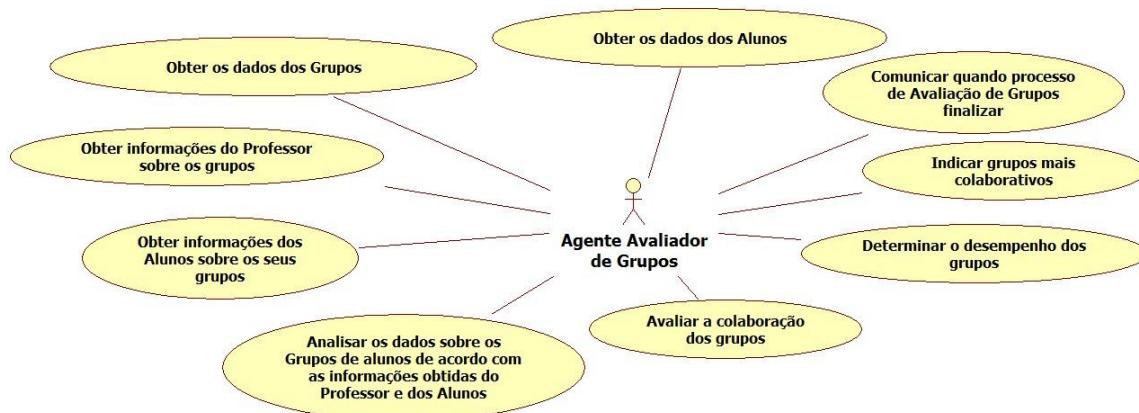


Figura 3. Diagrama de Caso de Uso do Agente AVALIADOR DE GRUPOS.

3.3 Avaliação de desempenho de grupos do SMA

Para o processo de avaliação e desempenho da aprendizagem em grupo, será utilizada uma metodologia, baseada em critérios propostos por Bonals (2003) e em questionários de avaliação de grupos propostos por Silveira (2006). As etapas da metodologia são descritas abaixo:

- (i) Orientar os alunos quanto ao desenvolvimento da atividade de aprendizagem colaborativa (aspectos colaborativos, tempo de realização da atividade, domínio de conhecimento da atividade, entre outros);
- (ii) Definição do tamanho de grupos (de 3 a 5 alunos por grupo);
- (iii) Definição do ambiente CSCL para a realização da atividade de aprendizagem (Moodle ou outro ambiente virtual de aprendizagem, de forma que o professor tenha um controle das frequências dos alunos (login), das quantidades de reuniões e do *feedback* das atividades desenvolvidas pelos alunos);
- (iv) Avaliação de desempenho dos grupos na realização das atividades desenvolvidas. A partir desse momento, o agente AVALIADOR DE GRUPOS obterá os perfis dos alunos e dos grupos formados, bem como as avaliações do professor e a avaliação individual de cada aluno sobre seus grupos de trabalho.

Na Tabela 2, tem-se o questionário de avaliação realizada pelo professor sobre os grupos de trabalho. Essa avaliação leva em consideração quatro níveis qualitativos, os quais são: *Fraco*, *Elementar*, *Bom* e *Ótimo*. Após essa etapa, tem-se a avaliação dos alunos sobre seus respectivos grupos de forma que se tenha um questionário distinto do professor, conforme é mostrada na Tabela 3, a qual considera, também, os mesmos quatro níveis qualitativos citados anteriormente.

Tabela 2. Avaliação do trabalho do grupo (avaliação do professor).

Aspectos	Conceitos
1. Modo como foram divididas as tarefas.	() Ótimo () Bom () Elementar () Fraco
2. Realização de todas as tarefas propostas e/ou justificativas das atividades que não puderam ser realizadas.	() Ótimo () Bom () Elementar () Fraco
3. Participação dos componentes do grupo na apresentação do trabalho.	() Ótimo () Bom () Elementar () Fraco
4. Frequência (não se afastou do grupo durante os trabalhos) em reuniões em ambiente virtual e/ou reuniões presenciais.	() Ótimo () Bom () Elementar () Fraco
5. Qualidade das referências das pesquisas realizadas.	() Ótimo () Bom () Elementar () Fraco
6. Layout da apresentação.	() Ótimo () Bom () Elementar () Fraco
7. Problemas apresentados pelo professor foram resolvidos.	() Ótimo () Bom () Elementar () Fraco
8. Comentários pertinentes durante a apresentação.	() Ótimo () Bom () Elementar () Fraco
9. Respostas adequadas aos questionamentos do professor durante a apresentação.	() Ótimo () Bom () Elementar () Fraco

Tabela 3. Avaliação do trabalho do grupo (avaliação dos membros do grupo).

Aspectos	Conceitos
1. Investigação e recolhimento de informação.	() Ótimo () Bom () Elementar () Fraco
2. Modo como aconteceu o compartilhamento de informação entre membros do grupo.	() Ótimo () Bom () Elementar () Fraco

3. Participação nas discussões.	() Ótimo () Bom () Elementar () Fraco
4. Comprometimento com as tarefas assumidas.	() Ótimo () Bom () Elementar () Fraco
5. Realização de todas as tarefas propostas e/ou justificativas das atividades que não puderam ser realizadas.	() Ótimo () Bom () Elementar () Fraco
6. Participação dos componentes do grupo na apresentação do trabalho.	() Ótimo () Bom () Elementar () Fraco
7. Frequência (não se afastou do grupo durante os trabalhos) em reuniões em ambiente virtual e/ou reuniões presenciais.	() Ótimo () Bom () Elementar () Fraco
8. Soube lidar com as dificuldades apresentadas na pesquisa.	() Ótimo () Bom () Elementar () Fraco
9. Comentários pertinentes durante a apresentação.	() Ótimo () Bom () Elementar () Fraco
10. Respostas adequadas aos questionamentos realizados pelo professor durante a apresentação.	() Ótimo () Bom () Elementar () Fraco

No primeiro momento, o agente AVALIADOR DE GRUPO fornecerá ao professor e alunos uma interface para avaliação de clusters apresentando os questionários descritos nas Tabelas 2 e 3. Em seguida, obterá os resultados dessas avaliações e a partir de seus resultados aplicará um cálculo para determinar quantitativamente o desempenho dos grupos e assim, verificar a necessidade de se reformular ou não os grupos de alunos, com objetivo de alcançar uma maior colaboração. O cálculo do Desempenho de Grupo, realizado pelo Professor, aqui convencionado de $DG_{Professor}$, é mensurado matematicamente pela Equação 1 expressa abaixo:

$$DG_{Professor} = \frac{1 * Qtd_{Fraco} + 2 * Qtd_{Elementar} + 3 * Qtd_{Bom} + 4 * Qtd_{Ótimo}}{Qtd_{Fraco} + Qtd_{Elementar} + Qtd_{Bom} + Qtd_{Ótimo}}$$

Equação 1 - Cálculo sobre a avaliação realizada pelo professor.

Da Equação 1, tem-se uma média ponderada, onde se leva em consideração a quantidade de atribuições destinadas a cada nível qualitativo: *Fraco*, *Elementar*, *Bom* e *Ótimo*; bem como um peso que varia de 1 a 4, respectivamente, para cada um desses níveis.

Em relação à avaliação dos alunos, tem-se, primeiramente, um cálculo idêntico ao cálculo de desempenho de grupo segundo o professor ($DG_{professor}$). Para explicar essa etapa de avaliação de desempenho, considerou-se um grupo hipotético de três alunos. Então, na Equação 2, tem-se um exemplo de cálculo para determinação de Desempenho de Grupo conforme aluno1. Esse mesmo cálculo de avaliação sobre esse cluster de alunos será realizada também pelos alunos2 e aluno3.

$$DG_{Aluno1} = \frac{1 * Qtd_{Fraco} + 2 * Qtd_{Elementar} + 3 * Qtd_{Bom} + 4 * Qtd_{Ótimo}}{Qtd_{Fraco} + Qtd_{Elementar} + Qtd_{Bom} + Qtd_{Ótimo}}$$

Equação 2 - Cálculo de avaliação do aluno1 sobre o grupo hipotético.

A partir de então, é feita uma média ponderada das avaliações feitas pelos alunos, de forma tal que o Líder do grupo tem peso igual a 2, nesse caso aluno1, e os demais membros do grupo tem peso 1. A Equação 3 representa o *Desempenho de Grupo para os alunos* (DG_{Alunos}). O Líder recebe peso dois, pois, segundo Boff (2008), o Líder matem o foco no processo e o funcionamento do grupo, coordenando as discussões, assim tendo uma maior capacidade analítica e de lidar com os problemas (ou conflitos) que o grupo enfrenta.

$$DG_{Alunos} = \frac{DG_{Aluno1} * 2 + DG_{Aluno2} * 1 + DG_{Aluno3} * 1}{2 + 1 + 1}$$

Equação 3 – Exemplo de cálculo de avaliação realizada por todos os membros de um grupo sobre o aluno2.

Finalmente, uma média ponderada do *desempenho de grupo atribuído pelo professor* ($DG_{Professor}$) e do *desempenho de grupo atribuído pelos alunos* (DG_{Alunos}) é obtida de forma que o desempenho atribuído pelo professor tenha peso 2 e pelos alunos peso 1. A escolha do peso 2 para o professor está relacionada com o fato dele ter os papéis, segundo Dias e Leite (2010) de: orientador e mediador, redirecionando o foco e oferecendo opções no apoio ao trabalho colaborativo de alunos. Esse resultado dessa média ponderada representa o *Desempenho de um Grupo considerado pelo agente AVALIADOR DE GRUPO* (DG). A Equação 4, expressa um exemplo de como esse cálculo é obtido.

$$DG = \frac{Av_{professor} * 2 + Av_{Grupo} * 1}{2 + 1}$$

Equação 4 - Desempenho geral de um grupo.

A partir da variável DG , tem-se a nota que representará o desempenho de um grupo. Após o cálculo do desempenho de cada grupo de trabalho e determinação do grupo de melhor desempenho, o agente AVALIADOR DE GRUPO comunicará o término do processo de avaliação ao agente FORMADOR DE GRUPO. Assim, se a nota DG de algum dos grupos for menor que 3, isto é $DG < 3$, o agente FORMADOR DE GRUPOS reformulará os grupos de modo que os novos clusters continuem considerando os caracteres ou temperamentos *emotividade, atividade e repercussão* dos alunos. Caso o desempenho obtido for igual ou maior que 3, isto é $DG \geq 3$, o agente mantém a formação dos grupos.

4. Conclusões e trabalhos futuros

Com base neste trabalho, observou-se que um Sistema Multiagente se mostra bastante útil em considerar fatores sócio-afetivos dos alunos, de forma que estes venham contribuir significativamente com os resultados esperados de atividades colaborativas, por conta das características que os agentes inteligentes possuem como a autonomia, flexibilidade, comunicação, habilidade social e por potencializar o aprendizado dos alunos de forma automatizada.

Assim, a definição de uma arquitetura SMA formada por dois agentes de software, denominados de Agente FORMADOR de GRUPOS e de Agente AVALIADOR, visa apresentar uma ferramenta computacional que modele as características de alunos, infira sobre esse modelo e busque uma abordagem pedagógica que se adapte ao aprendiz. Desse modo, os agentes propostos cooperam entre si para aperfeiçoar a aprendizagem colaborativa em ambientes CSCL, através da formação de grupos ao considerar os fatores sócio afetivos *emotividade, atividade e repercussão* dos alunos e o *feedback* dado por professor e alunos sobre os clusters de alunos formados.

Portanto, esta pesquisa teve o intuito de contribuir no desenvolvimento de agentes dotados de características sócio afetivas ao proporcionar a extensão e aprimoramento de um modelo consolidado de Lopes et al. (2011), a fim de integrá-lo em ambiente de aprendizagem baseado em arquitetura multiagente que dá suporte ao trabalho colaborativo.

Como trabalhos futuros, há a necessidade de uma implementação do SMA sócio-afetivos proposto em ambientes CSCL, assim como a realização de testes para validação e aperfeiçoamento de suas funcionalidades aplicadas no processo ensino-aprendizagem virtual. A validação desse sistema ocorrerá ao se propor experimentos cognitivos em ambientes virtuais de aprendizagem em que haverá grupos escolhidos aleatoriamente pelos alunos e grupos definidos pelo SMA proposto. De tal modo que ambos os tipos de grupos (aleatórios e afins) possam ser comparados e, consequentemente, a arquitetura desenvolvida possa ser testada e o seu desempenho verificado.

5. Referências

- Boff, Elisa. (2008) “Colaboração em Ambientes Inteligentes de Aprendizagem mediada por um Agente Social Probabilístico”. Elisa Boff. Porto Alegre: PPGC da UFRGS.
- BONALS, Joan. (2003) “O Trabalho em Grupo e a Avaliação: Enfrentando as Dificuldades de Avaliação do Desempenho dos Alunos”. Porto Alegre - RS.
- Cunha, Magela Cunha. (2002) “Formação de Grupos de Trabalho Utilizando Agentes de Software”. Departamento de Informática da PUC-Rio.
- Dias, Rosilânia Aparecida; Leite, Lígia Silva. (2010) “Educação à distância: da legislação ao pedagógico”. Petrópolis, RJ: Vozes.
- Jaques, Patrícia Augustin. (1999) “Agentes de software para monitoramento da colaboração em ambientes telemáticos de ensino”. /Patrícia Augustin Jaques. – Porto Alegre, 1999. 89f. Diss. (Mestrado) - Fac. de Informática, PUC-RS.
- Justo, F. S. C. (1966) “Teste de Caráter ao Alcance de Todos”. Editora Escola Profissional La Salle, Canoas Rio Grande do Sul.

- Le Senne, René. (1963) “*Traité de caractérologie*”. Paris: Presses universitaires de France, <http://caracterologie.ouvaton.org/>.
- Lima, M. R. C.; Labidi, S.; Filho, O. C. B.; Fonseca, L. C. C. (2005) “Aprendizagem cooperativa e o problema de formação de grupos”. In: *Renote – Revista de Novas Tecnologias na Educação*, Porto Alegre, v. 3, n. 1.
- Lopes, José Ahirton Batista Filho; Quarto, Cícero Costa e França, Rômulo Martins. (2011) “Um Algoritmo de Clustering no Auxílio à Formação de Grupos Sócio Afetivos Afins para o Ensino em Ambientes CSCL”. Sixth Latin American Conference on Learning Objects and Technology in Education LACLO, 2011, submission 73.
- Silveira, Sidnei Renato. (2006) “Formação de Grupos Colaborativos em um Ambiente Multiagente Interativo de Aprendizagem na Internet: um estudo de caso utilizando sistemas multiagentes e algoritmos genéticos”. Porto Alegre: PPGC da UFRGS, 2006. 125 f.: il.
- Stahl, G., Koschmann, T., e Suthers, D. (2006) “Computer-supported collaborative learning: An historical perspective”. Cambridge University Press. pp. 409-426.
- Vygotsky, L. S. (1998) “A Formação Social da Mente: o Desenvolvimento dos Processos Psicológicos Superiores”. São Paulo, Editora Martins Fontes, 10 p.

Cooperative UAVs using multi-agent coordination techniques for search operations

Aquila N. Chaves¹, Paulo Sérgio Cugnasca¹

¹Grupo de Análise de Segurança (GAS)

Escola Politécnica da Universidade de São Paulo (EP/USP)

Av. Prof. Luciano Gualberto, 158, trav. 3, nº 158 – 05508-900 – São Paulo, SP – Brazil

aquila.chaves@usp.br, paulo.cugnasca@poli.usp.br

Abstract. A multi-agent model of cooperative UAVs applied to search and rescue operations is proposed. The advantages of applying this kind of robot are: not exposing the crew to risks; cost reduction; and the possibility of flying during tens of hours without rest. The model comprises the agent knowledge modeling, the individual planning of each agent, the use of coordination mechanisms from the multi-agent systems theory and the cooperation strategy. Simulations of a search operation involving two cooperative UAVs show an average reduction of 55% of the time required to find all lost search objects, comparing with the time required to find the same objects in a non-cooperative operation with the same two UAVs.

Resumo. Este trabalho propõe um modelo multiagente de VANTS cooperativos para operações de busca e salvamento. As vantagens da utilização desse tipo de robô são: não exposição de pessoas a riscos; redução de custos; e a possibilidade de operar por longos períodos sem descanso. O modelo comprehende a modelagem do conhecimento dos agentes, o planejamento individual de cada agente, a utilização de mecanismos de coordenação e a estratégia de cooperação. Simulações envolvendo dois VANTS cooperativos foram realizadas e, comparado ao tempo de busca de uma operação envolvendo duas aeronaves não cooperativas, observou-se uma redução do tempo de busca de 55% em média.

1. Introdução

Numa operação de busca e salvamento (mais conhecida pela sigla em inglês SAR – *Search and Rescue*), as operações de busca constituem a fase mais importante [DECEA 2009]. Nesse contexto, o objetivo do trabalho é propor um modelo de Veículos Aéreos Não Tripulados (VANTS) cooperativos para uma operação de busca, aumentando, assim, as chances de sucesso da operação de busca e salvamento.

As principais vantagens que a utilização de VANTS traz para esse tipo de operação são: esses robôs podem se submeter a uma exposição maior ao risco do que aeronaves tripuladas (voos em baixa altitude, por exemplo); são capazes de executar operações de longa duração, sem necessidade de descanso (ao contrário dos voos tripulados); e apresentam menor consumo de combustível. Além disso, de acordo com a doutrina SAR internacional, não se deve utilizar mais de uma aeronave tripulada para realizar buscas em uma mesma subárea, pois essa situação geraria um estado de alerta prejudicial ao sucesso

da operação, fazendo com que a tripulação tenha a sua atenção desviada das buscas para a navegação e a coordenação com outras aeronaves [IMO/ICAO 2003].

Para atingir o objetivo deste trabalho, o problema de busca utilizando VANTs cooperativos foi dividido em quatro partes: modelo de conhecimento para troca de informações entre os agentes; proposta de planejamento baseada em algoritmos de navegação para VANTs e em padrões de busca estabelecidos pelas instituições responsáveis por operações SAR; obtenção de cooperação por meio de mecanismos de coordenação multiagente; e definição de estratégia de cooperação adequada ao cenário de busca.

A ideia é que, atuando de forma autônoma e distribuída, cada VANT tome suas próprias decisões levando em conta o objetivo de reduzir o tempo de busca e as ações dos outros VANTs. Com isso, espera-se que o desempenho do grupo cooperativo seja maior do que o desempenho de um grupo não cooperativo com o mesmo número de indivíduos.

Este trabalho está estruturado da seguinte forma: a Seção 2 apresenta alguns conceitos sobre os VANTs e as perspectivas de futuro (os VANTs cooperativos); apresentação do modelo de VANTs cooperativos, que compreende mecanismos de coordenação (Seção 3), modelagem do conhecimento do agente de modo adequado à solução do problema (Seção 4), planejamento individual dos agentes (Seção 5) e estratégia de cooperação (Seção 6); e, por fim, a Seção 7 e a Seção 8 apresentam as simulações realizadas e as considerações finais, respectivamente.

2. Veículos Aéreos Não Tripulados Cooperativos

Apesar de os primeiros protótipos de VANTs terem surgido na metade do século XIX e os primeiros modelos precursores dos que se vê hoje terem sido construídos na segunda metade do século XX, foi somente no início do século XXI que se observou um verdadeiro interesse por VANTs. Nesses últimos anos, verificou-se um grande aumento de investimentos e de pesquisas em VANTs, com previsão de crescer ainda mais [UAS Center of Excellence 2010, Teal Group 2011].

Após décadas de avanço, os principais desafios referentes aos veículos aéreos não tripulados estão relacionados ao voo cooperativo, que é um problema predominantemente computacional. Pesquisadores e relatórios governamentais apontam que, no futuro, múltiplos robôs voadores serão capazes de atuar, até sob a forma de enxames, cooperativamente e de modo autônomo [UAS Center of Excellence 2010], funcionando como uma rede coordenada de sensores que cumprirão missões complexas sem nenhuma intervenção humana.

Dentro dessa perspectiva, o problema a ser resolvido é a coordenação de múltiplos veículos. Sendo assim, há duas abordagens possíveis [Ren and Cao 2011]: adotar o controle centralizado ou o distribuído. A abordagem distribuída, apesar de mais complexa, apresenta vantagens tais como flexibilidade, escalabilidade e maior tolerância a falhas. A seguir, a Subseção 2.1 apresenta as principais linhas de pesquisa em VANTs cooperativos e a Seção 3 os caminhos para alcançar a cooperação de VANTs utilizando a teoria multiagente.

2.1. Controle distribuído

Para alcançar a cooperação de agentes autônomos distribuídos (incluindo VANTs), a revisão da literatura permite identificar seis categorias de cooperação de VANTs [Ren and Cao 2011]: (i) coordenação por consenso, em que os agentes tomam decisões consensuais, com participação de todos os indivíduos envolvidos; (ii) controle por formação distribuída, que é baseado na formação de um padrão geométrico (por exemplo, o padrão de voo adotado por grupos de aves migratórias); (iii) otimização distribuída, na qual heurísticas de otimização são utilizadas; (iv) distribuição de tarefas, em que um objetivo maior é dividido em tarefas menores, que são distribuídas entre os agentes; (v) controle por estimativa distribuída, cujos principais objetivos são a coleta de informações e a tomada de decisões distribuídas, em que sensores espalhados entre os agentes são utilizados para realizar estimativas globais; (vi) coordenação inteligente, em que cada agente é dotado de certa “inteligência” e, portanto, planeja suas próprias ações levando em conta o seu próprio conhecimento.

As três primeiras abordagens, (i), (ii) e (iii), não são atrativas para o objetivo deste trabalho. A busca por uma solução consensual degradará a robustez do sistema, de modo que as decisões serão tomadas apenas se houver consenso no grupo de VANTs, situação que nem sempre é possível. A formação de padrão geométrico, embora bastante utilizada em pesquisas utilizando múltiplos VANTs, também não é interessante para o cenário deste trabalho porque haveria perda de autonomia e os VANTs perderiam a capacidade de responder a modificações do ambiente. Por último, a otimização distribuída demandaria muito processamento, situação incompatível com a operacionalização de VANTs cooperativos.

Este trabalho se enquadra nas categorias (iv), (v) e (vi).

3. Coordenação Multiagente

A abordagem multiagente fornece um conjunto de mecanismos e protocolos de negociação, de tomada de decisão descentralizada e de coordenação [Pechoucek and Sislak 2009]. Ademais, não há qualquer restrição na utilização de agentes em sistemas complexos ou críticos [Jennings and Wooldridge 1998], como é o caso dos VANTs cooperativos.

Portanto, com o intuito de obter a cooperação de VANTs, também faz parte das contribuições deste trabalho o mapeamento dos principais mecanismos de coordenação encontrados na literatura, apresentados na Subseção 3.1, bem como a avaliação da adequação de cada um deles ao modelo de VANTs cooperativos.

Por fim, cabe ressaltar que os mecanismos de coordenação multiagente não são intercambiáveis. Ou seja, cada um possui um enfoque diferente dos outros, sendo mais, ou menos, adequado para cada cenário de cooperação.

3.1. Coordenação

Cooperação pode ser definida como coordenação com um objetivo comum [Luck et al. 2005]. Já a coordenação, que é o maior problema na cooperação de agentes (desempenhando um papel essencial na gestão de desastres [Pereira et al. 2011]), pode ser definida como o gerenciamento de interdependências das ações dos agentes

[Wooldridge 2009], visando coerência nas ações dos agentes [Nwana et al. 1996]. Portanto, o mapeamento de mecanismos de coordenação se faz necessário para alcançar a cooperação entre os VANTs.

Adicionalmente, há quatro situações em que a coordenação é necessária [Bellifemine et al. 2007]: (i) quando os objetivos dos agentes conflitam com suas ações; (ii) quando os objetivos dos agentes são interdependentes; (iii) existência de agentes com diferentes capacidades e conhecimentos complementares; (iv) ou, quando há o desejo de que os objetivos possam ser alcançados mais rapidamente, com cada um dos agentes se empenhando em um objetivo paralelamente. No modelo de VANTs cooperativos para operações de busca, as situações (i), (ii) e (iv) são bastante visíveis, pois as rotas dos VANTs podem ser coincidentes, seus objetivos são interdependentes e deseja-se realizar a busca no menor tempo possível.

3.1.1. Organização estrutural

O mecanismo de coordenação baseado na organização estrutural se baseia num *framework* que define atividades e interações entre os agentes por meio da definição de regras, canais de comunicação e relações de autoridade [Durfee 1999]. Esse mecanismo se utiliza da metáfora social, em que cada indivíduo possui habilidades, papéis e responsabilidades diferentes. Dessa forma, o comportamento coordenado do grupo de indivíduos é resultado dessas regras e relações.

Embora seja uma abordagem bastante interessante, nem sempre é possível prever, com segurança, as ações dos agentes. Por esse motivo, esse mecanismo de coordenação foi considerado inadequado ao modelo de VANTs cooperativos.

3.1.2. Acordos bilaterais

Outra importante técnica de coordenação multiagente é a utilização do protocolo *Contract Net* [Smith 1980]. Essa abordagem é inspirada num mercado descentralizado, em que os agentes podem desempenhar apenas dois papéis: contratante e contratado. Basicamente, a ideia é que, para alcançar um objetivo, um agente encontre outros agentes capazes e com disposição de colaborar.

Esse mecanismo possui duas variações [Wooldridge 2009]: “resolução de inconsistências” e “divisão de tarefas”. Na primeira, cada agente planeja suas atividades individualmente e apenas os planos conflitantes são resolvidos. Na segunda, o processo divide-se em três fases: decomposição do problema, resolução do subproblema e síntese da solução [Smith and Davis 1981] – assim, o maior desafio é descobrir como dividir e alojar as tarefas.

A simplicidade e a eficácia desse mecanismo faz que ele seja bastante empregado em pesquisas envolvendo múltiplos VANTs.

3.1.3. Compartilhamento de informações

O compartilhamento de informações nada mais é do que a coordenação pela troca de informações relevantes à solução do problema. Os agentes podem trocar informações pró-ativamente, quando um agente envia informação a outro(s) agente(s) porque acredita que esta será útil, ou reativamente, quando o agente envia informações somente quando é solicitado [Wooldridge 2009].

Esse mecanismo aumenta a confiabilidade, a completeza e a precisão da solução do problema, e reduz o tempo de resposta [Durfee 1999], pois informações de diversos agentes podem ser conferidasumas com as outras e a visão da operação é ampliada.

No modelo de cooperação de VANTs proposto, a utilização desse mecanismo se faz pela troca do conhecimento dos agentes (apresentado na Seção 4). Isso aumenta a capacidade de planejamento do VANT, na medida em que este possui mais informações disponíveis para tomar suas decisões.

3.1.4. Planejamento multiagente

Esse mecanismo aborda o problema de coordenação de agentes como um problema de planejamento, que pode ser centralizado ou distribuído. A estratégia é planejar previamente as ações dos agentes de modo a evitar inconsistências e conflitos com os planos dos outros agentes [Bellifemine et al. 2007].

A resolução de conflitos pode, inclusive, ter como objetivo a manutenção da segurança dos agentes. Durante a implementação da simulação, levou-se em consideração que a cooperação não deve ser incondicional, ou seja, o VANT deve cooperar com os outros, mas não deve colocar em risco sua própria segurança ou a segurança dos demais. Dessa forma, este mecanismo de coordenação foi utilizado para que os VANTs evitem rotas de colisão ao se cruzar subáreas de outros VANTs.

No planejamento centralizado, há um agente central, que recebe os planos parciais ou locais de cada agente, identifica os eventuais conflitos, altera pontualmente os planos eliminando os conflitos e os devolve aos agentes [Georgeff 1983]. Já no planejamento multiagente distribuído, os agentes devem comunicar suas intenções aos outros para eliminar os conflitos de planos existentes [Martial 1990].

A vantagem da utilização do planejamento multiagente distribuído é o alto grau de descentralização que pode ser obtido. No modelo de cooperação de VANTs proposto, esse mecanismo é utilizado para evitar colisões entre os VANTs. Na Seção 6 esse processo é apresentado em detalhes.

3.1.5. Negociação

O mecanismo de negociação é o processo de comunicação em que um grupo de agentes busca um acordo a respeito de alguma situação [Bussmann and Müller 1993]. Por isso, provavelmente esse é o mecanismo de coordenação multiagente mais confiável, pois todos os agentes do grupo são partes ativas no processo de decisão, podendo haver concessões

de algumas partes. Assim, evita-se que uma decisão de um agente prejudique outro agente sem o seu consentimento [Bellifemine et al. 2007].

Entretanto, perde-se em robustez, na medida em que uma falha de comunicação de um agente pode, erroneamente, levar o sistema a não tomar uma decisão. Além disso, o tempo de processamento pode ser consideravelmente comprometido pela necessidade de participação de todos os indivíduos.

Portanto, esse mecanismo de coordenação também foi considerado inadequado ao modelo de cooperação de VANTs.

4. Conhecimento do agente

As informações mantidas e trocadas pelos agentes representam o conhecimento sobre o ambiente, possibilitando, portanto, acompanhar a evolução da operação de busca. Sem essa troca de informações, o conhecimento de cada VANT estaria restrito às informações locais obtidas pelos seus próprios sensores. Assim, por meio da troca de conhecimento, os VANTs passam a “enxergar” o ambiente não só com o seu sensor, mas também com os sensores dos outros agentes. Consequentemente, melhora-se a qualidade do planejamento individual.

A Figura 1 ilustra a estrutura desse conhecimento. O espaço de busca é discretizado em células que possuem uma medida da probabilidade de conter um objeto de busca (objeto que se deseja buscar – pessoas, fuselagens, botes, etc.). Além disso, cada célula do conhecimento deve ter tamanho adequado, de modo que o quadrado fique circunscrito na área de visada do VANT quando este sobrevoa o ponto central da célula.

A situação descrita na Figura 1 é um exemplo de detecção de objeto, em que as células ao redor da célula onde o objeto foi detectado têm as medidas de probabilidade aumentadas. Os valores apresentados são meramente ilustrativos e indicam que as células com valores maiores que zero devem ser priorizadas em relação às outras. No contexto da estratégia de cooperação (Seção 6), esse processo – de detecção, atualização do conhecimento e compartilhamento de informações – será apresentado dentro do contexto da estratégia de cooperação.

5. Planejamento individual do agente

O plano de busca dos agentes corresponde a um plano de voo, que por sua vez é uma sequência de pontos (*waypoints*) gerados por algoritmos de navegação que consideram cada célula do espaço de busca (Figura 1) como um nó navegável. Durante a operação de busca, mais de um algoritmo pode ser utilizado – a Seção 6 apresenta como eles são combinados para formar a estratégia de cooperação.

Além dos padrões de busca estabelecidos pelos órgãos responsáveis por operações de busca e salvamento [DECEA 2009], identificou-se na literatura a existência de duas classes de algoritmos de navegação de VANTs: algoritmos de navegação ponto-a-ponto e de busca local.

Os algoritmos de navegação ponto-a-ponto fornecem uma trajetória de navegação de um ponto de origem até um ponto de destino. Já os de busca local tem o objetivo de, dada uma região, varrer-la exaustivamente da melhor maneira possível, maximizando a probabilidade de encontrar um objeto de busca.

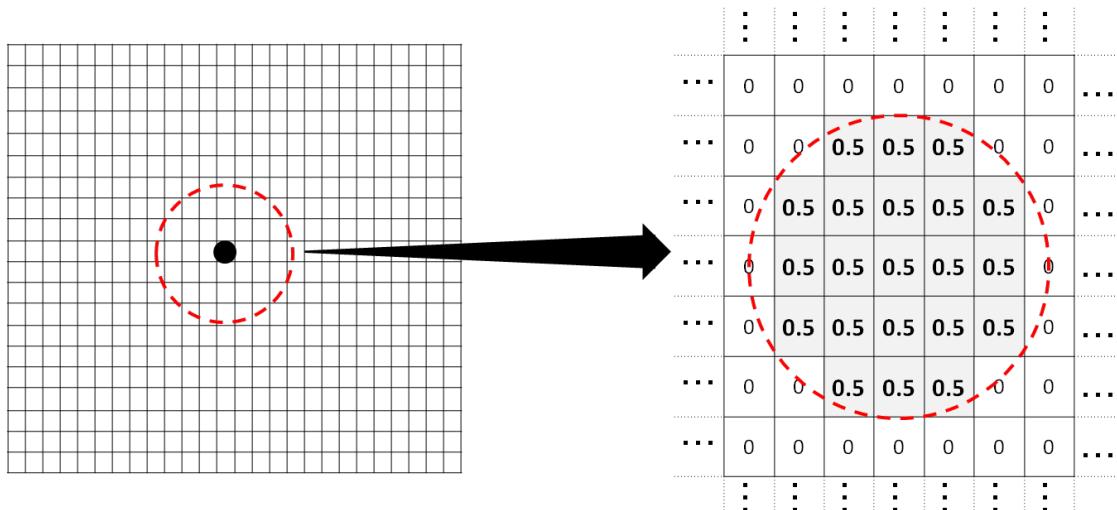


Figura 1. Conhecimento do agente

O algoritmo mais utilizado na navegação ponto-a-ponto de VANTs de pequeno porte é o algoritmo A* [Lester 2005], que muitas vezes é implementado com alguma adequação ao problema que se quer resolver. Esse algoritmo se mostra bastante flexível e rápido.

Outros algoritmos encontrados na literatura são os baseados em forças virtuais, entre eles o *fuzzy virtual force method* [Zhuoning et al. 2010]. Nesses algoritmos, os VANTs são atraídos aos pontos de destino por forças de atração e são repelidos de obstáculos por forças de repulsão. Sendo assim, esses algoritmos possuem a desvantagem de que a navegação é guiada por comandos do tipo “vire à esquerda” e “vire à direita” (ao invés de “passe pelos pontos...”), tornando-os menos adequados ao problema deste trabalho. Por isso, optou-se, neste trabalho, pela utilização do algoritmo A* para a navegação ponto-a-ponto.

Quanto aos algoritmos de busca local, é possível identificar a existência de dois grupos: a varredura concentrada e a navegação por gradiente. O primeiro, ilustrado na Figura 2, visa sobrevoar uma área de maneira uniforme com trajetória em “zig-zag”, cobrindo todas as células dessa área. O segundo, bastante utilizado em aplicações de robótica, guia o robô fazendo com este desloque sempre na direção em que determinado atributo, medido por sensores, é aumentado. No caso do modelo de VANTs cooperativos em questão, o sentido do gradiente seria o sentido em que probabilidade de detecção de objetos de busca aumenta.

Por fim, dentre os padrões de busca definidos pelo Manual de Busca e Salvamento [DECEA 2009], cabe destacar o padrão “Rotas Paralelas”. Semelhante ao algoritmo da Figura 2, esse padrão também se baseia em sobrevoos em “zig-zag”. Mas, ao contrário da varredura concentrada, esse padrão não visa necessariamente cobrir todas as células, mas, principalmente, cobrir uma área uniformemente. Ou seja, pode-se optar por não sobrevoar determinadas colunas ou linhas de células entre as idas e vindas, reduzindo, assim, a densidade da varredura e aumentando a velocidade de progressão do VANT.

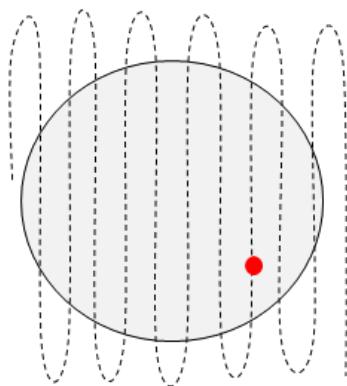


Figura 2. Busca local. A região circular representa a localização provável do objeto, o ponto escuro o objeto e a linha tracejada o plano de voo da aeronave.

6. Estratégia de cooperação

Durante uma operação de busca, os agentes compartilham as informações coletadas e escolhem o algoritmo de navegação mais adequado em cada momento, realizando, assim, um planejamento dinâmico da busca. Para isso, a definição prévia do cenário é crucial para o sucesso da operação.

O cenário de busca definido é a perda de uma aeronave em alto mar com desconhecimento de qualquer informação sobre sua posição, que representa um dos cenários mais complexos no que diz respeito a operações de busca de salvamento. Nessa situação, sabe-se apenas a região retangular onde é possível que estejam as partes que se espalharam (a partir de um ponto central de acordo com uma distribuição gaussiana). Cabe ressaltar que o espalhamento gaussiano está de acordo com o espalhamento marítimo indicado pelo Manual Internacional de Busca e Salvamento [IMO/ICAO 2003].

Assim, os VANTs iniciam a busca escolhendo o padrão de busca “Rotas Paralelas” descrito na Seção 5, que é o padrão ideal para iniciar uma busca nesse cenário [DECEA 2009].

Nessa varredura inicial, cada VANT aloca uma subárea para si e realiza o sobrevoo nessa subárea. O planejamento da trajetória é realizado periodicamente, de modo a incrementar a sequência de próximos de navegação e obedecendo ao padrão de busca “Rotas Paralelas”. Nessa fase da busca não há necessidade de trocar informações sobre o mapa discretizado, pois ainda não há nenhuma informação relevante a ser compartilhada – apenas se sabe, no caso de ainda não ter havido detecção de objeto de busca, onde eles não estão.

Quando um dos VANTs detecta um objeto de busca, ele atualiza o seu conhecimento sobre o espaço de busca, aumentando a probabilidade das células não visitadas ao redor desse ponto (Figura 1). Nesse momento, esse VANT possui uma informação relevante a ser compartilhada. Então, o conhecimento do espaço de busca é enviado aos demais VANTs (agentes).

A cada nova detecção, por qualquer um dos VANTs, o conhecimento é atualizado e compartilhado com os demais. Assim, quanto maior for o número de detecções melhor será o mapeamento da área de espalhamento do acidente. É interessante notar que esse

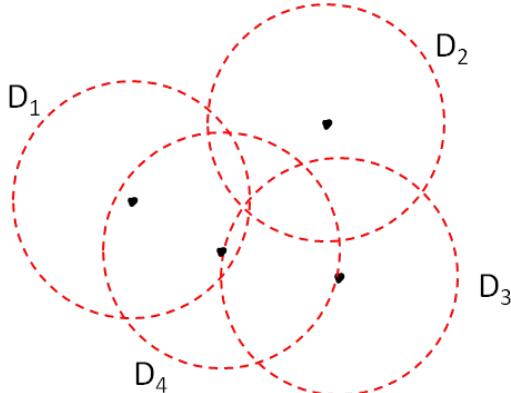


Figura 3. Exemplo de evolução do conhecimento dos agentes

conhecimento vai sendo cooperativamente construído e refinado ao longo da busca. A Figura 3 ilustra um caso hipotético de evolução do conhecimento do grupo de VANTs, em que D₁, D₂, D₃ e D₄ representam detecções sequenciais.

Esse conhecimento é compartilhado pelo envio de mensagens *broadcast*. Na simulação, considerou-se como nulo o tempo de envio (pelo transmissor via *broadcast*) mais a atualização (no receptor) do conhecimento. Assim, quando um VANT detecta um objeto, todos os outros atualizam os seus conhecimentos imediatamente.

No momento em que ocorre a primeira atualização do conhecimento, um dos VANTs se prontifica a realizar a busca local na região que agora apresenta células com maior probabilidade, pois, a existência desse objeto é indício de que há outros objetos próximos daquela célula onde ocorreu a primeira detecção. Então, esse VANT altera seu algoritmo de planejamento de voo e passa a utilizar o algoritmo A* para se deslocar até a célula de detecção. Enquanto isso, o VANT que detectou o primeiro objeto continua sua busca no padrão “Rotas Paralelas”.

Em caso de haver mais de dois VANTs realizando a busca cooperativa, a escolha de qual VANT será eleito para a busca local deve ser feita utilizando o mecanismo bidding (Subseção 3.1.2), em que será escolhido o VANT que estiver mais próximo. No entanto, a simulação apresentada neste trabalho, que envolveu apenas dois VANTs, não empregou esse mecanismo de coordenação.

Durante o deslocamento do VANT pelo algoritmo A*, há a possibilidade de cruzamento de rotas com outros VANTs. Sendo assim, nesse deslocamento o VANT emprega o mecanismo de coordenação Planejamento Multiagente (subseção 3.1.4). Assim, com o intuito de evitar colisões, o VANT que se desloca para a busca local passa utilizar informações de planos e posicionamento dos outros VANTs em seu planejamento de trajetória. Ou seja, no cálculo da trajetória utilizando o algoritmo A*, o VANT que se desloca calcula a distância necessária para chegar até o ponto de destino e, utilizando essa distância e o plano de voo do VANT “dono” da subárea onde se encontra o ponto de destino, estima onde este último estará quando primeiro VANT chegar ao seu destino. Então, no cálculo da trajetória, o algoritmo A* considera a célula onde o segundo VANT estará, e as vizinhas num raio de 500 metros, como células não navegáveis – evitando, assim, a colisão.

Chegando ao local da primeira detecção, o VANT que vinha utilizando o algoritmo A* altera novamente o algoritmo de planejamento de trajetória, passando a utilizar um algoritmo de busca local. Nesse momento, o objetivo desse VANT passa a ser a varredura exaustiva de toda a área contida dentro do círculo criado após a primeira detecção (Figura 1) e as demais que forem se acrescentando a esta (Figura 3), tentando maximizar a probabilidade de detecção.

Nesse processo de busca local, o mecanismo de Planejamento Multiagente também é utilizado. Ao construir a trajetória de busca, o VANT responsável pela busca local desconsidera qualquer célula que está mesma linha no espaço de busca do outro VANT, que continua sua varredura utilizando o padrão Rotas Paralelas. Como a trajetória da busca local é calculada incrementalmente, em passos pequenos, mesmo que o VANT do padrão Rotas Paralelas continue se deslocando, o VANT da busca local nunca “invadirá” alguma célula em rota de colisão.

Por fim, cabe ressaltar que na simulação apresentada na Seção 7 foram utilizados apenas dois VANTS. Por conseguinte, a escolha do VANT para realizar a busca local será óbvia. No caso de haver mais de dois VANTS, pode-se utilizar o mecanismo de “*bidding*” do protocolo *Contract Net*.

7. Simulação e resultados

Simulações utilizando dois VANTS cooperativos indicam uma redução média no tempo de busca de 55%, comparada a uma busca utilizando dois VANTS não cooperativos no mesmo cenário. Essa média foi obtida realizando 40 simulações, número suficientemente grande para assegurar a validade estatística do resultado.

Os parâmetros utilizados na simulação foram os seguintes: lado das células igual a cem metros; utilizando uma distribuição gaussiana com desvio padrão de 1.500 metros, foram espalhados vinte objetos ao redor de um ponto sorteado aleatoriamente; e o espaço de busca foi definido num quadrado com dez quilômetros de lado. Ao final da simulação, gera-se uma sequência de comandos de MatLab responsáveis pela construção da Figura 4.

Os pontos vermelhos da Figura 4 representam os objetos perdidos, as linhas azul (partindo da metade inferior) e verde (na metade superior) representam as trajetórias dos VANTS (que inicialmente empregam o padrão “Rotas Paralelas”) e a área amarela representa as células em que a probabilidade de detecção é maior (essas células tiveram suas probabilidades incrementadas após sucessivas detecções). Observa-se que, conforme o VANT que realiza a busca local vai avançando nessa tarefa, essa área vai sendo “consumida”. Na Figura 4, a simulação foi interrompida quando o último objeto foi detectado.

Por fim, cabe dizer a simulação foi desenvolvida em Java e as simulações foram realizadas numa máquina com sistema operacional Windows Vista (32 bits), com processador Intel Core 2 Duo (2,10 GHz), com memória RAM de 3 GB e utilizou-se a plataforma de desenvolvimento NetBeans 7.0.

8. Considerações finais

Combinando mecanismos de coordenação da teoria multiagente, modelando o conhecimento dos agentes na forma de espaço discretizado em células e concebendo o planejamento individual dos agentes na forma de algoritmos de navegação, este trabalho propôs

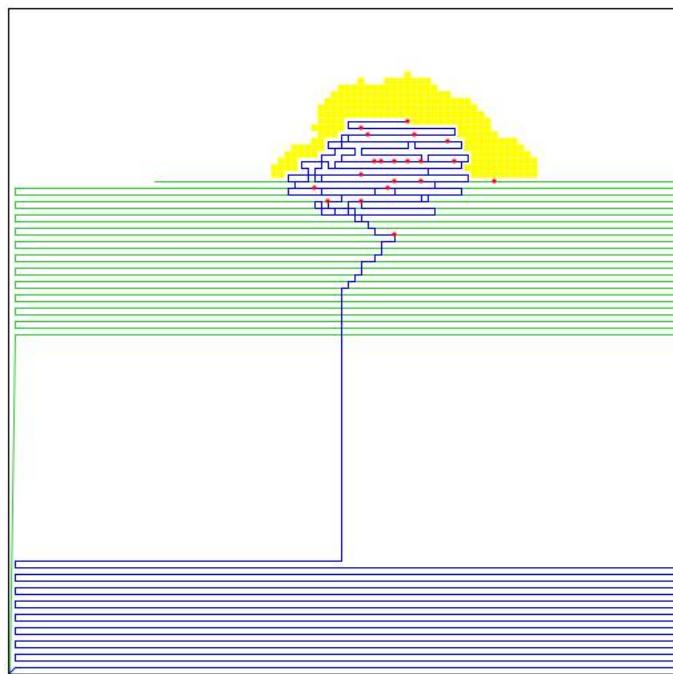


Figura 4. Simulação de cooperação com dois VANTs

um modelo de VANTs cooperativos, com uma estratégia de cooperação que se aplica a operações de busca.

Simulações empregando dois VANTs cooperativos revelaram, em média, 55% de redução do tempo de busca – em comparação a dois VANTs não cooperativos no mesmo cenário.

Outras simulações ainda serão realizadas e há expectativas de obtenção de uma redução do tempo de busca ainda maior. Simulações com quatro VANTs e reduzindo a densidade da varredura inicial pelo padrão “Rotas Paralelas” são exemplos de simulações que podem ser realizadas.

Agradecimentos

Os autores agradecem ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) pelo apoio concedido na forma de bolsa de mestrado.

Referências

- Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing multi-agent system with JADE*. Wiley.
- Bussmann, S. and Müller, J. (1993). A negotiation framework for cooperating agents. In Deen, S., editor, *1992 Proceedings of the Special Interest Group on Cooperating KnowledgeBased Systems*, pages 1–17, Keele, UK. Dake Centre, University of Keele.
- DECEA (2009). Manual de busca e salvamento (mca 64-3). Technical report, Comando da Aeronáutica.

- Durfee, E. H. (1999). Distributed problem solving and planning. In Weiß, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 121–164. MIT Press, Cambridge, MA, USA.
- Georgeff, M. P. (1983). Communication and interaction in multi-agent planning. In *Proceedings of the 3rd National Conference on Artificial Intelligence (AAAI-83)*, pages 125–129, Menlo Park, CA. AAAI Press.
- IMO/ICAO (2003). International aeronautical and maritime search and rescue manual (iamsar manual). Technical report, International Maritime Organization and International Civil Aviation Organization, London, United Kingdom.
- Jennings, N. R. and Wooldridge, M. (1998). *Applications of intelligent agents*, pages 3–28. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Lester, P. (2005). A* path finding for beginners. Acesso em 18/04/2012: <http://www.policyalmanac.org/games/aStarTutorial.htm>.
- Luck, M., McBurney, P., Shehory, O., and Willmott, S. (2005). *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink.
- Martial, F. V. (1990). Interactions among autonomous planning agents. In In Demazeau, Y. and Müller, J.-P., editors, *Descentralized AI - Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 105–120, Amsterdam, The Netherlands. Elsevier Science Publishers B.V.
- Nwana, H., Lee, L., and Jennings, N. (1996). Coordination in software agent systems. *British Telecom Technical Journal*, 14 (4):79–88.
- Pechoucek, M. and Sislak, D. (2009). Agent-based approach to free-flight planning, control, and simulation. *IEEE Intelligent Systems*, 24(1):14–17.
- Pereira, A. H., Nardin, L. G., and Sichman, J. S. (2011). Coordination of agents in the robocup rescue: A partial global approach. In *Anais do V Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações (WESAAC)*, pages 75–84, Curitiba, Brazil.
- Ren, W. and Cao, Y. (2011). *Distributed Coordination of Multi-agent Networks*. Springer.
- Smith, R. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *Computers, IEEE Transactions on*, C-29(12):1104 –1113.
- Smith, R. G. and Davis, R. (1981). Frameworks for cooperation in distributed problem solving. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(1):61 –70.
- Teal Group (2011). World unmanned aerial vehicle systems - 2011 market profile and forecast. Technical report, Teal Group Corporations, Fairfax, VA, EUA.
- UAS Center of Excellence (2010). U.s. army roadmap for unmanned aircraft systems 2010-2035. Technical report, Fort Rucker, Alabama.
- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons, second edition edition.
- Zhuoning, D., Rulin, Z., Zongji, C., and Rui, Z. (2010). Study on uav path planning approach based on fuzzy virtual force. *Chinese Journal of Aeronautics*, 23(3):341 –350.

Ambiente de derivação de sistemas multiagentes industriais apoiados por metodologia e ontologia

Vanderlei L. C. Weber

PPGEAS - Programa de Pós-Graduação em Engenharia de Automação e Sistemas --
Universidade Federal de Santa Catarina (UFSC) -- Caixa Postal 476 -- 88040-900 --
Florianópolis -- SC – Brasil

vweber@das.ufsc.br

Resumo. O estudo apresentado nesse artigo concentra-se em gerar um SMA (*Sistema multiagente*) que represente um cenário de chão de fábrica, apoiada por uma metodologia, em que são analisados os objetivos gerais da indústria em um nível abstrato para chegar a um esqueleto SMA que represente o cenário específico. Para auxiliar esta metodologia, fez-se uso de ontologia e um sistema computacional que segue as etapas modeladas pela metodologia.

1. Introdução

Os modernos sistemas de manufatura vêm requerendo cada vez mais de inteligência dos vários atores envolvidos no planejamento, execução e recuperação de ações, bem como uma grande capacidade de readaptação do seu layout na medida em que novos produtos ou mudanças de processos de inovação são concebidos pela empresa. A abordagem dos sistemas multiagente (SMA) vem sendo considerada uma das mais adequadas para o desenvolvimento de sistemas computacionais para suportar tal adaptação.

Embora haja vários trabalhos em termos de arquiteturas e metodologias para a construção de SMAs, pouco há quando os SMAs são voltados para aplicações industriais, em particular os que envolvem controle de chão-de-fábrica e equipamentos industriais. Uma das razões é a intrínseca complexidade de tal cenário, uma vez que cada equipamento (e seu controlador industrial) é diferente. Existem dezenas de tipos de equipamentos, cada chão-de-fábrica tem um layout e uma arquitetura de controle particular, entre outros aspectos. Paralelamente a isso, há diversas abordagens de “encapsulamento” (*wrapping*) que, numa filosofia de integração *bottom-up*, precisam adaptar-se aos requisitos de integração com SMAs em seus vários níveis de protocolos de comunicação. Dada essa enorme diversidade, é difícil conhecer qual seria a mais adequada arquitetura e conceitos associados para o dado SMA.

Nesse contexto, serão abordados nesse trabalho, como criar SMAs industriais de uma forma que, ao final, este seja coerente com a planta e completo em relação a todos os equipamentos envolvidos, robusto e interoperável em relação às comunicações envolvidas e flexível no sentido de permitir facilmente que um usuário possa introduzir mudanças gerais no sistema e na planta, sem que para isso, o SMA tenha que ser refeito completamente.

A figura 1.1 a seguir ilustra o objetivo dessa pesquisa, ou seja, o desenvolvimento de uma metodologia de construção de SMA para aplicação em chão de fábrica, com auxílio de uma ontologia e um ambiente integrado de criação de tais sistemas, que possa satisfazer as necessidades mencionadas anteriormente.

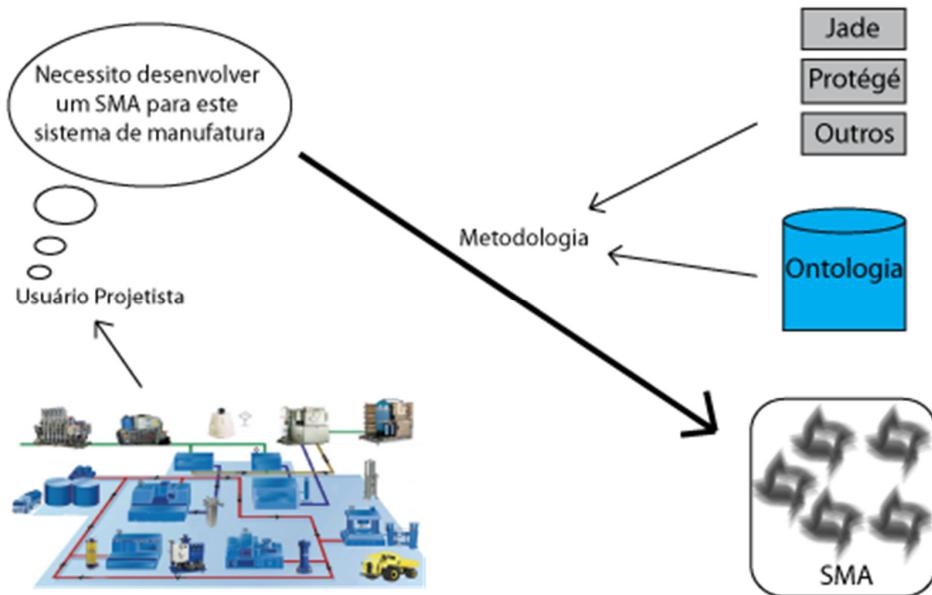


Figura 1.1. Derivação de um chão de fábrica

Na seção 2 é abordado a composição de um cenário industrial e a justificativa para uso de Sistemas multiagentes nesse ambiente; na seção 3, é apresentado uma metodologia e uma ontologia que servirão de base para esse trabalho e um *framework* que auxiliará no desenvolvimento da extensão da metodologia base; na seção 4, a metodologia O-MaSE é estendida, para melhor adequação ao problema de derivação; na seção 5, é demonstrada a utilização do *plugin agenttools* e da ape O-MaSE e na seção 6, as conclusões.

2. Sistemas Multiagentes Industriais

Quando se fala de indústria e planta fabril, normalmente imagina-se algo grande, composto por diversas máquinas, funcionários, transportadores, sistemas de controle, etc[Rabelo 1997]. Apesar da realidade nem sempre ser essa, em que se encontram poucas ou até mesmo uma máquina com algoritmos ótimos e processo de fluxo de produção bem definido, deve-se considerar cenários maiores, cenários que exijam uma característica colaborativa entre os diversos componentes do chão de fábrica, trabalhando em conjunto para atingir o objetivo final de produção. Nesse contexto, visando atender a demanda dos diferentes cenários fabris, independente de seu tamanho e complexidade, é necessário que se projete uma metodologia e um sistema que atenda um tamanho variado de indústrias.

Nesse ambiente característico de colaboração de chão de fábrica, um agente é definido como sendo um sistema de software que se comunica e coopera com outros sistemas de software (agentes ou não) para resolver um problema ou uma tarefa complexa que está além da capacidade de resolução por um único software (agente). Essa é uma definição apoiada por [Jennings, Wooldridge. 1998], que diz que um agente é um sistema de computador situado em algum ambiente e que é capaz de ações autônomas a fim de atender seus objetivos de design. Um agente autônomo deve ser capaz de agir sem a intervenção direta de seres humanos ou outros agentes e deve ter controle sobre suas próprias ações e estados internos. As características mais importantes e úteis num

ambiente de manufatura baseado em agentes são por exemplo: autonomia, cooperatividade, pró-atividade e adaptabilidade.

Algumas das principais características de um chão de fábrica são:

- natureza distribuída;
- sistemas heterogêneos;
- variedade de equipamentos, e;
- protocolos de comunicação particulares.

Relacionando as principais características de um chão de fábrica com SMAs, pode-se comparar a natureza distribuída com ambiente distribuído, sistemas heterogêneos com complexidade, protocolos de comunicação particulares e variedade de equipamentos com integração, justificando, assim, o uso dessa abordagem, em que os SMAs suprem as características principais encontradas em um ambiente industrial.

Segundo [Monostori L, Váncza J, Kumara 2006], o conceito de multiagente industrial está mais ligado à integração. A integração promovida pela adoção de multiagentes, possibilita continuidade operacional de uma determinada parte local da empresa, caso haja algum colapso de conectividade.

As fortes mudanças nos paradigmas das companhias de manufatura, consequentes da globalização da economia, requerem de uma empresa a máxima flexibilidade possível dos sistemas [Torre, Barata, Torre, Flores 1997], o que pode ser alcançado utilizando um sistema multiagente bem projetado. Como a flexibilidade exige aumento da agilidade, é necessário um sistema de supervisão com um nível de inteligência maior, que por sua vez, deve ser distribuído utilizando-se o paradigma de multiagentes.

Em [Rabelo, R.J. 1998] e [Jennings 1994], o comportamento de um agente industrial deve possuir as seguintes características:

- Semiautônomo – certo grau de autonomia, porém depende de outros agentes para concluir suas tarefas.
- Interação – interação com outros agentes visando solucionar um problema.
- Independente – conhecimento suficiente para concluir algumas tarefas.
- Cooperante – coopera com a organização, não possui comportamento destrutivo ou egoísta, buscando o bem comum e solução do problema.
- Benévolos – interesse sempre em cooperar.
- Honesto – não oculta e fornece informações sempre verdadeiras, interagem com outro agente.
- Responsável – ao assumir um compromisso, aplica todos os esforços ao seu alcance para cumprir uma tarefa.

Um comparativo entre SMA Industrial e SMA Tradicional é apresentado na tabela 2.1. A tabela estabelece uma relação de propriedades de SMAs que normalmente é esperada em um sistema multiagente, segundo [W. Shen, Q. Hao, H. Joong, and D.H. Norrie 2006],[Teixeira, L.F.P. and Miranda, R. and de Ávila, E.S. 2001],[Jennings, N.R. and Corera, J.M. and Laresgoiti, I. 1995].

2.1. Tabela comparativa de SMAs

Propriedades	SMA Industrial	SMA Tradicional
Modularidade	Permite o funcionamento de setores locais da fábrica caso haja problemas de conectividade, garantindo a continuidade da produção.	Permite o funcionamento de subsistemas de uma empresa em caso de falha de um módulo.
Flexibilidade	A estrutura pode ser adaptada para executar uma nova tarefa, caso haja uma mudança na produção, ou ingresso de novos equipamentos.	Muito importante devido a alterações de sistemas e agregações de subsistemas ao SMA.
Robustez	Característica importante em caso de quebra de algum equipamento ou ingresso de novos equipamentos no ambiente fabril. Deve também suportar qualquer tipo de alteração do meio.	Característica importante em caso de implementação de nova tecnologia, o SMA deve estar preparado para qualquer alteração no ambiente.
Integração	Ambiente extremamente heterogêneo que necessita de integração entre sistemas e equipamentos.	Ambiente pode ser heterogêneo, necessita de integração entre os diversos sistemas existentes.
Tolerância a Falhas	Muito importante em SMAs industriais, evitando desperdício de matéria-prima e quebra de equipamentos. Evitar a parada da linha de produção	Sua importância ocorre pela consistência de dados.
Cooperação	Deve ser cooperativo para atingir os objetivos da organização.	Pode ser cooperativo, mas existe organização com a presença de agentes não cooperantes.
Domínio	Heterogêneo com relação a equipamentos, sistemas legados e protocolos de comunicação. Forte necessidade de comunicação e coordenação entre equipamentos, o que justifica sua complexidade de controle.	Presença de sistemas legados e sistemas operacionais, bancos de dados e redes de comunicação variados.
Distribuição	Muito característico nesse tipo de domínio.	Pode ser distribuído.

3. Tecnologias de desenvolvimento

Para desenvolver um SMA que atenda as necessidades de chão de fábrica, utilizou-se a metodologia e o *framework* O-MaSE (*Organization-based Multiagent System Engineering*) que modela os fragmentos de métodos propostos pela metodologia através da ferramenta de desenvolvimento Eclipse Ganymede 3.4.2. É necessário também instalar a *api agenttools*, que possibilita construir diagramas UML dentro do Eclipse.

Com o objetivo de especificar e melhor representar o chão de fábrica, fez-se uso de uma ontologia, que permite definir a especificação e conceituação de um determinado domínio, em que basicamente, uma ontologia consiste dos conceitos, relações, suas definições, propriedades e restrições existentes no domínio [Gruber, T.R. 1995].

3.1. Metodología O-MaSE

A metodologia O-MaSE desenvolvida por [Deloach SA. 2006] é derivado da metodologia MaSE (*Multiagent Systems Engineering*) também desenvolvida por [Deloach SA. 1999]. Alguns problemas em relação à MaSE foram corrigidos, mas o maior avanço concentra-se na possibilidade de customização da O-MaSE, proporcionando a *designers* criarem suas próprias metodologias orientadas a agentes. Ela é baseada em um conjunto de fragmentos de métodos que definem um conjunto de produtos de análise e design que podem ser criados e usados dentro do *framework* O-MaSE.

3.2. Ontologia

A arquitetura ADACOR [Borgo, S. and Leit, P. 2004], desenvolvida para representar cenários industriais, possui uma ontologia baseada nas recomendações da FIPA *Ontology Service Recommendation*[C. Schlenoff, A. Knutilla, S. Ray 1996], e é utilizada como base genérica no desenvolvimento da ontologia nesse trabalho. A figura 3.1 a seguir representa essa ontologia, utilizando diagrama UML de classes.

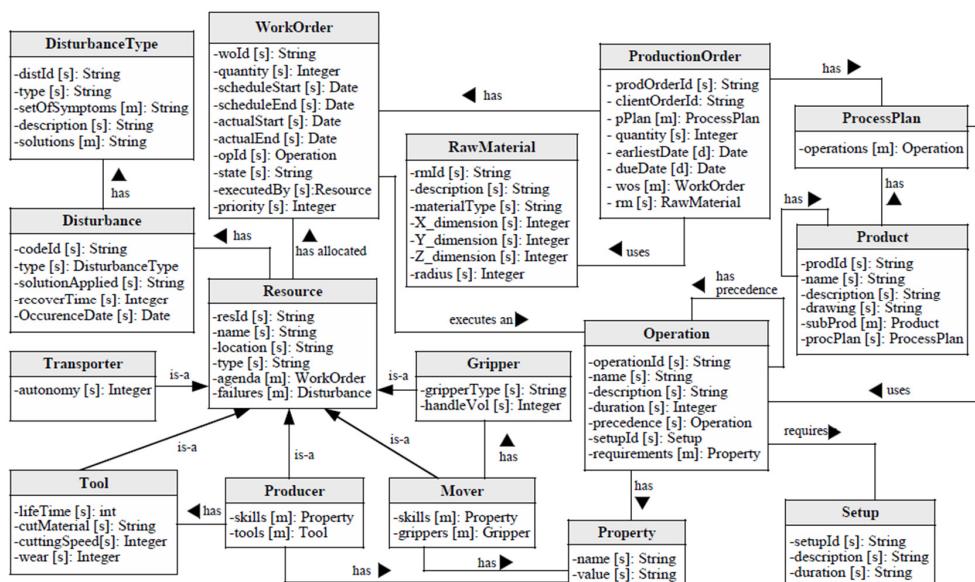


Figura 3.1. Ontologia de manufatura ADACOR

4. Extendendo a metodologia O-MaSE.

Tomando como base a metodologia O-MaSE e os fragmentos de métodos utilizados pelo *framework* O-MaSE, duas novas etapas foram criadas, a fim de melhorar a representação de conhecimento do chão de fábrica, com a finalidade de proporcionar maior integração com o sistema de controle. Essas etapas serão detalhadas em 4.2.

4.1. Construindo uma metodologia usando O-MaSE e Ontologia.

Para construir a metodologia proposta a seguir, foram utilizadas alguns fragmentos de métodos da metodologia O-MaSE e etapas que utilizam conceitos de ontologia. As etapas dessa metodologia são apresentadas da seguinte forma:

- 1 - Especificar e refinar requisitos;
- 2 - Decompor em metas, dividida em outras duas subetapas:
 - 2.1 – Modelar Metas e;
 - 2.2 – Refinar Metas;
- 3 - Descrever chão de fábrica e;
- 4 - Criar ontologia.
- 5 - Criar estrutura, é responsável por unir as especificações das etapas A1,A2,A3,A4 e é dividida da seguinte forma:
 - 5.1 - Definir papéis;
 - 5.2 - Definir Agentes;
 - 5.3 - Definir Protocolos;
 - 5.4 - Definir políticas e;
 - 5.5 - Definir Planos.

Para representar a sequência de etapas propostas, é utilizada a metodologia IDEF0 (*Integration Definition for Function Modeling*), que faz parte do conjunto de padrões da família IDEF, normalmente aplicada em modelagens de sistema. Como esse método é uniforme e de iteração limitada para facilitar o processo, foi muito bem aceita para modelagem de processos industriais, gerenciais, executivos e outros. As caixas retangulares, ICOMs (*Input Control Output Mechanism*), observadas na figura 4.1, representam cada tarefa. Setas indicam a finalidade ou procedência de um determinado dado. A lateral esquerda representa as entradas (*inputs*), a lateral direita as saídas (*outputs*), parte superior as restrições e parte inferior os recursos necessário para executar a tarefa. A figura 4.1, ilustra a estrutura IDEF0, com a proposta da metodologia.

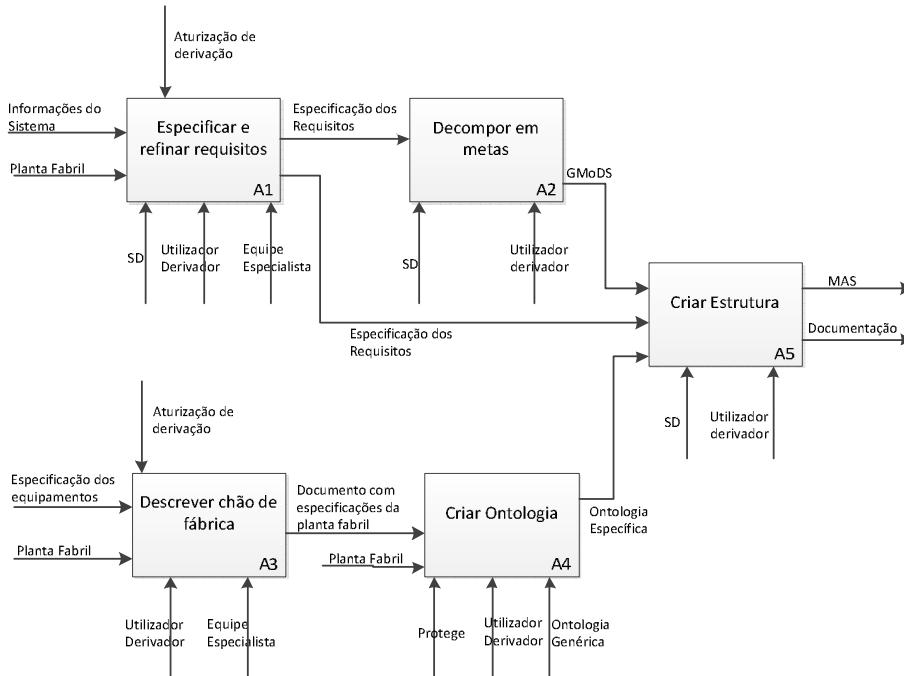


Figura 4.1. Diagrama da metodologia.

Como pode ser observado na figura 4.1, existem dois fluxos de desenvolvimento que podem ser realizados em paralelo. O fluxo iniciado em A1 e o fluxo iniciado em A3. A etapa A5 é responsável por unir os dois fluxos de desenvolvimento.

O cenário de chão de fábrica analisado para aplicar essa metodologia conta com um agv, uma esteira e uma válvula de preenchimento de líquidos. Na descrição das etapas da metodologia, serão detalhadas algumas etapas para melhor compreensão de suas aplicações dentro desse cenário industrial.

4.2. Etapas da metodologia

As etapas ilustradas na figura 4.1, A1, A2 e A5 são derivadas da metodologia O-MaSE e são usadas como fragmentos de métodos no framework O-MaSE. As etapas A3 e A4 foram criadas para melhor especificação do domínio, visando integrar os agentes que serão desenvolvidos na etapa A5.

Como A3 e A4 são adaptações a metodologia e são executadas em paralelo com A1 e A2, elas são primeiramente abordadas:

- Descrever chão de fábrica (A3)

Esta etapa tem como objetivo descrever o relacionamento entre os recursos de produção, suas características e fluxo de produção. Após a derivação ser autorizada têm-se como entrada de dados dessa etapa, as especificações dos equipamentos e a planta fabril do chão de fábrica. Uma equipe de especialistas deve auxiliar o utilizador derivador (usuário projetista, responsável por modelar o cenário), a fim de produzir um documento coerente com o cenário, que sirva como base no desenvolvimento da ontologia específica.

- Definir ontologia (A4)

A representação do conhecimento acerca do chão de fábrica, utilizando ontologia, facilita sua interpretação pelos agentes que serão criados na etapa seguinte. A ontologia ADACOR é originalmente modelada utilizando um diagrama de classe UML, como ilustrado da figura 3.1. Para melhor representação da ontologia ADACOR, utiliza-se a ferramenta Protégé, especialmente desenvolvida para desenvolvimento de ontologias, ilustrada na figura 4.2.

Como não é possível apresentar o processo completo de derivação, apresenta-se na ontologia ilustrada na figura 4.2, as instâncias de dois transportadores (agv01 e esteira01) e uma ferramenta (valvula01), de acordo com o cenário analisado. Essas entidades compõem os recursos de chão de fábrica, podendo sofrer perturbações, devem respeitar uma ordem de produção, compostas por operações e, basicamente seguir uma ordem de trabalho.

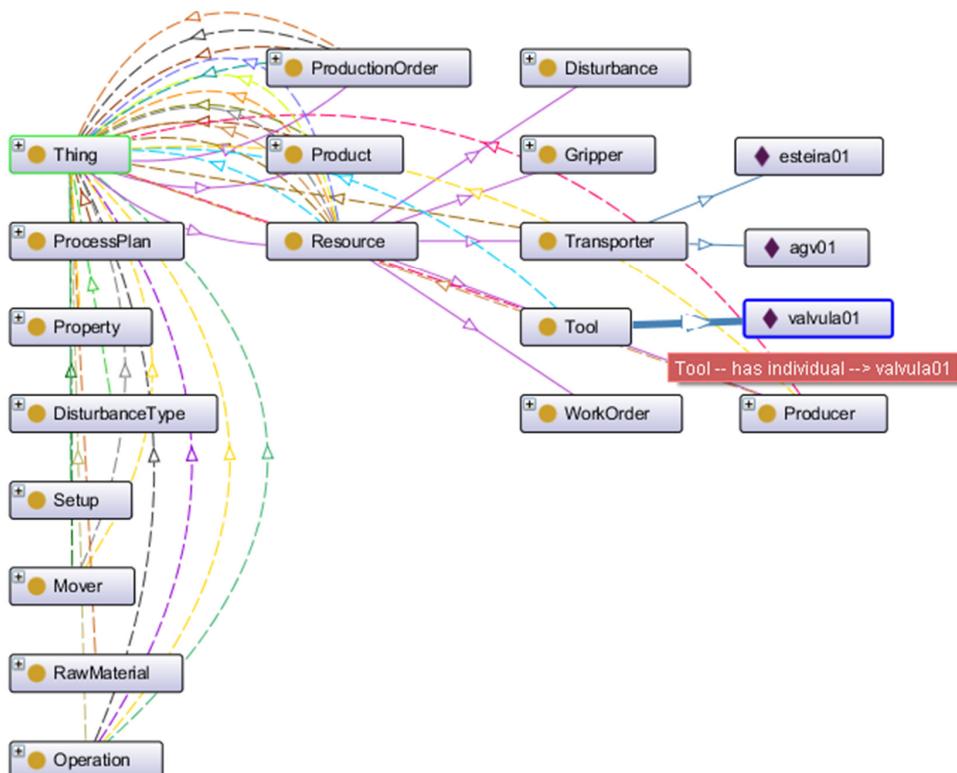


Figura 4.2. Ontologia ADACOR modelada no Protégé, com 3 instâncias do cenário.

As etapas A1 e A2 seguem o modelo especificado pelo framework O-MaSE, em que na etapa A1 será gerado um documento que contém as especificações de o quê e como serão realizados os processos do chão de fábrica. Nessa etapa são listados os requisitos funcionais e não funcionais do sistema. Na etapa A2, o documento gerado pela etapa A1 deve ser traduzido para um diagrama de classe UML, gerando a árvore de objetivos GMoDS(*Goal Model for Dynamic Systems*), ou seja, uma árvore AND/OR. Como a árvore AND/OR é representada por um diagrama de classes, cada classe deve ser

interpretada como uma folha da árvore. Nesse sentido, cada requisito é analisado e um conjunto de metas é traçado, afim de satisfazer os requisitos impostos pela etapa de especificação.

A etapa A5 é responsável por juntar as informações da árvore AND/OR e da ontologia, gerando um código base, esqueletos dos agentes do SMA. Essa etapa conta com as seguintes subetapas:

- Definir Papéis – A relação entre os objetivos da árvore AND/OR é de cardinalidade n para n, ou seja, um objetivo pode precisar de vários papéis para ser atingido ou um papel pode satisfazer mais de um objetivo. Um diagrama UML é utilizado para ilustrar essa etapa.
- Definir Agentes – Esta subetapa define os agentes e quais os papéis que cada agente exercerá. Cada instância gerada na ontologia deve ser representada por um agente nessa etapa. Um diagrama UML é utilizado para ilustrar essa etapa.
- Definir Protocolos – Para representar as mensagens trocadas entre agentes ou entre agentes e papéis é utilizado um diagrama AUML, definindo os protocolos existentes entre eles.
- Definir Planos – A técnica de planejamento captura as ações e protocolos usados pelo agente para adquirir um objetivo, processo representado por um diagrama UML de transição de estados.
- Definir Políticas - Essa etapa identifica as propriedades dos requisitos do sistema, escrevendo-os em linguagem natural. Todas as restrições são identificadas e formalmente especificadas usando linguagem formal.

A metodologia O-MaSE não menciona o uso de ontologia para representação do domínio. Com o intuito de satisfazer essa deficiência, a etapa A4 instancia as entidades agv01, esteira01 e a valvula01, modeladas e representadas pela ontologia ADACOR, que devem ser convertidas em agentes na subetapa Definir Agentes da etapa A5, como pode ser observado no diagrama UML da figura 4.3.

Como mencionada anteriormente nessa mesma seção, os agentes lógicos de transporte e controlador_preenchimento devem ligar-se com os agentes instanciados na ontologia, controlando as ações desses agentes e se comunicando através de protocolos de comunicação estabelecidos na etapa de Definir Protocolos.

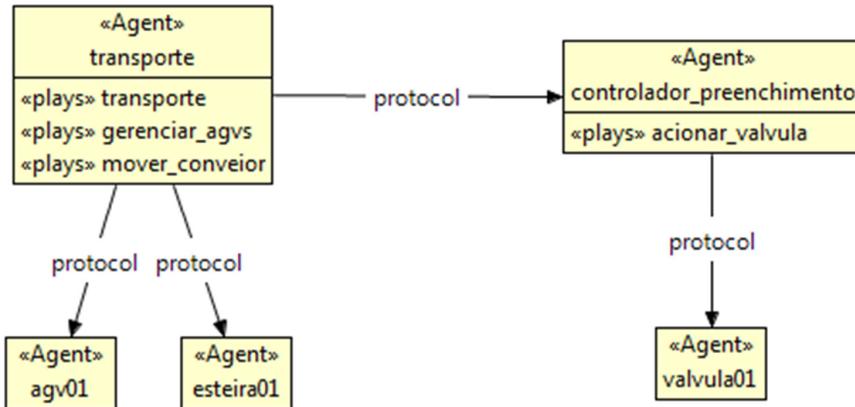


Figura 4.3. Subetapa Definir Agentes

5. Ambiente de desenvolvimento

Para implementar as etapas da metodologia mencionada em 4.1, utilizou-se a APE (Agent Process Editor) O-MaSE e os fragmentos de métodos: Especificação dos Requisitos, Modelo de Objetivos, Modelo de Papéis, Modelo de Agentes, Modelo de Protocolos, Modelo de Planos e Modelo de Políticas. Acrescentando ainda as etapas de Descrição de Chão de Fábrica e Ontologia, ilustrado na figura 5.1.

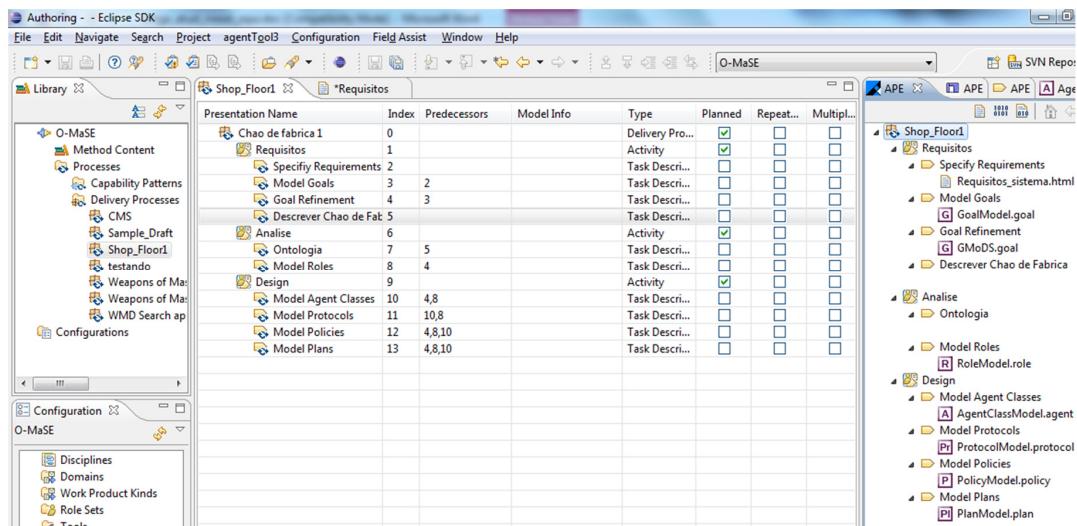


Figure 5.1. Configuração da APE O-MaSE

Cada etapa é desenvolvida usando documentos de especificação, ontologia, diagramas UML e diagramas AUML. Como resultado final da implementação obtém-se o código fonte, ou seja, um “esqueleto” dos agentes que posteriormente deve ser desenvolvida, implementando as ações, métodos e comportamentos específicos de cada agente, na figura 5.2 tem-se uma ideia de como ficaria o código fonte do esqueleto do agente gerado pelo APE O-MaSE. Os comentários no código proporcionam maior compreensão das instruções.

```
public class controlador_preenchimento extends Agent { //classe do agente
    private static final long serialVersionUID = 1;
    protected void setup() {
        System.out.println("Agent "+getLocalName()+" started.");
        //Adiciona o comportamento geral e suas especificações
        addBehaviour(new MyGlobalBehaviour(this));
    }

    private class MyGlobalBehaviour extends Behaviour{//cria classe de comportamento global para esse agente
        private Agent a;
        private static final long serialVersionUID = 1;
        MyGlobalBehaviour(Agent a){
            this.a=a;
        }
        public void action(){
            ACLMessage msg = myAgent.receive(); //Captura mensagem
            if(msg!=null){
                String content = msg.getContent(); // Captura o conteúdo da mensagem
                if(content.compareTo("achievacionar_valvula") == 0){
                    //Adiciona um plano comportamental do agente
                    addBehaviour(new acionar_valvulaFSMBehaviour(a)); // Executa o plano de ação
                }
            } else{ // Senão fica bloqueado
                block();
            }
        }
    }
}
```

Figura 5.2. Esqueleto de código do agente controlador de preenchimento

6. Conclusão

Nesse trabalho foi apresentada uma forma de derivação para um ambiente de manufatura, adaptando uma metodologia utilizada para desenvolvimento de SMA, usando ontologia para representar o conhecimento de um chão de fábrica.

Devido a complexidade encontrada neste ambiente e a necessidade de o SMA representar corretamente o sistema, atendendo as premissas estabelecidas neste trabalho, foi utilizada uma ontologia que oferece suporte à representação de características dos equipamentos e processos. Com isso, torna-se viável incrementar, remover ou substituir componentes do cenário, diminuindo a necessidade de reconstrução completa do sistema. Tais características também podem ser observadas durante as etapas da metodologia, nas quais novas características e processos podem ser adotados.

7. Referências

- Borgo, S. and Leit, P. (2004) “The Role of Foundational Ontologies in Manufacturing Domain Applications System”. In: On the Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE - journal, p. 670-688, Springer.
- C. Schlenoff, A. Knutilla, S. Ray, (1996) “Unified Process Specification Language: Requirements for Modelling Process”, NIST, Interagency Report 5910, Gaithersburg MD.
- Deloach S.A. (1999) “Multiagent systems engineering: A methodology and language for designing agent systems”, DTIC Document.
- DeLoach S.A. (2006) “Multiagent Systems Engineering of Organization-based Multiagent Systems”, 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05), St. Louis, MO. Springer LNCS Vol 3914, pp 109 - 125.
- Gruber, T.R. (1995) “Towards principles for the design of ontologies used for knowledge sharing”, Int. J. Human-Computer Studies, v. 43, n. 5/6.

- Jennings, N.R. and Corera, J.M. and Laresgoiti, I. (1995) “Developing industrial multi-agent systems”, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), p. 423-430.
- Jennings, N. (1994) “*Cooperation in Industrial Multi-Agent Systems*”, World Scientific Series in Computer Science, Vol 43.
- Monostori L, Váncza J, Kumara SRT (2006). “Agent-Based Systems for Manufacturing”. *CIRP Annals - Manufacturing Technology*. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1660277306000053> [Accessed November 2, 2011].
- Rabelo, Ricardo José. (1997) “Um enquadramento para o Desenvolvimento de Sistemas de Escalonamento Ágil da Produção – Uma abordagem Multiagente”. Dissertação Apresentada Para a Obtenção do Grau de Doutor em Engenharia Eletrônica, Especialidade de Robótica e Manufatura Integrada. Universidade de Nova Lisboa, Faculdade de Ciências e Tecnologia, Lisboa.
- Rabelo, R.J. (1998) “Uma Abordagem de Integração Balanceada para Sistemas Multiagente Industriais”, Dissertação (Submetida em Concurso Público Para Professor Adjunto)--Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis.
- Shen W, Hao Q, Yoon HJ, Norrie DH. (2006) “Applications of agent-based systems in intelligent manufacturing: An updated review”. *Advanced Engineering Informatics*. 415-431. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1474034606000292>. Accessed September 19, 2011.
- Teixeira, L.F.P. and Miranda, R. and de Ávila, E.S. (2001) “Engenharia de Sistemas Multiagentes : Uma Investigação sobre o Estado da Arte”, Murilo Juchem e Ricardo Melo Bastos.
- W. Shen, Q. Hao, H. Joong, and D.H. Norrie (2006) “Applications of agent-based systems in intelligent manufacturing : An updated review,” *Advanced Engineering Informatics*, vol. 20, pp. 415-431.

Evolução da Ferramenta MAS-ML *tool* para a Modelagem do Diagrama de Papéis

Állan R. Feijó¹, Felipe J. A. Maia¹, Francisco R. O. de Lima¹, Igor B. Nogueira¹, Enyo J. T. Gonçalves², Emmanuel S. S. Freire¹, Mariela I. Cortés¹, Leandro L. C. de Souza¹

Grupo de Engenharia de Software e Sistemas Inteligentes (GESSI)

¹Universidade Estadual do Ceará, Campus Itapery, 60.740-903 – Fortaleza – CE – Brasil

²Universidade Federal do Ceará, Campus Quixadá, 63.900-000 – Quixadá – CE – Brasil

{allanfeijo1987, felipe.ja.maia, us.robson7, igor.bnog, savio.essf}@gmail.com, enyo@ufc.br, mariela@larces.uece.br

Abstract. *The modeling activity can be highly complex and tending to errors, particularly, in the modeling of Multi-Agent Systems (MASs). Thus, the existence of an adequately support tool can be crucial for choosing and adopting the modeling language to be used. In this context, the MAS-ML tool, based on the MAS-ML modeling language, includes mechanisms to model only two proposed diagrams by MAS-ML. The goal of this paper is present the evolution of the MAS-ML tool aiming to support the role diagram defined in the modeling language follows the model-based approach.*

Resumo. *A atividade de modelagem pode ser altamente complexa e propensa a erros, em particular no caso da modelagem de Sistemas Multi Agente (SMA). Assim sendo, a existência de uma ferramenta de suporte adequada pode ser crucial na escolha e adoção da linguagem de modelagem a ser utilizada. Neste contexto, a ferramenta MAS-ML tool, baseada na linguagem de modelagem MAS-ML, incorpora mecanismos para a modelagem de somente dois dos diagramas propostos por MAS-ML. O objetivo deste artigo é apresentar a evolução da ferramenta MAS-ML tool de forma a dar apoio à modelagem do diagrama de papéis prevista na linguagem de modelagem seguindo a abordagem por modelos.*

1. Introdução

A crescente busca por sistemas mais eficientes leva indefectivelmente, ao desenvolvimento de sistemas cada vez mais complexos. Neste cenário, o paradigma orientado a agentes vem sendo cada vez mais utilizado para lidar com essa complexidade, tanto na indústria quanto na academia.

Um agente pode ser definido como uma entidade autônoma capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores [Jennings, 1996]. O termo Sistema Multi Agente (SMA) refere-se à subárea de Inteligência Artificial que investiga o comportamento de um conjunto de agentes autônomos, objetivando a solução de um problema que está além da capacidade de um único agente [Russel e Norvig, 2004]. Neste cenário, a engenharia de software orientada a agentes (AOSE) [Lind, 2001] surge no intuito de fornecer métodos e ferramentas

adequados para o desenvolvimento de aplicações neste novo domínio, no qual podemos destacar as linguagens de modelagem apoiadas por ferramentas de suporte adequadas e que possuem papel central no desenvolvimento de SMA. No contexto das linguagens de modelagem para SMA destacamos a MAS-ML (*Multi-Agent System Modeling Language*) [Silva, 2004].

O objetivo da MAS-ML é modelar todos os aspectos dinâmicos e estruturais definidos no *framework* TAO (*Taming Agents and Objects*) [Silva, 2004]. O metamodelo de MAS-ML é uma extensão do metamodelo da UML (*Unified Modeling Language*) [OMG, 2011] de acordo com os conceitos definidos no TAO. A linguagem MAS-ML prevê um conjunto de diagramas estruturais, a saber: diagrama de classe, diagrama de papéis e diagrama de organização. Adicionalmente, define também os diagramas dinâmicos de sequência e atividades [Silva, 2007].

A ferramenta MAS-ML *tool* [Farias, 2009] é um ambiente de modelagem (editor gráfico) para SMAs desenvolvido como um *plug-in* da plataforma Eclipse [Eclipse, 2011]. Isto implica que os usuários podem, ao mesmo tempo, modelar SMAs e utilizar os recursos disponíveis dentro da plataforma Eclipse. Em [Gonçalves, 2010] a ferramenta foi estendida para adequar o diagrama de classes à evolução definida em MAS-ML 2.0. Adicionalmente, o diagrama de organização foi implementado. Entretanto, na sua versão atual, o diagrama de papéis, previsto na linguagem de modelagem MAS-ML, não é contemplado na ferramenta.

O diagrama de papéis é importante no contexto de modelagem de SMAs por ser capaz de representar os relacionamentos entre os papéis de agente, os quais não são visíveis nos diagrama de classe e organização. A entidade papel ocupa um lugar de destaque pelo fato dela só poder existir no contexto de uma organização, e suas instâncias devem ser exercidas por agentes, objetos ou mesmo suborganizações [Silva, 2004]. A função do papel é orientar ou restringir o comportamento da entidade que está a exercê-lo.

O objetivo desse artigo é apresentar a evolução da ferramenta MAS-ML *tool* de forma a dar suporte à modelagem do diagrama de papéis de acordo com a especificação apresentada em MAS-ML 2.0. A nova versão da ferramenta é ilustrada a partir de um estudo de caso no qual os papéis utilizados no ambiente de aprendizagem virtual *Moodle* (*Modular Object-Oriented Dynamic Learning Environment*) [MOODLE, 2011] foram modelados.

Este artigo está estruturado da seguinte maneira: Na Seção 2 são apresentados os principais conceitos da linguagem de modelagem MAS-ML 2.0 e da ferramenta MAS-ML *tool*. Na Seção 3, a implementação do diagrama de papéis na ferramenta MAS-ML *tool* é descrita. Em seguida, o estudo de caso utilizando a ferramenta é ilustrado na Seção 4. Alguns trabalhos relacionados são apresentados na Seção 5. Finalmente, as conclusões e trabalhos futuros são descritos na Seção 6.

2. Referencial Teórico

Nesta seção é apresentado o referencial teórico em relação à linguagem de modelagem MAS-ML em sua versão 2.0, e a ferramenta de suporte à modelagem, MAS-ML *tool*.

2.1. MAS-ML 2.0 e o Diagrama de Papéis

MAS-ML é uma linguagem de modelagem que estende a UML e que incorpora o conceito de agente definido no *framework* conceitual TAO para a modelagem SMAs. Originalmente, MAS-ML foi projetada para modelar apenas agentes pró-ativos orientados a objetivos e guiados por planos. Na sua versão mais atual, a linguagem MAS-ML contempla o suporte à modelagem das diversas arquiteturas internas, a saber: arquiteturas internas de (i) agentes reativos simples; (ii) agentes reativos baseados em conhecimento; (iii) agentes baseados em objetivo com planejamento e (iv) agentes baseados em utilidade. Em MAS-ML 2.0 foram incluídos os conceitos de arquiteturas internas de agente na linguagem de maneira conservativa em relação à MAS-ML, ou seja, mantendo-se a representação inicial prevista na linguagem.

A linguagem MAS-ML contempla um conjunto de diagramas estáticos, a saber: diagrama de classes, organização e papéis. O diagrama de papéis é responsável por expressar relacionamentos entre papéis de agente e papéis de objeto identificados no diagrama de organização. Esse diagrama também identifica as classes acessadas pelos papéis de agente e papéis de objeto. As interações entre agentes e organizações de um sistema são descritas com base nos papéis ilustrados pelo diagrama de papel [Silva, 2004].

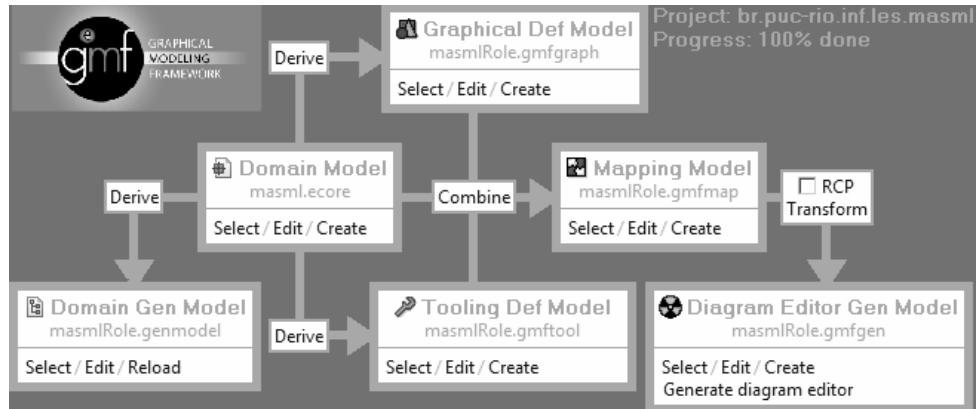
Segundo [Silva, 2004], um diagrama de papel pode mostrar os relacionamentos entre a classe do papel do agente e a classe do papel de objeto e entre esses papéis e classes. O conjunto de relacionamentos usado nesse diagrama é: (i) *Control*, usado entre classes de papel de agente; (ii) *Dependency*, usado entre classes de papel de objeto, entre classes de papel de agente e classes de papel de objeto e entre classes de papel de agente; (iii) *Association*, usado entre classes de papel de objeto, entre classes de papel de agente e classes de papel de objeto, entre classes de papel de agente e entre classes e classes de papel; (iv) *Aggregation*, usado entre classes de papel de objeto e entre classes de papel de agente; (v) *Specialization/ Generalization*, usado entre classes de papel de objeto e entre classes de papel de agente.

2.2. MAS-ML TOOL

A ferramenta MAS-ML *tool* é um ambiente de modelagem (editor gráfico) que serve como um *plug-in* da plataforma Eclipse, possibilitando a utilização dos recursos disponíveis dentro da plataforma Eclipse durante a modelagem de SMAs. MAS-ML *tool* foi criada para dar suporte à modelagem dos diagramas contemplados na linguagem MAS-ML, e na sua versão atual, a ferramenta fornece apoio para a construção dos diagramas de classe e organização de acordo com MAS-ML 2.0.

MAS-ML *tool* foi gerada a partir do *plug-in* do GMF (*Graphical Modeling Framework*) [GMF, 2011] que é um *framework* para desenvolvimento de editores gráficos para modelos de domínio. Ele surgiu a partir da união de dois *frameworks* denominados GEF (*Graphical Editing Framework*) [GEF, 2011], utilizado para a criação de editores gráficos genéricos, e EMF (*Eclipse Modeling Framework*) [EMF, 2011], que permite ao desenvolvedor construir metamodelos e gerar código Java relativo ao mesmo.

A construção do GMF seguiu uma abordagem dirigida por modelos, onde o modelo central e de maior abstração é o próprio metamodelo da linguagem MAS-ML.

**Fig. 1. GMF Dashboard.**

As etapas (Figura 1) para a sua construção são as seguintes: (i) *domain model*, onde é definido o modelo de domínio; (ii) *domain gen model*, que estende o modelo de domínio através da geração de dados (Classes java); (iii) *graphical def model*, onde são definidos os controladores e as figuras; (iv) *tooling gef model*, onde é definido o menu, a paleta, os botões, etc.; (v) *mapping model*, onde é feito o mapeamento dos controladores e as figuras; (vi) *diagram editor gen model*, onde é feita a geração do projeto executável.

3. Implementação do Diagrama de Papéis

A estratégia adotada para implementar as extensões propostas segue a abordagem dirigida por modelos, utilizada originalmente para desenvolver a própria ferramenta. Neste caso é utilizado como modelo central o metamodelo da linguagem MAS-ML 2.0. Essa implementação seguiu as etapas descritas a seguir, de acordo com o GMF.

3.1. Extensão do *Domain Model*

O *Domain Model* contém o metamodelo “*masml.ecore*”. Nesse metamodelo foram criados os relacionamentos entre a metaclasse “*MasmlClassDiagram*” e as metaclasses (i) *Class*, (ii) *ObjectRoleClass*, (iii) *AgentRoleClass*, (iv) *Association*, (v) *Dependency*, (vi) *Generalization*, (vii) *Aggregation*, (viii) *Control*. Note que as metaclasses citadas fazem parte do metamodelo do Diagrama de Papel.

Além disso, foi necessário adicionar dois novos atributos (*sourceMultiplicity* e *targetMultiplicity*) à metaclasse *Control* referentes à multiplicidade, para que a multiplicidade possa ser representada por este relacionamento.

A partir do “*masml.ecore*” estendido são gerados todos os outros metamodelos contidos no GMF.

3.2. Extensão do *Domain Gen Model*

O “*masml.ecore*” foi derivado (transformado) em um metamodelo mais específico, o “*masmlRole.genmodel*”, a partir do qual as classes Java (códigos) são geradas. Os códigos gerados são: (i) *Model Code* (masml, masml.impl e masml.util); (ii) *Edit Code* (br.puc-rio.inf.les.masml.edit); (iii) *Editor Code* (br.puc-rio.inf.les.masml.editor); (iv)

Test Code (br.puc-rio.inf.les.masml.tests). Essas classes são usadas no projeto executável e são necessárias para a geração da ferramenta MAS-ML *tool*.

3.3. Extensão do *Graphical Def Model*

O “*masmlRole.gmfgraph*”, derivado a partir do “*masml.ecore*”, é responsável pela criação dos componentes gráficos. Durante o processo de derivação é selecionada a classe principal ou raiz (*MasmlClassDiagram*), em seguida são marcados os componentes requeridos no Diagrama de Papel, indicando o tipo de componente em termos de forma, relacionamento ou texto.

Após a derivação, foram criados os seguintes componentes: (i) *Nodes* (representam as entidades e os compartimentos pertencentes ao diagrama de papel), (ii) *Conections* (representam os relacionamentos pertencentes ao diagrama), (iii) *Figures* (representam as figuras de cada entidade, relacionamentos e compartimentos pertencentes ao diagrama) e (iv) os *Diagram Labels* (representam os rótulos para que o usuário receba um *feedback* de cada nó) no “*masmlRole.gmfgraph*”. Adicionalmente, foram identificados e removidos os elementos que estão no metamodelo de MAS-ML do “*masml.ecore*” e que não fazem parte do diagrama de papel.

Nas entidades que fazem parte do diagrama de papel foram adicionados os *Compartments*, que representam os compartimentos como eles devem ser vistos na ferramenta. Na entidade *AgentRoleClass* foram adicionados os compartimentos *Belief*, *Goal*, *Right*, *Duty* e *Protocol*. De forma semelhante, na entidade *Class* e na entidade *Object RoleClass*, foram adicionados os compartimentos *Operation* e *Property*. Na Figura 2 são apresentados os compartimentos da entidade *AgentRoleClass*.

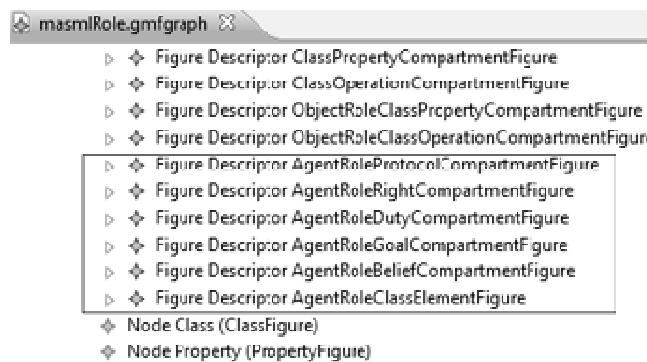


Fig.2. Compartimentos adicionados na entidade *agentRoleClass*.

No *graphical def model*, são definidas as formas visuais das entidades do diagrama. Dentre as formas disponíveis, nenhuma delas era adequada para representar o relacionamento *control* de acordo com a representação proposta em [Silva, 2004], isto é, através de uma reta com um círculo em uma das extremidades. Desta forma, foi necessário criar uma nova classe Java (*CircleDecoration.java*) no pacote “*masml.util*”, a qual trata da representação do círculo. Assim, utilizando o círculo gerado por esta classe, é possível identificar quem é a entidade controladora (*controller*) do relacionamento.

Nos relacionamentos *Dependency*, *Generalization*, *Aggregation* e *Control* foram adicionadas as figuras correspondentes aos relacionamentos possibilitando a

diferenciação entre a entidade fonte e destino. Além disso, os relacionamentos *Association*, *Control* e *Aggregation* foram alterados para permitir a visualização das multiplicidades no diagrama de papel.

3.4. Extensão do *Tooling Def Model*

O “*masmlRole.gmftool*”, derivado a partir do “*masml.ecore*”, é responsável pela criação da paleta com os botões utilizados na ferramenta.

Durante o processo de derivação, é selecionada a classe principal ou raiz (*MasmlClassDiagram*), e em seguida, são marcados os componentes requeridos no Diagrama de Papel, indicando seu tipo (forma ou relacionamento).

Após a derivação, foram criados os botões no “*masmlRole.gmftool*”, identificando e removendo os elementos que não fazem parte do diagrama de papel.

Após a remoção, a paleta passou a ser constituída por dois grupos: entidades e relacionamentos. O primeiro grupo é constituído por entidades (*AgentRoleClass*, *ObjectRoleClass*, *Class*) e comportamentos (*Belief*, *Goal*, *Right*, *Duty* e *Protocol*, *Operation*, *Property*). O segundo grupo é constituído por relacionamentos (*Association*, *Dependency*, *Generalization*, *Aggregation*, *Control*). Também foram definidas e adicionadas as figuras referentes a cada botão da paleta.

3.5. Extensão do *Mapping Model*

O “*masmlRole.gmfmap*” é resultado da combinação entre os metamodelos: “*masml.ecore*”, “*masmlRole.gmgraph*” e “*masmlRole.gmftool*”. A combinação consiste na união de todas as informações contidas nesses três metamodelos. Com isso, é possível mapear cada elemento que compõe a paleta com o controlador e figura correspondentes. Durante o processo de combinação, é selecionada a classe principal ou raiz (*MasmlClassDiagram*). Em seguida, são identificados os elementos que são *Nodes* ou *Links* no diagrama de papel.

Após a combinação ser concluída, foram criados os *Nodes* e os *Links* correspondentes às entidades e relacionamentos, respectivamente. Em seguida, foi feita uma verificação no “*masmlRole.gmfmap*” e houve a necessidade de adicionar uma série de elementos, tanto nos *Nodes* como nos *Links*, para garantir que o diagrama de papel pudesse ser modelado corretamente.

Nos *Nodes*, foi feito o mapeamento dos elementos e seus comportamentos. No *Node* referente à entidade *Class*, foi adicionado em seu mapeamento os comportamentos *Property* e *Operation*, e suas devidas referências. Similarmente, no *Node* referente à entidade *ObjectRoleClass* foram inseridos os dois comportamentos citados. No *Node* referente à entidade *AgentRoleClass*, foi adicionado em seu mapeamento os comportamentos *Belief*, *Goal*, *Duty*, *Right*, *Protocol*, e suas devidas referências. Nas referências de cada comportamento foram feitas configurações, tais como o tipo dos elementos que seriam adicionados dentro do comportamento e a sua formatação.

Nos *Links* foi feito o mapeamento dos relacionamentos. No *Link* correspondente ao *Generalization* foi adicionada uma *Feature Label* que é responsável por mostrar o nome do relacionamento dentro do Diagrama de Papel. No *Link* correspondente ao *Control* foi determinado que a criação do relacionamento só fosse possível entre as entidades papel de agente (*AgentRoleClass*). Além disso, ainda no relacionamento

control, foram adicionadas três *Feature Label* responsáveis por mostrar o nome do relacionamento, multiplicidade da fonte e a multiplicidade do alvo do relacionamento dentro do Diagrama de Papel. No *Link* correspondente ao *Dependency* foi adicionada uma *Feature Label* é responsável por mostrar o nome do relacionamento *Dependency* dentro do Diagrama de Papel. No *Link* correspondente ao *Association* foram adicionadas cinco *Feature Label* as quais são responsáveis por mostrar a multiplicidade da fonte, a multiplicidade do alvo, o nome da fonte, o nome do alvo e o nome do relacionamento dentro do diagrama de papel. As mesmas cinco *Feature Label* foram adicionadas no *Link* correspondente ao *Aggregation*.

Além do mapeamento, foi preciso incorporar ao “*masmlRole.gmfmap*” a definição de algumas regras responsáveis pela validação da corretude dos modelos. Mais especificamente, estas regras se referem ao nome, de forma a garantir que toda entidade deve ter um nome, ao *Duty* e ao *Right*, estabelecendo que pelo menos um dos dois deva existir no papel de agente, e em relação ao *Protocol*, garantindo que esse compartimento sempre exista no papel de agente.

As regras adicionadas foram implementadas na linguagem OCL (*Object Constraint Language*) [OCL, 2011] que é uma linguagem declarativa para descrever as regras que se aplicam aos modelos. As novas regras criadas são descritas no Quadro 1, no qual pode ser vista a identificação da regra na ferramenta, propósito e sua definição em OCL.

Quadro I. Regras de Validação dos Modelos

Regra	Propósito e Definição em OCL
Regra 1	Todos os elementos do modelo devem ter um nome. $\text{name.size()} > 0$
Regra 2	Se não existe o Duty, então existe o Right. $\text{self.ownedDuty->isEmpty()} = \text{true} \text{ implies } \text{self.ownedRight->isEmpty()} = \text{false}$
Regra 3	Se não existe o Right, então existe o Duty. $\text{self.ownedRight->isEmpty()} = \text{true} \text{ implies } \text{self.ownedDuty->isEmpty()} = \text{false}$
Regra 4	O papel do agente deve ter protocolo. $(\text{self.ownedRight->isEmpty()} = \text{false} \text{ or } \text{self.ownedDuty->isEmpty()} = \text{false}) \text{ implies } \text{self.protocol->isEmpty()} = \text{false}$

3.6. Extensão do *Diagram Editor Gen Model*

Com a conclusão do mapeamento, é criado o “*masmlRole.gmfgen*”. Em seguida é feita a transformação e a geração (*Generate diagram editor*) do código executável do projeto. Os modelos gerados a partir da ferramenta podem ser validados aplicando as restrições OCL implementadas, as quais podem ser acessadas através do menu *edit* e da opção *validate*. Caso alguma regra seja violada, o elemento que apresenta o problema é assinalado, e o detalhamento dos problemas é apresentado através do recurso *problems* do Eclipse.

A seguir é apresentado um estudo de caso onde é ilustrada a representação de entidades e relacionamentos modelados através da ferramenta MAS-ML *tool* de acordo com a especificação na linguagem.

4. Estudo de Caso

Como exemplo de modelagem para esse trabalho foi escolhido o ambiente de aprendizagem *Moodle* [MOODLE, 2011]. Esse ambiente é utilizado por instituições de ensino superior e médio como ambiente de aprendizagem colaborativa. O *Moodle* facilita a comunicação entre professores e alunos, para que eles possam trocar informações de maneira mais fácil, compartilhando os diversos recursos do ambiente via internet.

O *Moodle* assume que as pessoas aprendem melhor quando engajadas de forma colaborativa em um processo social de construção de conhecimento. Com base nas suas características e funções no sistema, uma variedade de agentes pode ser definida, onde cada um deles precisa desempenhar pelo menos um papel na organização. No ambiente *Moodle*, o papel Coordenador é responsável por administrar todo o ambiente e gerenciar a troca de informações entre as demais entidades. Adicionalmente, outros papéis de agente foram incorporados na modelagem do sistema: o CompanheiroAprendizagem, o Pedagógico, o BuscadorDeInformações, o FormadordeGrupos e o AuxiliarDeUsabilidade.

De maneira geral, foi usado o relacionamento *Association* entre as classes de papel de agente, uma vez que esse relacionamento é o menos específico de todos e pode ser aplicado em qualquer caso. O uso do relacionamento *Control* é exemplificado entre as classes de papel de agente Coordenador e Pedagógico, uma vez que o agente que executa o papel Coordenador controla o agente que executa o papel Pedagógico.

Além desses papéis de agente, na modelagem foi adicionada a classe Disciplina, que mantém as informações que podem ser usadas pelo papel do agente Pedagógico.

Também foram adicionadas classes de papéis de objeto referentes às ferramentas de busca, criadas para fornecer ao papel de agente BuscadorDeInformações as informações que ele precisa. Na Figura 3 pode ser vista a modelagem dos papéis exercidos no ambiente *Moodle* e seus relacionamentos.

5. Trabalhos Relacionados

Um ponto chave em relação às ferramentas de modelagem é que, normalmente, estas são projetadas com foco no suporte a uma linguagem de modelagem específica. Assim, as vantagens e desvantagens dessas linguagens são propagadas para as ferramentas que as implementam. Dentre as linguagens de modelagem que possuem ferramenta de suporte podemos destacar: (i) AUML [Odell, 2000], (ii) ANote [Choren e Lucena, 2004] e (iii) MAS-ML [Silva, 2004].

A AUML é uma linguagem que estendeu a UML para que pudesse modelar aspectos relacionados a agentes. No entanto, na AUML não são descritas as propriedades das organizações e dos papéis, além disso, os papéis de agentes e de objetos são definidos pela mesma metaclasses, o que segundo [Silva, 2004] não poderia acontecer. Com isso, a sua ferramenta de suporte não é capaz de modelar corretamente os papéis de um sistema. O ANote, apesar de possuir um *plugin* para a geração de código na plataforma eclipse chamado Albatroz, não identifica papéis. Portanto, não é possível modelá-los nessa ferramenta.

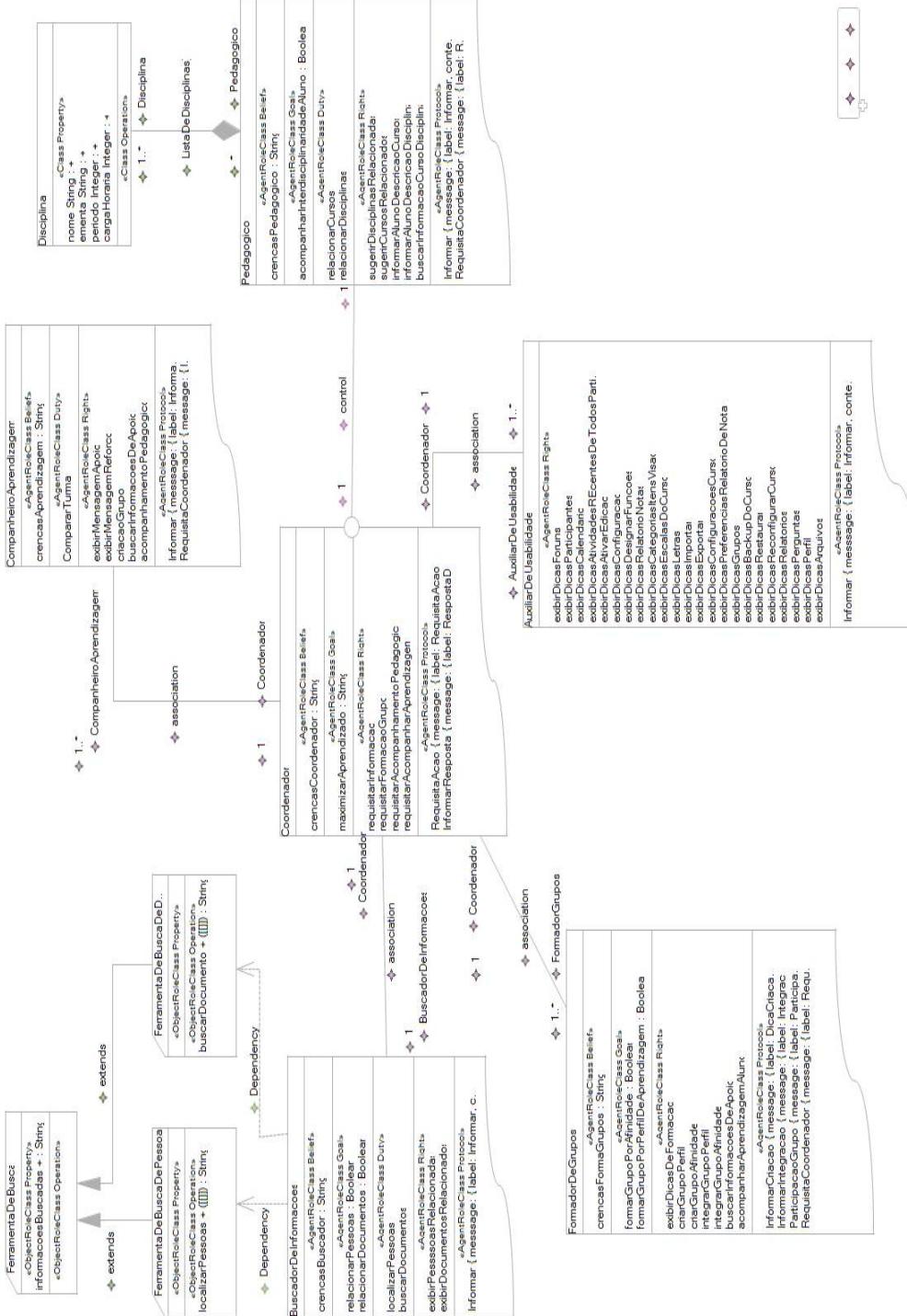


Fig.3. Diagrama dos papéis exercidos no ambiente Moodle gerado com MAS-ML tool.

Considerando as ferramentas de modelagem já existentes relacionadas a MAS-ML, *VisualAgent* [De Maria, 2005] é um ambiente de desenvolvimento que ajuda o desenvolvedor na produção de especificações, projeto, e implementação de SMAs.

O *VisualAgent* é baseado no metamodelo original da MAS-ML, e consequentemente, o suporte para à modelagem de agentes com diferentes arquiteturas internas é limitado. Além de também não fornecer suporte à modelagem de todos os diagramas descritos na linguagem de modelagem. Adicionalmente, a ferramenta *VisualAgent* não possui nenhum mecanismo que possibilite a checagem pela corretude dos modelos. A ausência desse recurso no *VisualAgent* pode comprometer a qualidade dos modelos elaborados e consequentemente, a do código gerado a partir de tais modelos. Além disso, a falta de documentação e o difícil acesso ao código fonte dificultam a continuidade do projeto.

Com a evolução proposta em MAS-ML *tool*, os três diagramas estáticos previstos para a modelagem de SMA são contemplados em consistência com a versão mais atual da linguagem possibilitando a modelagem das diversas arquiteturas de agentes. Adicionalmente, a ferramenta incorpora um mecanismo para a checagem de correção dos modelos gerados responsável pela verificação com base nas restrições estabelecidas em nível de linguagem.

6. Conclusão e Trabalhos Futuros

A função do papel no contexto de SMA é orientar ou restringir o comportamento da entidade que está incumbida a exercê-lo. O diagrama de papéis é importante no contexto de modelagem de SMAs por ser capaz de mostrar os relacionamentos existentes entre os papéis de agente no sistema. Neste trabalho foi apresentada uma extensão da ferramenta MAS-ML *tool* para possibilitar a modelagem do diagrama de papéis definido na linguagem MAS-ML de acordo com a versão 2.0. Com isso, os três diagramas estáticos previstos pela linguagem podem ser gerados através da ferramenta.

Com a extensão da ferramenta MAS-ML *tool*, é possível exibir todas as interações entre as entidades pertencentes ao diagrama de papéis que foram definidas na MAS-ML 2.0. Adicionalmente, a extensão proposta prevê a validação da corretude do diagrama gerado de forma a verificar a consistência da sua construção, reduzindo a ocorrência de erros e tornando a modelagem mais segura.

Existem alguns trabalhos futuros previstos no intuito de dar continuidade ao projeto de evolução da ferramenta apresentado nesse artigo. Dentre eles podem ser citados: (i) melhoramentos na representação gráfica dos construtores de acordo com a representação proposta pela linguagem MAS-ML; (ii) implementação dos diagramas dinâmicos na ferramenta MAS-ML *tool*, a saber: diagrama de sequência e atividades; (iii) geração de código a partir dos diagramas gerados pela ferramenta MAS-ML *tool* e (iv) unir todos os diagramas criados em um único projeto executável, pois os diagramas foram criados de forma separada na ferramenta (cada diagrama tem seu executável).

Agradecimentos

A. R. Feijó e F. R. O. Lima agradecem o apoio financeiro do CNPq/Brasil e da Funcap, respectivamente. A. R. Feijó, F. J. A. Maia e F. R. O. Lima agradecem aos professores orientadores M. I. Cortés e E. J. T. Gonçalves.

Referências

Choren, R. Lucena, C. “*Agent-Oriented Modeling Using ANote*”, *3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS*

2004), 3rd; *The Institution of Electrical Engineers, IEE, Stevenage, UK*, 2004, pp. 74-80, ISBN: 0-86341-431-1, May 24-25 , 2004.

De Maria, B. A. V. T. da Silva, C. J. P. Lucena, R. Choren, “*VisualAgent: A software development environment for multi-agent systems*,” *Proceedings of the 19º Simpósio Brasileiro de Engenharia de Software (SBES 2005), Tool Track*, Uberlândia, MG, Brazil, October 3-7, 2005.

Eclipse, “*Eclipse Platform*,” disponível em:<<http://www.eclipse.org>>, acessado em 2 de Junho de 2011.

EMF, disponível em:<<http://www.eclipse.org/modeling/emf/>>, acessado em 7 de Junho de 2011.

Farias, K. I. Nunes, V. T. da Silva, C. J. P. de Lucena, “*MAS-ML Tool: Um ambiente de modelagem de sistemas multi-agentes*,” *Fifth Workshop on Software Engineering for Agent-oriented Systems (SEAS@SBES 09), Brazil*, 2009.

GEF, disponível em:<<http://www.eclipse.org/gef/>>, acessado em 7 de Junho de 2011.

GMF, disponível em:<<http://www.eclipse.org/modeling/gmf/>>, acessado em 7 de Junho de 2011.

Gonçalves, E. J. T. K. Farias, M. I. Cortés, V. T. da Silva, R. G. F. Feitosa, “Modelagem de organizações de agentes inteligentes: uma extensão da MAS-ML Tool”, *1st Workshop on Autonomous Software Systems September*, 27, 2010, Salvador – Bahia – Brasil.

Jennings, N. “*Coordination Techniques for Distributed Artificial Intelligence*,” In: *Foundations of Distributed Artificial Intelligence*, pp. 187-210, Wiley, 1996.

Lind, J. “*Issues in Agent-Oriented Software Engineering*”. In: Ciancarini P. e Wooldridge M., LNCS 1957, Germany, Springer, p.45-58, 2001.

MOODLE, disponível em:<<http://www.moodle.org.br>>, acessado em 10 de Junho de 2011.

OCL, disponível em:<<http://www.eclipse.org/modeling/mdt/?project=ocl>>, acessado em 20 de Junho de 2011.

Odell, J. Parunak, H. V. D. Bauer, B. “*Extending UML for Agents*”. Proc. of the Agent-Oriented Information Systems Workshop (AOIS’00) at the 17th National conference on Artificial Intelligence (AII’00) (3-17), 2000.

OMG, “*UML: unified modeling language specification*,” versão 2.2, disponível em:<<http://www.uml.org>>, acessado em 2 de Junho de 2011.

Russell, S. P. Norvig, “Inteligência artificial: uma abordagem moderna,” 2ª Ed. Prentice-Hall: São Paulo, 2004.

Silva, V. T. “Uma linguagem de modelagem para sistemas multi-agentes baseada em um framework conceitual para agentes e objetos,” Tese de doutorado. Rio de Janeiro: PUC, Departamento de Informática, 2004.

Silva, V. T. R. Choren, C. J. P. Lucena, “*MAS-ML: A Multi-Agent System Modeling Language*,” In: *Conference on Object-oriented programming, systems, languages, and applications*, 18th annual ACM SIGPLAN; CA, USA, ACM Press, pp. 304-305, 2007.

Planejamento de Rotas de Robôs Móveis: Estudo da Viabilidade de Uma Abordagem Baseada em Algoritmos Genéticos em um Ambiente Multiagente

Tauã M. Cabreira¹, Marilton S. de Aguiar^{1,3}, Graçaliz P. Dimuro^{1,2}

¹Programa de Pós-Graduação em Modelagem Computacional

²Programa de Pós-Graduação em Computação
Universidade Federal do Rio Grande (FURG)
96.203-900 - Rio Grande - RS - Brasil

³Programa de Pós-Graduação em Computação
Universidade Federal de Pelotas (UFPel)
96.010-610 - Pelotas - RS - Brasil

tsiad.taua@gmail.com, marilton@inf.ufpel.edu.br, gracaliz@gmail.com

Abstract. This paper describes an approach of genetic algorithms for the path planning of mobile robots in static and dynamic environments. With the software Netlogo, used in simulations of multi-agent applications, we developed a seminal model for the given problem. The model, which contains a robot and scenarios with or without obstacles, is responsible for determining the best path used by a robot to achieve the objective state in a shorter number of steps, avoiding collisions. Additionally, a performance evaluation of this model in comparison with A* algorithm is presented.

Resumo. Este artigo descreve uma abordagem de algoritmos genéticos para o planejamento de rotas de robôs móveis em ambientes estáticos e dinâmicos. Através do software Netlogo, usado em simulações de ambientes multiagentes, foi desenvolvido um modelo seminal para o problema em questão. O modelo, composto por um robô e cenários sem ou com obstáculos fixos e móveis, é responsável por determinar a melhor rota para alcançar o estado objetivo no menor número de passos, evitando colisões com os obstáculos. Além disso, uma avaliação de desempenho deste modelo em comparação ao algoritmo A* é apresentada.

1. Introdução

O problema de planejamento de rotas de robôs móveis consiste em determinar uma rota para um robô, em um ambiente estático e/ou dinâmico, capaz de levá-lo do estado inicial ao estado objetivo através do melhor caminho, ou seja, o caminho mais curto, evitando possíveis colisões com obstáculos. Este problema já possui diversas abordagens sugeridas por diferentes pesquisadores, dentre as quais se pode citar a *Dead-reckoning*. No entanto, este e outros métodos apresentam ineficiências motivadas pelo acúmulo de erros (TU, 2003).

A abordagem de algoritmos genéticos surgiu como uma importante alternativa em diversos trabalhos recentes relacionados ao problema do planejamento de rotas de

robôs. Além da extensa possibilidade de variações, configurações e modificações, a abordagem de algoritmos genéticos possibilita ainda a agregação de outras técnicas que complementem e/ou aprimorem os resultados obtidos. Um Algoritmo Genético (AG) é uma variante de busca em feixe estocástica, na qual os estados sucessores são gerados pela combinação de dois estados pais, em vez de serem gerados pela modificação de um único estado (RUSSEL e NORVIG, 2003).

Os AG's foram inventados por John Holland nos anos 60, cujo principal objetivo não foi desenvolver algoritmos para solucionar problemas específicos, mas dedicar-se ao estudo formal do fenômeno de evolução, como ocorre na natureza, e desenvolver maneiras de importá-lo aos sistemas de computação (AGUIAR, 1998). Neste modelo, tem-se inicialmente um conjunto de estados (indivíduos) gerados aleatoriamente intitulado de população. Cada indivíduo é representado por um único cromossomo, contendo a codificação (genótipo) de um candidato à solução de um determinado problema (CONCILIO, 2000).

Em (SADATI e TAHERI, 2002), a abordagem de algoritmos genéticos foi combinada com o uso das Redes Neurais de Hopfield para evitar a colisão com obstáculos no processo de *crossover* entre dois caminhos distintos. Através do uso deste tipo de rede neural, existe a possibilidade de adicionar novos nodos na rota do robô, gerando desvios ao longo do trajeto e, evitando assim, a colisão com obstáculos. Já em (LEI, WANG e WU, 2006), o campo de potencial numérico foi agregado aos algoritmos genéticos, sendo levado em consideração o *feedback* da posição e dos movimentos dos obstáculos para evitar colisões através de replanejamento da rota.

Além da agregação de novas técnicas ao uso de algoritmos genéticos na resolução do problema de planejamento de rotas, também são realizadas mudanças nas características tradicionais do modelo dos AG's. Este é o caso do trabalho de (LI, TONG, XIE e ZHANG, 2006), onde um algoritmo genético híbrido é proposto com a finalidade de evitar uma convergência prematura no modelo. Para tal, uma estratégia de controle *fuzzy* é empregada para o ajuste das probabilidades de *crossover* e mutação, tornando o modelo autoadaptativo.

Conhecimentos adicionais sobre o problema em questão também podem ser empregados no modelo de algoritmos genéticos. No trabalho de (HU e YANG, 2004), o conhecimento do domínio foi incorporado em operadores especializados, combinados com técnicas de busca local. Neste modelo, foram desenvolvidos alguns operadores dedicados especialmente ao refinamento das soluções, onde se podem ajustar linhas entre dois pontos que atravessam um obstáculo, um ponto que esteja localizado justamente sobre um obstáculo ou até mesmo o posicionamento dos nodos.

Através dos trabalhos pesquisados e analisados, percebe-se o uso das mais variadas técnicas em conjunto com os algoritmos genéticos. A própria abordagem dos algoritmos genéticos muda drasticamente conforme a proposta de trabalho desenvolvida, variando na modelagem do ambiente, operadores utilizados, composição dos cromossomos, etc. A proposta deste trabalho é fazer uso de uma abordagem de algoritmos genéticos para o problema do planejamento de rotas de robôs móveis utilizando um ambiente de desenvolvimento multiagente.

Este trabalho está organizado como descrito a seguir. Na Seção 2, apresenta-se o modelo de planejamento de rotas de robôs, bem como a descrição detalhada da interface e das funcionalidades do mesmo. Na Seção 3, descrevem-se os testes de desempenho comparativos realizados entre o A* e o AG, além dos resultados obtidos através das simulações. Por fim, a Seção 4 conclui o trabalho e sinaliza a realização de trabalhos futuros.

2. O Modelo de Planejamento de Rotas de Robôs

A abordagem de algoritmos genéticos é uma técnica robusta e flexível, com eficiência comprovada na resolução de diversos tipos de problemas. Através do uso de AG, pretende-se desenvolver simulações em diversos tipos de ambientes estáticos e dinâmicos, visando analisar a capacidade de evitar colisões e de determinar a escolha da melhor rota.

A plataforma de desenvolvimento adotada foi o software *NetLogo*, um ambiente de desenvolvimento de modelos multiagentes, baseado na linguagem Logo, amplamente utilizado por estudantes, professores e pesquisadores (WILENSKY, 2011). Neste caso, fez-se uso dos agentes para a representação do robô e dos obstáculos, visto que o software possibilita, entre outras opções, a movimentação e interação entre os agentes, bem como a modelagem de um AG.

O modelo é composto por um robô, alguns obstáculos fixos e obstáculos móveis – a quantidade de obstáculos móveis pode ser alterada através da interface do aplicativo. Inicialmente, o robô projeta a melhor rota, baseando-se em um ambiente estático. Ao concluir o processo, o robô começa a realizar a sua rota, ao mesmo tempo em que os obstáculos começam a se movimentar, caracterizando um ambiente dinâmico. Ao encontrar um obstáculo pelo caminho, o robô recalcula a rota, redirecionando seu caminho para evitar as colisões. Ao atingir o estado objetivo, o robô para e o procedimento conclui-se.

2.1. O Fluxo da Simulação

Inicialmente, ao configurar o ambiente, é gerada uma população inicial aleatória de indivíduos, onde cada um destes indivíduos é composto por um *cromossomo* que constitui uma determinada rota. Todos os indivíduos que compõem a população são possíveis candidatos à solução do problema proposto.

Neste instante, no entanto, encontram-se no modelo apenas soluções geradas aleatoriamente sem nenhum critério de escolha. Encontram-se ainda no modelo obstáculos estáticos e móveis que poderão interferir na rota determinada para o robô. Este é o cenário inicial configurado a partir do botão SETUP da interface. Ao iniciar a simulação, através do botão GO, é dado início aos processos que constituem um algoritmo genético, que basicamente são a reprodução e a mutação, além da especificação da função de avaliação de cada novo indivíduo gerado. Os processos que constituem o algoritmo genético são executados por um determinado número de vezes, intitulado de gerações ou épocas. Ao término da execução destes procedimentos, o melhor indivíduo, ou seja, a melhor solução para o problema é retornada.

Em ambiente estático, onde os obstáculos estão imóveis, estes procedimentos seriam suficientes. Porém, o problema consiste em um ambiente estático e dinâmico, onde os obstáculos podem se movimentar livremente pelo ambiente, afetando diretamente a rota do robô. Desta forma, foram implementados procedimentos de detecção de obstáculos para evitar colisões do robô com os obstáculos. Portanto, a determinação da melhor rota para o robô, com ponto de partida e de chegada definidos, em um ambiente estático e dinâmico, consiste na busca pela rota mais breve e sem colisões, contendo ainda, a capacidade de identificação de obstáculos presentes no caminho e de recálculo da rota para alcançar o estado objetivo.

2.2. A Caracterização da Ferramenta

O ambiente de desenvolvimento do Netlogo é composto por três abas principais: *interface*, *information* e *procedures*. Na aba *interface*, pode-se controlar, executar e visualizar todas as ações do modelo. Por sua vez, na aba *information*, são acrescentadas todas as informações e características do modelo, bem como a explicação de seu funcionamento. E por fim, na aba *procedures*, têm-se todas as funções do modelo programadas na linguagem Logo. Nas seções que seguem, serão descritos os elementos que compõem a *interface* e as funções relacionadas a estes elementos implementadas na aba *procedures*. A explicação adotada na aba *information* dar-se-á ao longo do texto.

2.3. A Interface

Na aba *interface* do Netlogo, adicionaram-se diversos elementos e botões para a configuração, a execução e o controle da simulação do problema de planejamento de rota para robôs móveis, conforme apresenta a Figura 1.

Os botões SETUP e GO são os responsáveis pela configuração e execução do modelo. Inicialmente, o ambiente é composto por uma população (invisível) de candidatos à melhor rota e obstáculos fixos (cor laranja) e móveis (cor azul) distribuídos pelo mapa. Ao iniciar a simulação através do botão GO, o algoritmo genético é executado por um determinado número de gerações, que é definido pelo próprio usuário através do *slider* (botão que corre horizontalmente) GERACOES.

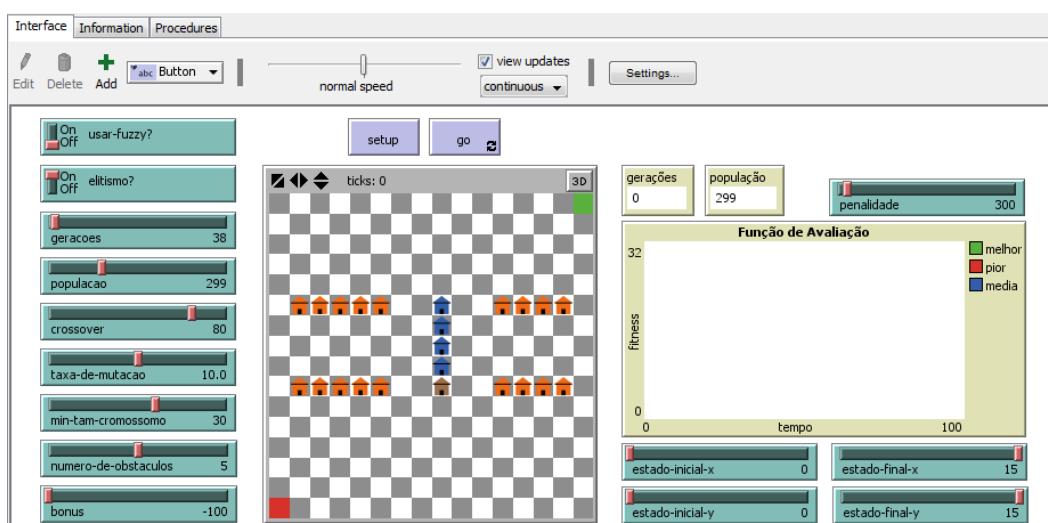


Figura 1. Interface da Ferramenta e Visão Geral do Netlogo

Além dos botões SETUP e GO, existem outros *sliders* cuja função é configurar os parâmetros adotados no algoritmo genético, tais como número de gerações, quantidade de indivíduos (população), taxa de *crossover* e de mutação, tamanho mínimo do cromossomo, índices de bônus e de penalidade da função de avaliação. Já os monitores e *plots* (gráficos) apresentam ao usuário informações pertinentes ao modelo. Neste caso, através dos monitores POPULACAO e GERACOES, é possível visualizar a quantidade de indivíduos do modelo e o progresso das gerações do algoritmo genético durante a execução da simulação. No *plot*, pode-se acompanhar o valor da função de *fitness* da melhor e da pior da solução através do tempo.

E por fim, têm-se os botões *switch* (chaves de liga-desliga), responsáveis por ativar ou desativar determinada característica do modelo. No modelo constam dois botões *switch*, ELITISMO? e USAR-FUZZY?. O primeiro controla a presença ou não de elitismo no algoritmo genético. Em caso afirmativo, o melhor indivíduo de cada geração é sempre repassado à geração seguinte. Se o elitismo estiver desativado, existe a possibilidade do melhor indivíduo de sua espécie em uma determinada época não ser repassado à próxima geração. Já o *switch* USAR-FUZZY? é responsável por ativar e desativar o uso da função *fuzzy* no recálculo da rota do robô.

2.4. As Funcionalidades

Na aba *procedures*, encontram-se todas as funcionalidades desenvolvidas. Para facilitar a compreensão do modelo, apresenta-se na Figura 2 um algoritmo representando as funcionalidades desenvolvidas no projeto.

O algoritmo inicia com a geração da população inicial do algoritmo genético. A quantidade de indivíduos da população é determinada pelo *slider* POPULACAO, que se encontra na interface do modelo. Cada indivíduo da população é gerado com a variável denominada *cromossomo*. Este *cromossomo* é um vetor binário que representa uma sequência de movimentos, onde cada movimento é codificado em três bits (necessários para codificar as oito possibilidades de movimentação) da seguinte forma: direita (000), acima-direita (001), acima (010), acima-esquerda (011), esquerda (100), abaixo-esquerda (101), abaixo (110) e abaixo-direita (111). Por exemplo, um *cromossomo*, que representa um percurso com 15 movimentos, composto por 15 genes, tem como fenótipo 0010010010010000010010010010011001001001000.

Os *cromossomos* podem possuir tamanhos distintos, iniciando como o tamanho determinado pelo *slider* MIN-TAM-CROMOSSOMO. Os movimentos que compõem os *cromossomos* também são escolhidos aleatoriamente. Logo em seguida, são realizados os movimentos presentes em cada cromossomo e calculado o valor da função de avaliação, também conhecida como *fitness*, de cada indivíduo.

A função de avaliação é responsável por quantificar a qualidade de cada indivíduo como uma possível solução para o problema proposto. Em um problema de maximização, quanto maior for o valor calculado pela função de avaliação, melhor será a solução encontrada. O problema do planejamento de rotas de robôs móveis consiste em um problema de minimização, pois se almeja determinar a melhor rota de deslocamento de um robô a partir de um ponto inicial até um estado objetivo. Portanto,

a melhor rota é determinada pela rota mais curta, ou seja, o menor cromossomo, desde que evite as colisões com os obstáculos.

```
1 Algoritmo
2
3 Gere a população de indivíduos aleatoriamente
4 Realize os movimentos presentes em cada cromossomo
5 Avalie o fitness de cada indivíduo ao final do trajeto
6
7 Repita o processo de AG pelo número de gerações
8     Repita o processo de crossover de acordo com a taxa definida
9         Selecione os pais através da "seleção por torneio"
10        Divilde em dois, numa posição randômica, cada um dos cromossomos dos pais
11        Gere dois novos indivíduos com a mescla do cromossomo dois pais
12
13     Se o elitismo estiver ativado
14         Selecione o melhor indivíduo
15         Repasse-o para a geração subsequente
16
17     Copie os demais indivíduos para a geração seguinte
18     Elimine a população antiga
19     Realize a mutação de acordo com a taxa definida
20     Realize os movimentos presentes em cada cromossomo
21     Avalie o fitness de cada indivíduo ao final do trajeto
22
23 Faça enquanto o menor fitness for >= 0
24     Repita os mesmos processos do AG
25     Adicione cromossomos randômicos a cada 10 gerações
26
27 Imprima a rota do melhor indivíduo
```

Figura 2. Pseudocódigo do Modelo

Para calcular o *fitness*, são executados todos os movimentos presentes no *cromossomo* de cada indivíduo. A cada movimento, verifica-se a presença de obstáculos pelo caminho, penalizando aqueles indivíduos cuja rota colide com os obstáculos. O procedimento funciona da seguinte forma: ao mover-se para um novo *patch*, de acordo com o movimento estipulado pelo *cromossomo*, é verificada a presença de obstáculos neste local. Em caso afirmativo, a candidata à solução de melhor rota é penalizada na função de avaliação de acordo com o valor estipulado na interface através do *slider* de penalidade.

Caso não haja obstáculos no *patch* selecionado para o movimento, uma nova verificação é realizada, mas desta vez relacionada ao estado objetivo. Se o *patch* escolhido for o estado objetivo a ser alcançado pela rota, a solução é bonificada (o valor desta bonificação é determinado pelo *slider* BONUS na interface) em sua função de avaliação. E finalmente, se o *patch* escolhido for apenas um *patch* comum, sem obstáculos ou estado objetivo presente, o valor da função de avaliação é apenas acrescido do valor de deslocamento do movimento.

Abaixo encontra-se a função de avaliação adotada para o problema:

$$\text{Fitness} = \sum_{i=0}^n \text{distância} * (1 + \text{peso}) + (15 - \text{posição}x + 15 - \text{posição}y)$$

onde: *distância* é o trecho percorrido no movimento (movimentos para frente, para trás ou para os lados consomem uma unidade de distância; já os movimentos diagonais consomem $\sqrt{2}$ unidades de distância); *peso* é o valor atribuído pela penalização ou bonificação (este valor será zero caso nenhuma das opções anteriores seja atribuída à variável); *posiçãox* e *posiçãoy* indicam a posição atual da solução em relação aos eixos x e y.

Após a realização dos movimentos presentes nos cromossomos e o cálculo da função de avaliação de todos os indivíduos da população inicial, dá-se início ao processo do algoritmo genético e de seus operadores de *crossover* e mutação. Com base na taxa de *crossover*, determinada pelo *slider CROSSOVER*, são gerados novos indivíduos através da reprodução ou cópia (ex.: Se a taxa de *crossover* estiver marcando 70, isto significa que 70% da nova população será gerada através da reprodução e os outros 30% serão meramente uma cópia dos indivíduos da atual população).

O processo de reprodução empregado é o da seleção por torneio. Neste processo, são selecionados aleatoriamente três indivíduos da população atual. O indivíduo com o melhor *fitness* dentre os três é escolhido como um dos *pais*. O mesmo procedimento é executado para a escolha do outro *pai* do novo indivíduo. Em seguida, o código genético (*cromossomo*) de cada um dos pais selecionados é dividido aleatoriamente em duas partes, num processo intitulado de *crossover* de um ponto.

Na sequência, dois novos indivíduos são gerados e parte do *cromossomo* de cada progenitor é adicionada ao *cromossomo* do novo indivíduo. Neste caso, o primeiro indivíduo gerado a partir da reprodução armazenará em seu *cromossomo*, a primeira parcela dos genes de seu pai e a segunda parcela dos genes de sua mãe. Por sua vez, o segundo indivíduo gerado herdará a segunda parte dos genes de seu pai e a primeira parte dos genes de sua mãe.

Ao término do processo de reprodução, os demais indivíduos da nova população são gerados por clonagem, ou seja, selecionados através do mesmo processo de seleção por torneio empregado no *crossover*, porém sem o cruzamento entre dois indivíduos. Sendo assim, o indivíduo escolhido é apenas copiado para a geração seguinte. Se o elitismo estiver ativado através do *switch ELITISMO?* presente na interface, o melhor indivíduo da população atual é copiado para a geração subsequente. Com a nova população criada, a população antiga é eliminada e o processo de mutação tem início.

A mutação é efetuada com base na taxa de mutação determinada pelo *slider TAXA-DE-MUTACAO* (ex.: se o slider estiver posicionado em 10, as chances de ocorrer mutação no indivíduo são de 10%). O processo de mutação consiste na substituição de um dos genes do *cromossomo*, ou seja, a alteração de um dos movimentos da rota do indivíduo. Escolhe-se randomicamente um dos genes do cromossomo e o substitui por outro gene, também escolhido arbitrariamente, que será composto por um dos oito movimentos possíveis.

Com o fim do processo de mutação, todos os indivíduos da população são submetidos à movimentação (através das rotas presentes em seus cromossomos) e ao cálculo da função de avaliação. O processo do algoritmo genético repete-se até que o

número de gerações determinado na interface seja atingido. Caso o estado objetivo não seja alcançado através da melhor rota obtida ao final do processo, a simulação continua sendo executada, adicionando a cada 10 gerações, novos indivíduos gerados aleatoriamente, com o objetivo de agregar diversificação à população e fugir de máximos locais.

Uma vez que o estado objetivo é alcançado na melhor solução, executa-se a rota final do robô. Os movimentos executados pelo robô são alternados com os movimentos realizados pelos obstáculos, que por sua vez, são acionados através de outra função. Os obstáculos movimentam-se como uma espécie de serpente, onde sua cabeça é identificada pela cor marrom e determina a direção do movimento dos obstáculos, sendo seguidas pelas demais partes do corpo. A cabeça seleciona um dos *patches* vizinhos (vizinhança de von Neumann) para deslocar-se. Se o *patch* selecionado já estiver sendo ocupado por um agente, a função é chamada novamente para a escolha de um novo vizinho. A função é executada recursivamente até que um vizinho disponível seja encontrado. Satisfeita esta condição, a cabeça move-se para este *patch* e o restante dos obstáculos acompanham o movimento.

Com o deslocamento aleatório dos obstáculos móveis e a presença de obstáculos fixos no ambiente, a rota adotada pelo robô pode não alcançar o estado objetivo, visto que o cenário muda a cada passo. Sendo assim, é necessário realizar uma verificação constante da proximidade dos obstáculos com a rota em questão. Esta verificação pode ser realizada com ou sem o uso de uma função *fuzzy*. Ao utilizar-se da função *fuzzy*, dá-se ao robô autonomia de antecipar a decisão de recálculo de rota, de acordo com algum grau de pertinência para situação de colisão iminente. O robô pode perceber até quatro patches à sua frente. Caso a função detecte a presença de obstáculos dentro deste perímetro de até quatro patches, um grau de pertinência para colisão é calculado e utilizado como a probabilidade de recálculo da rota do robô. Se os obstáculos forem detectados a uma distância de quatro patches do robô, a probabilidade de recálculo é de 40%; a uma distância de três patches, a probabilidade é de 60%; a uma distância de dois patches, a probabilidade é de 80%; a uma distância de um patch, a probabilidade é de 100% (neste caso, o recálculo é obrigatório, visto que o próximo movimento do robô irá fazê-lo colidir com o obstáculo).

Ao não utilizar a função *fuzzy* para detectar a presença de obstáculos no entorno da rota, o recálculo da mesma só é efetuado quando o obstáculo estiver no *patch* subsequente em que se encontra o robô. Independentemente da abordagem empregada, com ou sem *fuzzy*, o recálculo da rota irá sempre ocorrer para evitar a colisão do robô com os obstáculos.

No recálculo da rota, são realizados os mesmos procedimentos descritos anteriormente, mas desta vez, em um cenário completamente diferente. O recálculo da rota pode ser executado diversas vezes ao longo do trajeto do robô – a cada novo obstáculo encontrado pelo caminho, um novo recálculo de rota deverá ser efetuado. Desta forma, evitam-se as colisões e obtém-se a melhor rota para o robô. Ao atingir o estado objetivo, o *cromossomo* é editado, removendo-se todos os genes excedentes que se encontram após o gene que atingiu o estado objetivo (ex.: se um *cromossomo*

composto por 25 genes atingiu o estado objetivo no gene 23, os genes 24 e 25 são descartados).

Durante a simulação, são impressos no gráfico os valores da função de avaliação do melhor e o pior indivíduo da população de cada em função do tempo. Enquanto o gráfico gerado pelos piores indivíduos ao longo das gerações apresenta um aspecto altamente oscilatório, com grandes variações no valor da função de avaliação, o gráfico gerado pelos melhores indivíduos ao longo do tempo exibe um decrescimento no valor da função de avaliação, por vezes atingindo uma estabilidade em determinado ponto da simulação.

3. Simulações e Resultados

Para avaliar a viabilidade da proposta de planejamento de rotas de robôs móveis utilizando a abordagem de algoritmos genéticos, utilizou-se o algoritmo A* (*A star*) para a realização de testes de desempenho comparativos. O algoritmo A* foi desenvolvido por Peter Hart, Nils Nilsson e Bertram Raphael em 1968 e é amplamente utilizado em problemas de planejamento de rotas e grafos transversais, devido a sua alta *performance* – é um algoritmo completo e ótimo (RUSSEL e NORVIG, 2003).

O algoritmo genético utilizado nos testes de desempenho foi configurado da seguinte forma: população de 299 indivíduos, 80% de taxa de *crossover*, 20% de taxa de mutação e elitismo ativado. Durante a simulação, o AG é executado por pelo menos 38 gerações – caso o estado objetivo não seja alcançado nas 38 primeiras gerações, o modelo segue sendo executado até que o robô consiga atingir a sua meta.

Tabela 1. Medição da quantidade de movimentos realizados nas rotas

Cenários					
Sem Obstáculos		Obstáculos Fixos		Obstáculos Fixos e Móveis	
A*	AG	A*	AG	A*	AG
15	16	15	21	18	17*
15	18	15	30	21*	25****
15	16	15	21	18	25*
15	16	15	21	16*	22*
15	18	15	19	20**	23
15	16	15	20	16*	25
15	18	15	17	18	20***
15	20	15	21	18	23
15	15	15	18	17*	21
15	16	15	26	18	24
15	15	15	20	18	21
15	16	15	24	21*	20
15	16	15	20	21*	24*

15	19	15	24	18	29
15	16	15	20	18	25
15	15	15	21	18	30
15	17	15	18	18	25*****
15	16	15	20	18*	20*****
15	15	15	16	18	19*
15	16	15	19	18	25

Ao todo, foram realizados vinte simulações em três cenários distintos: um cenário sem obstáculos, um cenário com obstáculos fixos e um cenário com obstáculos fixos e móveis. Como parâmetro de comparação de desempenho dos dois modelos, fez-se uso do tamanho da rota – número de movimentos empregados pelo robô para atingir o estado objetivo. Na Tabela 1, apresentam-se os resultados obtidos durante os testes de desempenho comparativos entre o A* e o AG.

Para avaliação dos testes de desempenho realizados utilizou-se o Teste t de Student. Nos testes realizados com os três cenários elaborados, o A* apresentou uma diferença estatística extremamente significante em relação ao AG, ou seja, há 99,9% de certeza de que o A* foi superior ao AG nos testes realizados. Comparando apenas os resultados obtidos nos testes de desempenho onde houve o recálculo da rota em ambientes com obstáculos fixos e móveis (os asteriscos na tabela de resultados representam o número de recálculos realizados em cada rota), a diferença entre os dois modelos é estatisticamente significante, ou seja, 95% de certeza.

Apesar dos resultados adversos apresentados pela abordagem do AG em relação ao A*, há um indicativo de melhora nos resultados do AG à medida que cresce a complexidade do cenário. Por exemplo, no cenário 3 (com obstáculos fixos e móveis) as rotas calculadas pelo AG foram, em média, 26% maiores do que as rotas calculadas com o A*. Entretanto, ao serem levadas em consideração apenas as situações em que houve recálculo de rota, a rotas calculadas pelo AG foram, em média, 19% maiores. Isso leva a crer que em cenários maiores (nos testes o cenário era de tamanho 15x15), com maior quantidade de obstáculos e, por conseguinte, com maior probabilidade de colisão, o modelo AG emparelharia com o A* na qualidade da rota calculada.

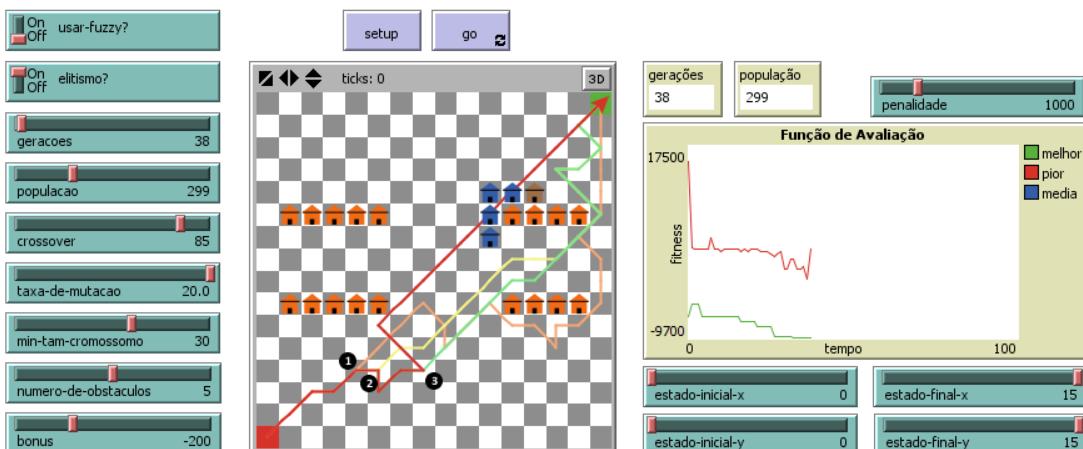


Figura 3. Solução final gerada através do AG após três recálculos de rota

No exemplo demonstrado pela Figura 3, pode-se constatar o encontro do robô com os obstáculos durante a execução de sua rota em três pontos distintos – pontos pretos 1, 2 e 3. Nestes momentos, a rota é recalculada para evitar a colisão do robô com os obstáculos. A cada recálculo de rota, o robô encontra um novo caminho livre de obstáculos. No entanto, nos dois primeiros recálculos, os obstáculos acabaram cruzando o caminho do robô novamente, forçando-o a buscar uma rota alternativa. Após o terceiro recálculo, o robô encontrou uma trajetória e alcançou o estado objetivo sem colidir com nenhum obstáculo.

4. Conclusões

Após o desenvolvimento do modelo proposto e da análise dos resultados obtidos, conclui-se que a abordagem de algoritmos genéticos para o problema de planejamento de rotas de robôs móveis em ambientes estáticos e dinâmicos apresentou-se com uma alternativa viável para a solução do problema em questão.

Através da abordagem de algoritmos genéticos, foi possível modelar soluções para o problema através dos *cromossomos* compostos pelos movimentos do robô no ambiente. A partir de soluções geradas arbitrariamente, os processos de *crossover* e mutação, baseados na função de avaliação de cada indivíduo, foram capazes de evoluir as candidatas à solução do problema, atingindo uma solução satisfatória para cada uma das simulações.

A utilização do *software* NetLogo também foi de suma importância no desenvolvimento do modelo proposto, pois possibilitou a modelagem de um ambiente dinâmico, composto por obstáculos móveis, onde o robô é capaz de detectar os obstáculos e evitar possíveis colisões. Através da interação entre os agentes do ambiente, o robô pode perceber os obstáculos em seu caminho e recalcular a sua trajetória para alcançar o estado objetivo.

O modelo de AG empregado neste trabalho ainda é um modelo em estágio seminal, que faz uso apenas dos operadores básicos de *crossover* e mutação. Conforme foi explanado na introdução do trabalho, o algoritmo genético é uma poderosa ferramenta capaz de ser combinada com diferentes técnicas para aprimorar o seu desempenho, tais como Redes Neurais de Hopfield e Campo de Potencial Numérico. Além disso, é possível acrescentar novos operadores mais robustos, capazes de elevar a *performance* do modelo.

Apesar da constatação da superioridade do A* em relação ao AG nos testes desempenho realizados, pode-se esperar uma melhora no modelo de AG em modelos mais complexos – cenário com obstáculos fixos e móveis envolvendo o recálculo da rota. Os cenários empregados nos testes favorecem o modelo do A*, pois este é um algoritmo ótimo. No entanto, este tipo de algoritmo tende a ter seu desempenho reduzido em ambientes mais complexos. Desta forma, em trabalhos futuros, pretende-se aperfeiçoar o modelo de AG com a adição de novos operadores e técnicas de refinamento, submetendo os modelos utilizados a testes de desempenho em ambientes maiores e mais complexos, com um número superior de obstáculos fixos e móveis, além da adição de outros robôs no ambiente, caracterizando um ambiente multiagente.

Agradecimentos

Os autores deste trabalho agradecem à CAPES pelo auxílio na forma de bolsa de mestrado REUNI. Este trabalho é parcialmente financiado pelo CNPq (Proc. 476234/2011-5, 560118/10-4,305131/2010-9) e FAPERGS (Proc. 11/0872-3).

Referências

- AGUIAR, M. Análise Formal da Complexidade de Algoritmos Genéticos. Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil, 1998. (Dissertação de Mestrado)
- CONCILIO, R. Contribuições à Solução de Problemas de Escalonamento pela Aplicação Conjunta de Computação Evolutiva e Otimização com Restrições. Universidade Estadual de Campinas, Campinas, SP, Brasil, 2000. (Dissertação de Mestrado)
- HU, Y.; YANG, S. X. A knowledge Based Genetic Algorithm for Path Planning of a Mobile Robot. Nova Orleans, EUA, 2004.
- LI, Q.; TONG, X.; XIE, S.; ZHANG, Y. Optimum Path Planning for Mobile Robots Based on a Hybrid Genetic Algorithm. China, 2006.
- LIN, L.; WANG, H.; WU, Q. Improved Genetic Algorithms Based Path Planning of Mobile Robot Under Dynamic Unknown Environment. Louyang, China, 2006.
- RUSSEL, S. J.; NORVIG. Artificial Intelligence: A Modern Approach. Reading: Prentice Hall, 2003.
- SADATI, N.; TAHERI, J. Genetic Algorithm in Robot Path Planning in Crisp and Fuzzified Environments. Tehran, Irã, 2002.
- TU, J.; YANG, S. X. Genetic Algorithm Based Path Planning for a Mobile Robot. Taipei, Taiwan, 2003.
- WILENSKY, URI. Guia do Usuário: Netlogo. Disponível em <http://ccl.northwestern.edu/netlogo/docs>. Acessado em 21 de Outubro de 2011.

Simulação do Trânsito no Centro da Cidade do Rio Grande/RS

Josimara de Ávila Silveira, Felipe Neves da Silva, Leonardo Martins Rodrigues

¹Programa de Pós Graduação em Modelagem Computacional

Universidade Federal do Rio Grande (FURG)

Av. Itália, Km 8 – Campus Carreiros – 96.201-900 – Rio Grande – RS – Brazil

Abstract. *Large cities suffer with the increase of the number of vehicles on their streets. In this situation, the city traffic managers try to find solutions to avoid traffic congestion and improve traffic flow in these locals. In this work, we simulated the central area of the Rio Grande City (Rio Grande do Sul – Brazil) to assess the impact on the growing vehicles fleet in the city, mainly in the central zone. Thus, the goal is to check which streets are more affected by traffic jams and the interference level of traffic lights in the streets. We used the NetLogo software to model the simulation.*

Resumo. *É comum o aumento na quantidade de veículos nas grandes cidades. Diante disso, as secretarias municipais de trânsito tentam encontrar soluções para evitar engarrafamentos e melhorar a fluidez do trânsito nessas cidades. Neste trabalho foi simulada a zona central da cidade de Rio Grande/RS para avaliar o impacto quanto ao crescimento da frota de veículos na cidade, especialmente na zona central. Com isso, o objetivo é verificar quais ruas sofrem mais com engarrafamentos e qual o nível de interferência dos semáforos em tais ruas. Utilizou-se o software NetLogo para executar a simulação.*

1. Introdução

Com a população de Rio Grande em torno de 200 mil habitantes, a cidade enfrenta sérios problemas de trânsito. Diversos fatores explicam esses problemas, como o crescimento da população devido, principalmente, ao pólo naval presente na região, bem como pelo aumento da frota de veículos que acompanha o aumento no poder de compra da população através das facilidades nas linhas de crédito. Estatísticas apontam que em apenas três anos a frota de veículos na cidade aumentou em quase 50%, saltando de 40 para 60 mil. Com isso, a cidade alcançou a média de um automóvel para cada três habitantes [Marchioro 2012].

Na tentativa de solucionar os problemas no tráfego de veículos, a Secretaria dos Transportes promove mudanças periódicas, como alterações nos sentidos de ruas, troca de semáforos por rótulas, implantação de Zona Azul (estacionamento rotativo) nas principais ruas do centro, etc. Porém, observações indicam que tais medidas são apenas paliativas, dado que em determinados horários algumas ruas sofrem com engarrafamentos e trânsito muito lento.

NetLogo é uma linguagem de programação multiagente com um ambiente de modelagem de sistemas multiagente integrado. Esse ambiente de desenvolvimento habilita a exploração de fenômenos emergentes [NetLogo 2011].

Além disso, o *software* vem acompanhado com uma extensa biblioteca de modelos, incluindo uma variedade de domínios como: economia, biologia, física, química, psicologia, dinâmica de sistemas, ciências sociais e etc.

A utilização de agentes nesta simulação é importante, pois apresentam características que facilitam a execução da simulação. Dentre elas, podem-se destacar [WOOLDRIDGE and JENNINGS 1995]: a *autonomia* é o fato do agente atuar independentemente da interferência do usuário; a *habilidade social* indica uma interação com outros agentes ou, até mesmo, com seres humanos diante de uma linguagem de comunicação; a *reatividade* ocorre diante de mudanças no ambiente, levando o agente a modificar o seu estado atual; a *proatividade* é a capacidade do agente interagir com o ambiente de acordo com sua própria percepção; a *continuidade temporal* induz o agente a estar sempre executando algum processo, podendo ser em *foreground* ou *background*.

Tendo em vista os assuntos abordados anteriormente, o objetivo do trabalho é implementar um código na linguagem NetLogo que realiza a simulação do trânsito na zona central da cidade de Rio Grande, tomando como base duas importantes ruas: 24 de Maio e Gen. Neto. Ou seja, o trânsito da cidade será através dos cruzamentos, semáforos e ruas existentes. Dessa forma, tentar-se-á verificar as causas da lentidão nessas ruas e suas redondezas através da remoção de semáforos, simulação de horário de “pico”, etc.

A ideia para este trabalho surgiu a partir de um exemplo existente na biblioteca de modelos do *software* NetLogo. Dessa forma, encontrou-se o modelo denominado “Traffic Grid” na seção de Ciências Sociais e, então, surgiu a motivação para realizar um trabalho na mesma área, porém, atribuindo um foco específico, neste caso um exemplo do dia-a-dia de nossa cidade. É importante ressaltar que o código desenvolvido em nenhum momento utilizou partes ou funções presentes no modelo existente, sendo implementado um código totalmente novo.

Alguns trabalhos tratam dos assuntos mencionados. Dentre eles, pode-se destacar [Bazzan 2005]. Em seu artigo, a autora trata sobre problemas de coordenação dos sinais de trânsito em cruzamentos. Dessa forma, foi proposta a descentralização dos sistemas atuais através da utilização de técnicas de inteligência artificial distribuída e sistemas multiagentes. Segundo a autora, o conceito de coordenação utilizado supera algumas desvantagens com relação aos métodos atualmente utilizados.

Este artigo está estruturado da seguinte forma. Na Seção 2 o modelo Traffic Grid é apresentado. A Seção 3 apresenta o modelo proposto para a simulação. Já a Seção 4 apresenta alguns resultados obtidos através das simulações. Por fim, a Seção 5 apresenta a conclusão deste trabalho e ideias para trabalhos futuros.

2. Modelo Traffic Grid

Existe na biblioteca de modelos do NetLogo uma implementação, na área de Ciências Sociais, chamada “Traffic Grid”. Nesse modelo é possível controlar os semáforos e outras variáveis, tais como limite de velocidade e o número de automóveis no ambiente. Com isso, é possível explorar a dinâmica do tráfego, sendo interessante para diversas áreas de estudo, desde cálculo até ciências sociais.

O objetivo do modelo é fazer com que o utilizador da simulação desenvolva estratégias para melhorar o tráfego, bem como entender as diferentes maneiras de medir a

qualidade do trânsito. A interface de simulação pode ser visualizada na Figura 1.

Neste exemplo, existem alguns elementos de interface para que o usuário possa alterar características da simulação, por exemplo: os *sliders* (barras deslizantes para ajustar o valor de alguma variável) “num-cars” e “speed-limit”. No primeiro, pode-se configurar a quantidade de *turtles* (agentes) no ambiente e o segundo é possível indicar qual a velocidade máxima permitida na simulação. Outros elementos, como os *plotters* (representação gráfica dos valores) e os *monitors* (indicação numérica que indica o valor atual de alguma variável), também são observados na interface.

A Figura 1 ilustra o que acontece ao longo da simulação deste modelo.

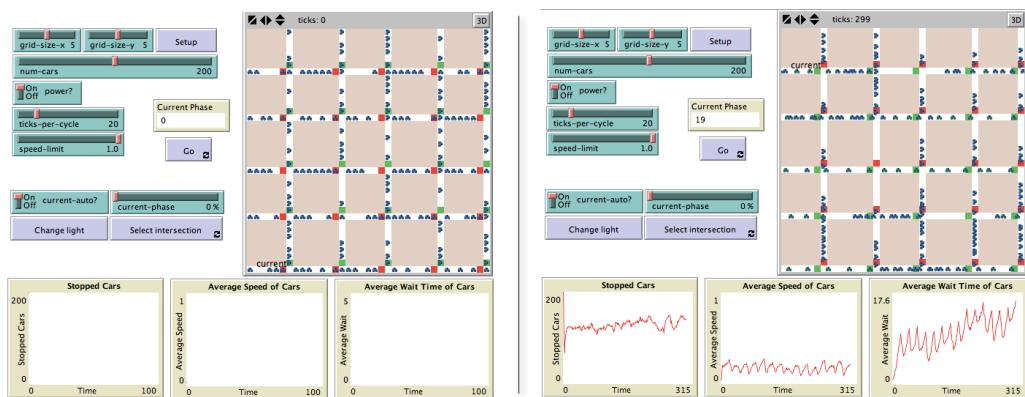


Figura 1. Acontecimentos conforme o andamento da simulação.

A ideia básica do código por trás da simulação, conforme mostra o Algoritmo 1, é a seguinte:

```

begin
    Atualizar semáforos.
    Zerar número de automóveis parados.
    for Cada veículo do
        Ajustar velocidade.
        Mover-se.
        Salvar dados.
        Ajustar cor de acordo com velocidade.
    end
    Desenhe nos plotters.
    Conte um tick no relógio.
end

```

Algoritmo 1: Pseudo-código utilizado no modelo *Traffic Grid*.

No cenário, os semáforos se alternam entre vermelho e verde fazendo com que os automóveis parem e se movam, respectivamente. Através de alterações nas variáveis existentes na interface é possível perceber a dinâmica da simulação pelas alterações no fluxo de veículos, geração de engarrafamentos e, inclusive, *deadlocks* (situação em que a simulação para pelo fato de que nenhum automóvel consegue se movimentar).

3. Modelo Proposto

O objetivo é realizar uma simulação semelhante à descrita anteriormente (Seção 2), porém, como foco em uma situação conhecida e muito discutida pela população de nossa cidade: o **trânsito** na zona central. Para isso, criou-se um ambiente que simula parte da zona central da cidade de Rio Grande, abrangendo desde a Rua Sen. Corrêa até a Rua Luiz Lorea, focando a observação nas ruas 24 de Maio e Gen. Neto. Existem duas ruas nas laterais do ambiente que apenas escoam o trânsito, pois, infelizmente, não é possível simular todo o mapa do centro da cidade devido à complexidade nas disposições das ruas. A Figura 2 mostra o mapa da região.

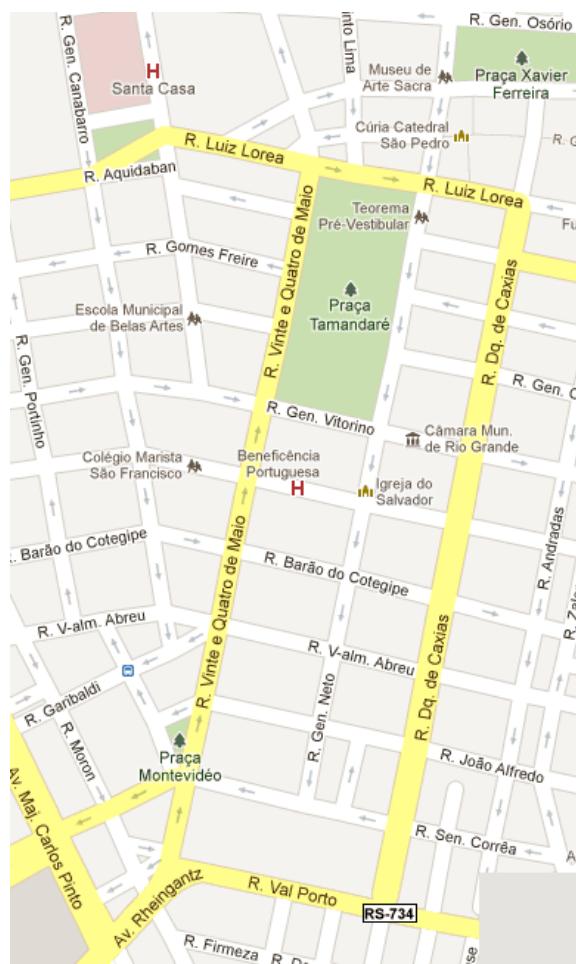


Figura 2. Mapa da zona central da cidade de Rio Grande (Fonte: Google Maps).

Os semáforos foram dispostos exatamente nos cruzamentos onde realmente existem. Os *patches* nas cores verdes que não são semáforos funcionam como se fossem placas de trânsito, indicando quais os sentidos que o automóvel que está em cima dele pode seguir. Por exemplo, se a cor for Verde=65.0, o veículo pode somente andar para frente; se a cor for Verde=65.1, o veículo pode seguir em frente ou dobrar a direita e assim por diante. Uma observação importante diz respeito às placas de cor 65.2 e 65.4, apesar de serem placas com os mesmos sentidos, representam situações diferentes dentro da simulação. Assim, quando a placa for da cor Verde=63.4, indica que o automóvel está em uma via de sentido único próximo a um cruzamento com uma via de sentido duplo,

mas que pode apenas seguir reto ou virar a esquerda (cruzamento da Gen. Neto com a Gen. Câmara, por exemplo). Por outro lado, quando a placa for da cor Verde=63.2, o automóvel está em uma via de sentido único próximo a um cruzamento com uma rua também de sentido único. Ou seja, em cada uma dessas situações é necessário uma rotina de programação diferente para que os veículos sejam posicionados de forma correta dentro do ambiente após decidirem pela conversão à esquerda ou seguir em frente. A Figura 3 mostra com mais detalhes essas situações.

Cor	Placa
Verde = 65.0 (ou Verde - 50 = Vermelho)	
Verde = 65.1 (ou Verde - 50 = Vermelho)	
Verde = 65.2 (ou Verde - 50 = Vermelho)	
Verde = 65.3 (ou Verde - 50 = Vermelho)	
Verde = 65.4 (ou Verde - 50 = Vermelho)	
Verde = 65.5 (ou Verde - 50 = Vermelho)	
Verde = 65.6 (ou Verde - 50 = Vermelho)	
Vermelho (Entre 15.0 - 15.6)	

Figura 3. Legendas implementadas para simular as placas do trânsito.

Algumas ruas apresentam sentido duplo e, por isso, foram simuladas através do posicionamento de “duas” ruas, uma ao lado da outra. As ruas na cor *Violeta* têm sentido para a esquerda e as ruas na cor *Laranja* têm sentido para a direita. As ruas na cor *Azul* têm sentido para cima e as ruas na cor *Rosa* têm sentido para baixo. Neste modelo, o tamanho físico real das ruas, bem como o tamanho dos veículos, não são levados em consideração. Esta simulação apresenta apenas uma aproximação da realidade.

Na interface da simulação o usuário pode definir a quantidade de automóveis presentes no ambiente através do *slider* “number-of-cars”. O botão *setup* inicializa o ambiente, traçando as ruas, posicionando os semáforos e inserindo a quantidade de veículos selecionada em locais apropriados (ruas). Existem dois monitores: um mostra a média de automóveis parados durante todo o tempo decorrido da simulação (“Média”) e o outro a quantidade de automóveis parados naquele momento da simulação (“Parados”). Foi colocado um *plotter* (“Carros”) que mostra o histórico da quantidade de automóveis parados. Como opção para o usuário, foi inserido um *switch* para possibilitar o *desligamento* dos semáforos. A Figura 4 ilustra o ambiente criado.

A ideia do código implementado é descrita no Algoritmo 2.

A cada *tick* os veículos verificam a rua que estão e verificam seu sentido para realizar o devido deslocamento. Quando o automóvel encontra uma esquina, se o *patch*



Figura 4. Zona central da cidade de Rio Grande simulada no NetLogo.

```

begin
  for Cada veículo do
    Verifica sentido das ruas.
    Verifica semáforos.
    Locomover-se.
  end
  if Semáforos ligados then
    | Alterne as luzes.
  end
  Conte quantos veículos estão parados no momento.
  Desenhe no ploter.
  Conte um tick no relógio.
end

```

Algoritmo 2: Pseudo-código utilizado no modelo proposto.

for verde, ele escolhe um caminho disponível e verifica se não vai causar nenhum acidente olhando a posição para que deverá se posicionar, se existir algum automóvel lá, o *turtle* em questão pode escolher outro caminho. Caso o *patch* for vermelho, o veículo deverá permanecer parado até que a cor do *patch* seja alternada para verde. A Figura 5 mostra uma evolução da simulação.

É importante ressaltar que os agentes têm uma probabilidade um pouco maior de seguir nas ruas mais importantes da zona central. Neste caso, quando um agente está na Rua “24 de Maio” ou na Rua “Gen. Neto”, o veículo terá uma maior chance de seguir na mesma rua. Esta condição foi incluída na simulação pelo fato de ser uma situação facilmente observada em tais ruas.

4. Resultados da simulação

Nesta seção serão apresentados algumas ponderações com relação aos resultados obtidos nas simulações realizadas.

Para efeitos de comparação, foi determinado um número máximo de *ticks* (passos de simulação) onde a evolução do modelo atinge a estabilidade na média de veículos para-

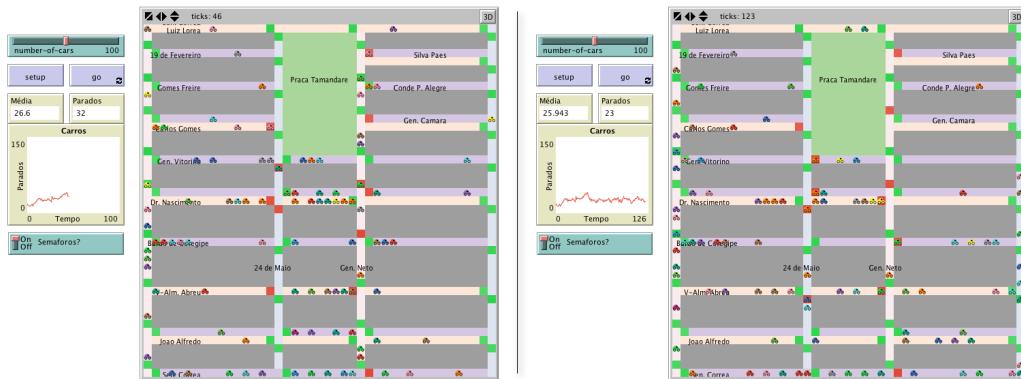


Figura 5. Acontecimentos conforme o andamento da simulação no modelo implementado.

dos, a qual foi utilizada para comparar a fluidez do trânsito em cada uma das simulações. Neste caso, optou-se por 2000 ticks. Desta forma, serão comparadas duas situações. Na primeira, os semáforos fazem parte do ambiente, como mostrado na Seção 2. Na segunda, os semáforos do ambiente estão desligados. A Tabela 1 mostra os valores obtidos com as simulações.

Tabela 1. Comparação da média de veículos parados nas simulações com e sem semáforo, para quantidades diferentes de automóveis.

Quantidade de automóveis	Media de automóveis parados	
	(com semáforos)	(sem semáforos)
35	4.31	0.74
75	14.42	5.76
120	43.57	29.89
180	87.31	67.76
200	102.77	82.77

Como é possível verificar, o modelo sem os semáforos apresenta melhor fluidez do trânsito do que o modelo com semáforos, em todos os testes realizados.

Baseado na média de veículos parados no modelo, era esperado que tais resultados fossem obtidos. Ou seja, no modelo onde existem semáforos, os automóveis são obrigados a ficarem parados por um determinado tempo enquanto que, no modelo sem a existência de semáforos, os veículos (agentes) têm a possibilidade de atravessar os cruzamentos sem nenhum tipo de prioridade. Neste caso, a organização do cruzamento é realizada de forma aleatória permitindo a passagem de um veículo por vez sem dar preferência a uma determinada rua.

5. Conclusão

Infelizmente, o sistema sem a presença de semáforos é inviável. Isso, pois, seria necessário que os motoristas se comportassem com o respeito e paciência dos agentes criados nesta simulação (situação ideal). Além disso, algumas situações não estão presentes nas simulações como, por exemplo, a presença de pedestres. Apesar de aumentar a fluidez do trânsito, uma cidade sem semáforos não permitiria uma boa relação entre veículos

e pedestres podendo ocasionar acidentes e maiores transtornos à gestão do trânsito. Por isso, para que os pedestres tenham sua vez com relação aos automóveis, é necessário o uso de semáforos nas cidades.

Como sugestão de trabalho futuro, pode-se melhorar o modelo adicionando pedestres com possibilidade de circulação nos cruzamentos com e sem semáforos. Dessa forma, a simulação ficará mais próxima da realidade, proporcionando uma melhor avaliação das ruas onde ocorrem engarrafamentos e acidentes.

Estudos indicam que no futuro os automóveis serão automáticos a ponto de não precisar de motoristas, funcionando como agentes autônomos que interagiriam entre si para fazer o trânsito funcionar da melhor maneira possível. Isso se aproximaria muito da modelagem realizada e vice-versa.

Referências

- Bazzan, A. L. C. (2005). A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10:131–164.
- Marchioro, E. (2012). Plano de mobilidade urbana. Secretaria Municipal da Segurança, dos Transportes e do Trânsito. Prefeitura Municipal do Rio Grande. Disponível em Abril de 2012. <http://www.riogrande.rs.gov.br/pagina/index.php/conteudos-gerais/detalhes+16859,,plano-de-mobilidade-urbana.html>.
- NetLogo (2011). Netlogo home page. Disponível em Dezembro de 2011. <http://ccl.northwestern.edu/netlogo/>.
- WOOLDRIDGE, M. and JENNINGS, N. R. (1995). *Intelligent Agents: Theory and Practice*, volume 10 of *The Knowledge Engineering Review*. 2 edition. p. 115–152.

Parte V

Artigos Resumidos

Sistemas Multiagente Plenamente Distribuídos

Tiago Mazzutti¹, Ricardo A. Silveira²

¹Departamento de Informática - Instituto Federal
de Educação Ciência e Tecnologia Catarinense (IFC) - Câmpus Concórdia
Caixa Postal 56 - 89700-000 - Concórdia - SC - Brasil

²Departamento de Informática e Estatística - Universidade Federal
de Santa Catarina (UFSC)
Caixa Postal 476 - 88.040-900 - Florianópolis - SC - Brasil

tiago.mazzutti@ifc-concordia.edu.br, silveira@inf.ufsc.br

Resumo. A fim de termos sistemas multiagente plenamente distribuídos na Web é necessário que os agentes possam ser executados na máquina do cliente satisfazendo os requisitos da aplicação e ignorando os problemas habituais causados por questões como a segurança e execução concorrente. As soluções atuais disponíveis fazem uso de applets, que dependem da extensão, instalação ou configuração de plug-ins pelo usuário, ou através de aplicações completamente dependentes de servidores, que têm apenas uma fina camada no lado do cliente. Neste trabalho é apresentado um novo modelo de implementação de um ambiente de execução para agentes plenamente distribuídos. A solução proposta permite a execução de uma plataforma multiagente através do navegador do usuário e faz uso do Google Web Toolkit para para compilar Java para código JS que é compatível com a maioria dos navegadores web modernos, deixando o desenvolvedor em um nível mais abstrato de desenvolvimento, programando na linguagem Java.

Abstract. In order to have fully distributed multiagent systems in the Web it is necessary that agents can be executed on the client machine satisfying the requirements of the application and bypassing the usual problems caused by issues such as safety and concurrency in currently available systems. Current solutions available use applets, which depend on the extension, installation or configuration of plug-ins by the user, or through applications completely reliant on servers, which only have a thin layer on the client side. This paper presents a new model of implementation of an execution environment for fully distributed agents. The proposed solution will allow the execution of a multiagent platform through the user's browser and makes use of Google Web Toolkit to compile Java to JS code that is compatible with the majority of modern browsers, leading the developer to a higher abstract level of development, programming in Java.

1. Modelo para o Desenvolvimento de SMAs no lado do Cliente

No desenvolvimento de aplicações web no lado do cliente existem algumas características que devem ser levadas em consideração, entre elas a portabilidade, execução sem a instalação/atualização de plug-ins e facilidade de aprendizado das tecnologias envolvidas. Esses fatores têm influência positiva em todo o processo de desenvolvimento.

Ao se planejar e desenvolver um sistema multiagente destinado a execução no lado do cliente essas boas características devem ser bem exploradas. Sendo assim, o sistema deve ser compatível com o maior número de tecnologias possíveis, proporcionando a mesma experiência, independentemente de onde e como o aplicativo esteja sendo acessado. Do ponto de vista do desenvolvedor, é extremamente importante que a portabilidade seja obtida de maneira fácil e natural, evitando-se a necessidade de retrabalho ao acrescentar suporte a uma nova plataforma.

Forçar o usuário a instalar, estender, atualizar/configurar qualquer software no seu dispositivo para poder ter acesso a um aplicativo é um fator preocupante para qualquer desenvolvedor

web. Portanto, um SMA no lado do cliente deve explorar todos os recursos disponíveis sem a necessidade de novos plug-ins ou extensões.

2. Recursos e Restrições no lado do Cliente

Procurando atender as características e requisitos apresentados anteriormente, e buscando respostas para questões sobre como e onde executar?, quais os recursos de software/hardware disponíveis? e quais as principais restrições? realizou-se um estudo no sentido de elucidar como e sob que condições um SMA no lado do cliente poderia atender as necessidades de seus usuários.

Inicialmente, verificou-se através de um estudo das tecnologias web atualmente disponíveis, a maneira mais indicada para executar um SMA web no lado do cliente é através de um motor de execução JavaScript de um navegador web. No entanto, existem duas alternativas dentro desta solução: aplicações híbridas (web/desktop) - quando o motor é incorporado ao aplicativo - e aplicações *standalone* - onde o motor utilizado é parte do navegador web do usuário.

2.1. Aplicações Web/Desktop Híbridas

Um motor de navegador web permite que sejam criados aplicativos híbridos. Essas aplicações híbridas podem usar recursos de desktop e também pode acessar o conteúdo web. Como exemplos de motores podemos citar o Webkit [WebKit 2011, Nokia 2011, Network 2011]. Alguns motores de navegador web contêm uma biblioteca com recursos implementados que facilitam sua utilização. Por exemplo, QTWebKit [Nokia 2011] possui bibliotecas que incluem recursos NPAPI [Wikipedia 2011] disponíveis em uma interface JavaScript.

Embora com os motores dos navegadores Web seja possível a integração e utilização de bibliotecas destinadas a aplicativos desktop, execução nativa e a disponibilidade de implementações de código aberto, existem alguns problemas relacionados a portabilidade e a necessidade de instalação/configuração. Também existe a necessidade de incorporar o motor na própria aplicação, levando a um tamanho final do aplicativo muito maior. Esta abordagem possui outro ponto negativo onde o desenvolvedor é forçado a usar mais de uma linguagem de programação para implementar sua aplicação, levando a uma curva de aprendizagem e probabilidade de erros maior.

2.2. Aplicações Web *Standalone*

A ideia de uma aplicação web *standalone* é que os aplicativos sejam executados diretamente no navegador web do cliente, eliminando a necessidade de qualquer instalação e/ou configuração. Neste caso, as aplicações são desenvolvidas com base nos recursos já disponíveis nos navegadores (HTML e JavaScript), sendo que o usuário sempre tem acesso a versão mais atualizada do aplicativo automaticamente. A portabilidade é mais um fator favorável de aplicações *standalone*.

Por outro lado, existem problemas relacionados com a curva de aprendizado da linguagem JavaScript e algumas limitações em relação a característica *single-threaded* desta linguagem, e também, restrições relacionadas a segurança como por exemplo o acesso ao sistema de arquivos locais. Lidar com essas questões pode exigir um maior conhecimento do desenvolvedor, embora existam algumas soluções práticas para estes problemas, como por exemplo, o uso de *frameworks* como o Google Web Toolkit para o desenvolvimento.

Pelo exposto acima podemos observar que com o uso de recursos de uma aplicação web *standalone*, é possível obter muitas vantagens tais como atualizações automáticas, portabilidade e a não necessidade de instalação e configuração por parte dos usuários, portanto, é a melhor opção. Assim, apresentamos neste trabalho um modelo para o desenvolvimento de sistemas multiagente plenamente distribuídos, que utiliza apenas recursos disponíveis no navegador web do usuário.

3. Modelo Proposto

Para apoiar o desenvolvimento de sistemas multiagentes web utilizando os recursos nativos no lado do cliente e observando as restrições anteriormente relatadas, o modelo apresentado na Figura 1 foi proposto.

No modelo, há uma distinção intencional do lado do cliente e do lado do servidor. De acordo com o estudo realizado, o paradigma de agentes pode proporcionar maior autonomia e proatividade aos aplicativos rodando e utilizando recursos nativos no lado do cliente, permitindo que os agentes possam atender as necessidades de seus usuários, e, portanto, ter um sistema multiagente plenamente distribuído e autônomo.

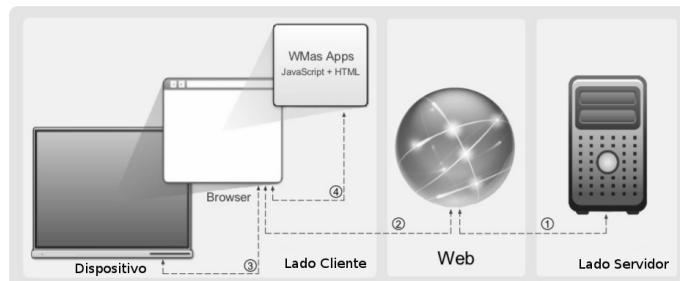


Figura 1. Modelo Proposto

No lado esquerdo da Figura 1, identificada como do lado do cliente, é possível observar que as aplicações executam em navegadores web usando apenas os recursos fornecidos por eles (JavaScript + HTML). As linhas bidirecionais tracejadas e enumeradas de 1 a 4 mostram os fluxo possível de informações.

O modelo não restringe a comunicação entre dois aplicativos que estejam sendo executados no mesmo navegador, em diferentes navegadores no mesmo cliente, ou entre diferentes navegadores em diferentes clientes. Em outras palavras, é possível que as aplicações web seguindo o modelo possam efetuar troca de dados entre duas aplicações rodando em diferentes clientes sem a necessidade de um servidor.

Outro aspecto importante é que o dispositivo representado no modelo pode ser qualquer dispositivo que permita a execução de um navegador Web, como desktops, tablets e smartphones.

Neste modelo, o servidor tem o papel de armazenar o código fonte da aplicação e prestar serviços globais, tais como autenticação, tabela de endereços, suporte a migração e banco de dados. Isso faz com que o servidor atue como uma espécie de coordenador das sub-plataformas em execução no lado dos clientes.

4. WebMAS

A solução em desenvolvimento para sistemas multiagente plenamente distribuídos apresentada nesta seção usa a ferramenta de Web Google Toolkit (GWT) [Murugesan 2007, Dewsbury 2007] para criar o front-end usando linguagem de programação Java e o GWT é responsável pela compilação cruzada para JavaScript otimizado que executa nativamente nos navegadores web mais utilizados atualmente. Como resultado desta proposta, a Figura 2 apresenta a arquitetura abstrata de um aplicativo desenvolvido usando WebMAS.

Na porção superior da imagem está representada a execução no lado do cliente do sistema multiagente. Com o uso de GWT é possível compilar um sistema multiagente desenvolvido em Java, gerando um arquivo executável (JavaScript + HTML) compatível e otimizado para cada navegador selecionado.

Do ponto de vista do desenvolvedor é virtualmente possível utilizar qualquer biblioteca Java com WebMAS. Portanto, com o WebMAS, é possível que qualquer *framework* de agentes desenvolvimento em Java, seja utilizado, levando em consideração que isso pode exigir que algumas adaptações ao *framework* em utilização sejam necessárias para integrá-lo as bibliotecas disponíveis ao desenvolvimento.

Na porção inferior da Figura 2 está representado o lado do servidor que executa parte do sistema multiagente desenvolvido com WebMAS. Além de hospedar a plataforma WebMAS

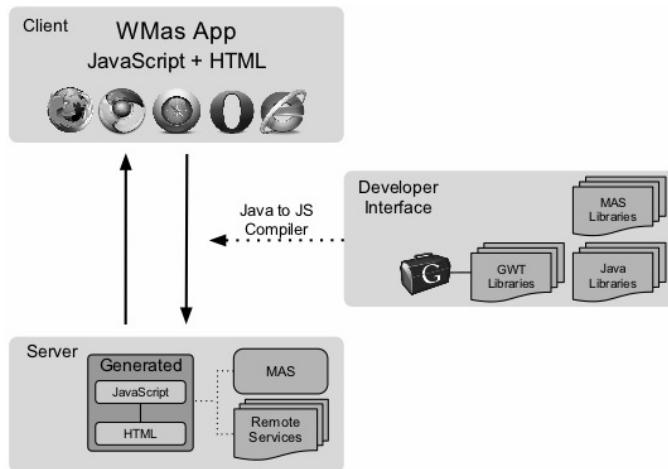


Figura 2. Framework WebMAS

global, o servidor e responsável por manter e servir os arquivos executáveis, e fornecer serviços remotos como comunicação, localização, páginas amarelas e páginas brancas.

5. Considerações Finais

Uma Implementação experimental para o desenvolvimento de sistemas multiagente plenamente distribuídos está em desenvolvimento. Os resultados obtidos são muito promissores e mostram que a abordagem pode ser uma solução para os problemas aqui apresentados.

A solução satisfaz os requisitos encontrados e tenta resolver as limitações de desenvolvimento de aplicações que executem no lado do cliente de uma forma elegante, com o uso de estruturas robustas e bem documentadas, como o uso do GWT. Alguns problemas como as limitações de JavaScript estão sendo contornadas com o uso dos novos recursos do HTML5.

Confrontado-se a solução aqui apresentada com as já existentes, que fazem uso de applets, obteu-se um grande avanço, uma vez que o WebMAS não depende da extensão, instalação e/ou configuração de plug-ins pelo usuário, e tem grande potencial para a utilização dos recursos de computação disponíveis no lado do cliente. Sendo assim, o WebMAS pode ser utilizado para o desenvolvimento de sistemas multiagente web, destinados a solução dos problemas computacionais apresentados pelo usuário.

Referências

- Dewsbury, R. (2007). Google web toolkit applications.
- Murugesan, S. (2007). Understanding web 2.0. *IT professional*, 9(4):34–41.
- Network, M. D. (2011). Gecko. <https://developer.mozilla.org/en/Gecko>. Online: acessado em Julho-2011.
- Nokia, C. (2011). Webkit in qt. <http://doc.qt.nokia.com/4.7-snapshot/qtwebkit.html#details>.
- WebKit (2011). The webkit open source project. <http://www.webkit.org/>. Online: acessado em Julho-2011.
- Wikipedia (2011). Npapi — wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=NPAPI&oldid=434583869>. Online: acessado em Julho-2011.

Uso do PopOrg na modelagem de personagens autônomos em jogo com a técnica de *interactive storytelling*

Gleifer V. Alves¹, Antonio C. da Rocha Costa², Raquel de M. Barbosa², Priscilla D. Gaertner¹

¹Coordenação de Ciência da Computação – Câmpus Ponta Grossa
Universidade Tecnológica Federal do Paraná (UTFPR)
84.016-210 – Ponta Grossa – PR

²Centro de Ciências Computacionais - C3
Programa de Pós-Graduação em Modelagem Computacional - PPGMC
Universidade Federal do Rio Grande (FURG)
96.203-900 – Rio Grande - RS

gleifer@utfpr.edu.br, {ac.rocha, rq.mbarbosa}@gmail.com, pridgaertner@hotmail.com

Abstract. In this work we describe a proposal for autonomous characters modelling in a digital game with the interactive storytelling technique. The agent modelling will be build in the PopOrg, a population and organizational agent model. The interactive storytelling technique aims to use believable agents, which could create, modify and interact through stories in a digital game, therefore improving some AI game features. The game considered here is a car simulator.

Resumo. Este trabalho descreve uma proposta para modelagem de personagens autônomos em jogos que utilizem a técnica de interactive storytelling. Tal modelagem será construída através do modelo populacional organizacional PopOrg. A técnica de interactive storytelling busca viabilizar que agentes credíveis possam gerar, alterar e interferir em histórias dentro de um jogo, o que eventualmente aprimora a técnica de IA do jogo eletrônico. O jogo considerado aqui é um simulador veicular.

1. Introdução

Em um projeto relacionado ao presente trabalho, está sendo desenvolvido um jogo eletrônico para simulação veicular. Um dos objetivos desse jogo consiste em simular diferentes comportamentos dos personagens, para assim, por exemplo, fazer com que um condutor de veículo tenha que lidar com situações inesperadas. Além disso, é desejado que os personagens possam de alguma forma interferir e interagir no enredo do jogo. Para tal existe uma técnica denominada *interactive storytelling*, que pode ser aplicada a um jogo para viabilizar a construção de novas histórias de forma dinâmica. Em [Cavazza et al. 2001], tem-se que um sistema de *interactive storytelling* é baseado em atores virtuais autônomos, os quais criam um enredo através de suas interações em tempo real.

Conforme mencionado em [Vuono 2008], um sistema de *interactive storytelling* tem dois componentes principais: **(i)** gerenciador de enredo, o qual é usado para controlar a coerência da história; **(ii)** e agentes credíveis (do inglês, *believable agents*), que caracterizam os personagens autônomos do jogo. Nota-se

que ambos componentes têm uma relação estreita com agentes inteligentes e sistemas multiagentes. Dentro da área de sistemas multiagentes existem diferentes abordagens relacionando a modelagem de organizações de agentes. Barbosa [de Miranda Barbosa 2011] descreve uma análise de alguns modelos como MOISE⁺, OPERA e PopOrg. O modelo PopOrg criado por Demazeau e Rocha Costa [Demazeau and da Rocha Costa 1996] caracteriza-se como base para um modelo formal de sistemas multiagentes (**SMA**) com organizações dinâmicas. Conforme já argumentado em [de Miranda Barbosa and da Rocha Costa 2011] e [de Miranda Barbosa 2011], o modelo PopOrg caracteriza-se por ser um modelo minimalista de organizações de sistemas multiagentes, onde o modelo representa apenas os componentes-chave de uma organização, assim permitindo que modelos mais complexos (caso necessário) sejam construídos de forma modular, conforme descrito em [da Rocha Costa and Dimuro 2009]. Portanto, o objetivo principal desta proposta é utilizar o modelo PopOrg na modelagem dos personagens autônomos de um jogo de simulação veicular, bem como na organização (social) dos personagens envolvidos nas histórias interativas. O restante do trabalho é organizado como segue. Na Seção 2 descreve-se o modelo PopOrg. A Seção 3 destaca a proposta de modelagem através do PopOrg. Na Seção 4 estão as considerações finais.

2. Modelo PopOrg

No PopOrg são analisados dois aspectos relevantes dos sistemas multiagentes: população e organização. A população de um **SMA** é o conjunto de agentes que o habitam, juntamente com o conjunto de todos os comportamentos que eles são capazes de executar, além do conjunto de todos os processos de interação que eles podem estabelecer entre si [de Miranda Barbosa and da Rocha Costa 2011]. Alguns conceitos estabelecidos em [de Miranda Barbosa and da Rocha Costa 2011] a respeito do PopOrg são citados, já que serão relacionados com aspectos da proposta descrita na Seção 3. **(i)** Agente: elemento central da população de um sistema multiagente; **(ii)** Ação: ações que podem ser realizadas pelos agentes; **(iii)** Comportamento: é uma sequência de conjuntos de ações indicando o conjunto de ações que o agente que realiza o comportamento pode realizar em cada instante; **((iv))** Processo de Troca: modela as interações entre pares de agentes representando o conjunto de ações que cada um dos participantes da interação realiza.

Outro conceito relacionado é o de *sociedade de agentes*, definido por Rocha Costa [da Rocha Costa 2011].

DEFINIÇÃO 2.1 (sociedade de agentes) *Uma sociedade de agentes é um sistema multiagentes aberto e persistente.*

Por **sistema multiagente aberto**, deve-se entender um sistema em que os agentes podem entrar e sair livremente. Por **sistema multiagente persistente**, deve-se entender um sistema cujas estruturas e funções, em cada instante, têm existência e funcionamento independente do particular conjunto de agentes que habitam a sociedade naquele instante, de modo que tais estruturas e funções são capazes de persistirem no tempo independentemente da entrada e saída de agentes do sistema [da Rocha Costa 2011].

3. Proposta de Modelagem de Personagens Autônomos através do PopOrg

Nesta Seção descreve-se uma proposta inicial para modelagem de personagens autônomos em um jogo que faça uso da técnica de *interactive storytelling*. O primeiro passo é a

definição dos tipos de agentes existentes dentro do jogo. Em [Cai et al. 2010], descreve-se que algumas abordagens usualmente estabelecem dois tipos de agentes: **diretor** e **ator**. Contudo, os próprios autores, no trabalho supracitado, propõem uma arquitetura para *interactive storytelling* baseada em agentes, onde cada agente pode tanto atuar como diretor, ou como ator. Nessa proposta optou-se pela seguinte classificação: **agente externo** (ou *gerenciador de enredo*), **diretor** e **ator**, onde os dois últimos são ditos também agentes credíveis.

Tendo definido o conjunto de agentes, nos passos que seguem serão definidos: ações e interações, o que irá auxiliar no entendimento da classificação dos agentes empregada nessa proposta. O conjunto de ações é determinado como segue: (i) **Agente externo**: controlar a coerência das histórias; (ii) **Agente diretor**: criação e alteração de histórias; (iii) **Agente ator**: interpretam os papéis envolvidos na história e realizam os processos de interação entre papéis especificados na mesma.

O conjunto de processos de interação é definido da seguinte maneira:

- **Agente externo**: esse agente pode interferir na história h_1 criada pelo agente diretor A_{d1} , desde que A_{d1} não respeite a propriedade de coerência em h_1 .
- **Agente diretor**: um agente diretor A_{d1} pode definir o conjunto (inicial) de agentes atores que vão atuar na história h_1 criada por A_{d1} .
- **Agente ator**: os agentes atores realizam os processos de interação a respeito da *coordenação* e da *negociação*.

Conforme descrito em [Riedl et al. 2003], as interações entre os agentes, especificamente a coordenação de operação e as técnicas de negociação entre agentes constituem dois aspectos fundamentais para assegurar a coerência nas histórias.

Adicionalmente, através do conceito de Sociedade de Agentes (ver Definição 2.1) é possível destacar alguns aspectos para estabelecer se os agentes credíveis utilizados nesse proposta devem fazer parte de uma Sociedade de Agentes. Considerando o uso de tal conceito seria possível descrever a seguinte situação. Dada uma história h_1 criada pelo agente diretor A_{d1} e outra história h_2 criada por A_{d2} . Tendo dois agentes atores (A_{a1} e A_{a2}), seria possível afirmar que A_{a1} atua em h_1 e A_{a2} atua em h_2 . Mas, como (dentro do contexto de Sociedade de Agentes) os agentes atores teriam liberdade para entrar e sair de uma história, o seguinte poderia ocorrer: o agente A_{a1} deixa h_1 (*i.e.*, não atua em h_1); o agente A_{a1} entra (atua) em h_2 .

Um exemplo é ilustrado para agregar as idéias descritas na presente Seção. Considerando, inicialmente os seguintes elementos de um jogo: (i) Cenário: ambiente urbano; (ii) Conjunto de agentes atores: condutores de veículos (A_{a1}, A_{a2}); transeunte (A_{a3}, A_{a4}); (iii) Conjunto de agentes diretores: A_{d1} cria a história h_1 , A_{d2} cria a história h_2 .

- **história 1 (h_1)**: agente A_{a1} tem comportamento de um condutor que avança o sinal vermelho quando A_{a1} está atrasado para o trabalho. Esse comportamento pode interferir no transeunte A_{a3} , que atravessa um cruzamento no local apropriado, *i.e.*, o local onde há semáforo ou faixa de segurança.
- **história 2 (h_2)**: agente transeunte A_{a4} tem comportamento de um transeunte que atravessa a rua em local não apropriado, *i.e.*, local onde não há semáforo ou faixa de segurança. Tal comportamento pode interferir no agente condutor A_{a2} , que eventualmente precisará fazer uma manobra inesperada.

Agora, considerando o cenário descrito acima com as definições dos agentes e suas interações, é possível refletir a respeito de como a noção de Sociedade de Agentes poderia ser aplicada nesse exemplo. No caso, o agente A_{a1} pode deixar h_1 e passar a atuar em h_2 , ao passo que A_{a2} deixa h_2 e passar a atuar em h_1 . Essa alteração provocaria modificações nas duas histórias e afetaria as interações e comportamentos dos agentes envolvidos no cenário.

4. Considerações Finais

Destaca-se que a proposta aqui descrita está em fase inicial e que nas etapas subsequentes é necessário lidar com certos aspectos, como: (i) Adaptar a modelagem para o jogo de simulação veicular; (ii) Mas, ao mesmo tempo, definir a modelagem de forma que possa ser usada para outros jogos similares, que utilizem a técnica de *interactive storytelling*. Para assim, aplicar o PopOrg na modelagem de jogos com *interactive storytelling*; (iii) Definir formalmente a propriedade de *coerência* em histórias dentro do PopOrg.

Ademais, através das seguintes referências: [de Miranda Barbosa and da Rocha Costa 2011] e [de Miranda Barbosa 2011], almeja-se especificar formalmente a modelagem feita no PopOrg através do método de especificação formal RAISE/RSL.

Referências

- Cai, Y., Shen, Z., Miao, C., and Tan, A. (2010). DIRACT: Agent-Based interactive storytelling. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 273–276. IEEE.
- Cavazza, M., Charles, F., and Mead, S. J. (2001). Ai-based animation for interactive storytelling. In *Proceedings of IEEE Computer Animation 2001*.
- da Rocha Costa, A. C. (2011). O nível cultural das sociedades de agentes. In *WESSAC 2011*, Curitiba, PR.
- da Rocha Costa, A. C. and Dimuro, G. P. (2009). A minimal dynamical mas organization model. In Dignum, V., editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 419–445. Hershey. IGI Global.
- de Miranda Barbosa, R. (2011). *Especificação Formal de Organizações de Sistemas Multagiêntes*. PhD thesis, Programa de Pós-Graduação em Computação - UFRGS.
- de Miranda Barbosa, R. and da Rocha Costa, A. C. (2011). Uso de rsl para a especificação formal de organizações de smas baseadas em poporg. In *WEIT 2011 - Workshop Escola de Informática Teórica*.
- Demazeau, Y. and da Rocha Costa, A. C. (1996). Populations and organizations in open multi-agent systems. In *Proceedings of the 1st. National Symposium on Parallel and Distributed AI (PDAI'96)*, Hyderabad, India.
- Riedl, M., Saretto, C., and Young, R. M. (2003). Managing interaction between users and agents in a multi-agent storytelling environment. In *AAMAS '03 , July 14-18, 2003, Melbourne, Australia*.
- Vuono, V. (2008). Interactive storytelling via intelligent agents. In Damian, M. and Way, T., editors, *CSRS 2008 - 2nd Villanova University Undergraduate Computer Science Research Symposium*.

Inserindo o Deslocamento de Multidões no Simulador ITSUMO

Roger Ferreira da Cruz, Paulo Roberto Ferreira Jr.

Centro de Desenvolvimento Tecnológico
Universidade Federal de Pelotas - UFPel

{rfdcruz, paulo.ferreira}@inf.ufpel.edu.br

Abstract. This paper proposes an initial study for the creation of a medium-scale model to include the movement of crowds on the urban vehicular traffic simulator called ITSUMO. This model is the first step of the developing of a simulator for emergency situations. This simulator is an extension of the simulator ITSUMO. The theoretical basis of the proposed model comes from studies on evacuation and movement of crowds in restricted environments and empirical analysis of the displacement of people in disaster situations. The proposed model will compose a new agent in the simulator ITSUMO, now called Crowd. This agent will represent crowds and will be developed based on a hybrid model based on Brownian agents and the Theory of Social Forces. As a result of the inclusion of this agent will allow the interaction of people and vehicles in emergency scenarios contained in the simulation environment of ITSUMO.

Resumo. O presente artigo propõe um estudo inicial para a criação de modelo de mesoescala para a inclusão do deslocamento de multidões no simulador de tráfego veicular urbano denominado ITSUMO. Este modelo é a etapa inicial do desenvolvimento de um simulador de situações de emergência como uma extensão do simulador recém mencionado. A base teórica do modelo proposto provem de estudos sobre evacuação e deslocamento de multidões em ambientes restritos e sobre análises empíricas do deslocamento de pessoas em situações de catástrofes. Pretende-se, então, incluir um novo agente no simulador ITSUMO, atualmente denominado Crowd, que implementa o modelo proposto. Esse agente representará multidões e será desenvolvido com base em um modelo híbrido baseado em agentes Brownianos e a teoria das Forças Sociais. Como resultado a inclusão do referido agente permitirá a interação de pessoas e veículos em cenários de emergência contidos no ambiente de simulação ITSUMO.

1. Introdução

Com o crescimento da população aglomerada nas cidades, eleva-se o risco de que incidentes causados pelo homem ou por causas naturais ocorram em larga escala. Dado o potencial de assumir grandes proporções e causar sérios problemas, fica explícita a necessidade de gerenciar esses incidentes de forma efetiva.

A resposta coordenada a incidentes de larga escala, envolvendo as diferentes forças (policiais, bombeiros, ambulâncias, etc.), sob diferentes esferas da administração

(municipal, estadual, etc.), representa um problema bastante complexo. Um dos maiores desafios nesse caso é a falta de oportunidade de treinar equipes, assim como tomadores de decisão tão heterogêneos, para lidar com estas situações de emergência.

Uma das formas de auxiliar no gerenciamento de situações de emergência é a denominada Simulação Social, utilizada no contexto para testar o comportamento organizacional de determinadas forças sociais em diversas situações.

O ITSUMO [Andriotti 2003] é um simulador social baseado em multiagentes que possibilita a inserção de módulos configuráveis, escritos em C++, com o objetivo de permitir o uso de diferentes tipos de agentes para diversas situações que envolvem o trânsito nas cidades. O modelo proposto neste artigo visa extender o ITSUMO para incluir o deslocamento de multidões. Esta extensão se dará através da implementação de um agente específico que se comportará como uma multidão, deslocando-se pelas vias e interagindo com os veículos.

Tal modelo adapta e estende o modelo de simulação de deslocamento de pedestres híbrido baseado em agentes Brownianos e no modelo contínuo de Forças Sociais [SABOIA 2010] para o modelo de Nagel-Schreckenberg, este último usado para o deslocamento de veículos.

O presente artigo está organizado da seguinte forma: Primeiramente será apresentado os trabalhos relacionados, posteriormente o modelo proposto e por fim serão apresentadas as conclusões preliminares.

2. Trabalhos Relacionados

O ITSUMO [Andriotti 2003], é uma ferramenta de simulação discreta baseada no modelo Nagel-Schreckenberg, que tem por objetivo fundamental realizar simulações de tráfego veicular. A menor escala de representação de objetos que se deslocam pelas vias no ITSUMO é um veículo. Portanto, para adequar o simulador ao modelo pretendido pelo presente artigo é necessário estudar uma maneira de representar multidões na mesma escala de tamanho dos veículos.

Em [SABOIA 2010] é apresentado um modelo de simulação de deslocamento de pedestres híbrido baseado no framework de agentes Brownianos de Schweitzer, no modelo contínuo de Forças Sociais e mais algumas modificações inspiradas no modelo discreto Lattice Gas. Este modelo foi implementado em um simulador de agentes autônomos que simula multidões de pedestres. Assim, são apresentadas técnicas que modelam comportamentos de pessoas com objetivos fixos e comuns em ambientes fechados. Esta base teórica será utilizada na formulação do novo agente multidão que, na escala de um veículo, deve comportar-se como um grupo de pessoas.

Tendo como objetivo descrever o comportamento de multidões em um estado de proliferação proveniente de eventos de pânico, um estudo preparado pelo Federal Highway Administration - U.S. Department of Transportation [BOLTON 2007] definiu as políticas para evacuação de pedestres baseado nas observações feitas nos eventos de 11 de setembro em New York Washington D.C.. Basicamente o estudo americano observou a ausência de uma política eficiente que orientasse a melhor maneira de prever evacuação de pedestres. O trabalho citado buscou definir um conjunto de regras para evacuação de pessoas em deslocamento por vias urbanas entre veículos automotores. Tais políticas serão associadas ao modelo, uma vez que nas situações que se pretende submeter ao ITSUMO os deslocamentos serão, muitas vezes, influenciados pelo pânico.

Como grande parte do comportamento de massas populacionais deve-se a fatores de cultura local buscou-se seguir como padrão a ser representado no modelo a quantidade de indivíduos em multidões de [Riso 2006], livro guia preparado para o Centro de Pesquisa de Desastres do município do Rio de Janeiro, onde é apresentado um método para estimativa da quantidade de pessoas em diversos tipos de aglomerações, bem como os padrões de velocidade e deslocamento.

Com a combinação dos estudos mencionados nesta seção será possível implementar e fundamentar as diretrizes do modelo que está sendo proposto.

3. O Modelo Proposto e a Extensão do ITSUMO

No estudo de [Saboia 2010], foi definido um modelo de forças sociais modificado, baseado no modelo Browniano em que uma partícula move-se devido aos choques com as moléculas que as cercam (agente reativo). Cada pedestre é visto como um agente autônomo e o ambiente por sua vez é definido como uma área poligonal que circunscreve um espaço contínuo. O cenário simulado no referido estudo constitui-se de espaços fechados em que cada agente, representando uma pessoa, busca escapar por uma única saída. O modelo discutido acima não pode ser aplicado diretamente no ITSUMO por diversas razões. A principal delas diz respeito ao tamanho da célula mínima do autômato do simulador.

O agente Car é definido como um artefato composto por cinco células de um metro. Como o dimensionamento celular não pode ser alterado, pois essa alteração implicaria em uma mudança também no modo como é representado e simulado o agente Carro, estabeleceu-se que o espaço mínimo para representação de pessoas nas vias será o mesmo do carro, desta forma a menor porção para representação será de 5 m².

Para estender o ITSUMO com o objetivo de permitir o deslocamento de multidões será feita então a inclusão de um novo agente, denominado Crowd. O novo agente herdará as características básicas do agente Car e conterá um grupo de pessoas que terá interação com outros grupos de pessoas e, por consequência, com os veículos.

Isso significa dizer que o conjunto dos micro mundos compostos por duas multidões trocarão mensagens, e essas mensagens possibilitaram que pessoas fundam-se em uma só multidão compacta ou se redistribuam pelo cenário de forma uniforme, essas decisões estarão diretamente relacionadas com o tipo de multidão assim como a situação de emergência, seguindo também as regras de densidade observadas por [Riso 2006].

O presente trabalho terá como ponto inicial a estimativa do quantitativo de multidões elaborado por [Rizzo 2007], que em seu livro, determinou por meio de observação, os valores para tabelar a distribuição de pessoas por tipo de aglomeração. Os valores variam de uma pessoa por metro quadrado até no máximo seis pessoas. A velocidade de uma multidão varia de 4 Km/h até 9 Km/h. Os valores dependerão do tipo de aglomeração, situação de emergência assim como estado da via.

Para tornar o comportamento das multidões mais real e flexível, tomou-se a diretriz observada nos estudos de [Bolton 2007], que determina que multidões motivadas por situações de pânico, possuem características diferenciadas como a

tendência de compactar-se e buscar a direção de suas casas na maior velocidade possível.

Desta forma, adaptando o modelo utilizado com um agente representando um indivíduo para um modelo de um agente representando um grupo de indivíduos, que chamamos de multidão, é possível implementar o deslocamento de multidões no ITSUMO.

Os estudos acerca do comportamento de multidões em vias e do comportamento de multidões em pânico serão considerados e complementarão os detalhes que serão precisos para a definição do deslocamento das multidões para a futura extensão do ITSUMO que irá permitir a ampla simulação de diversas situações de emergência.

4. Conclusões

Planejar, entender e organizar sistemas sociais reais representa uma tarefa bastante complexa. Sendo assim, novas políticas precisam ser testadas em laboratório antes de serem aderidas em cenários reais, os quais são inherentemente abertos, dinâmicos, complexos e imprevisíveis. O poder público (mas também o setor privado de negócios) tem necessidade de testar, em ambientes de simulação, do tipo laboratório, as políticas e ações antes que elas sejam efetivamente postas a prova na sociedade.

Sendo assim, o presente trabalho apresenta uma proposta na qual diferentes níveis para representação de agentes são combinadas. Os carros possuem uma escala um pouco maior do que a célula mínima simulada no agente Crowd, essas por sua vez formam simulações de agentes tipicamente microscópicos. A fusão dessas abordagens resulta em um modelo mesoscópico, modelo esse que se caracteriza por representar uma solução otimizada nesse tipo de abordagem.

Referências

- F. SCHWEITZER. Brownian Agents and Active Particles: Collective Dynamics in the Natural and Social Sciences. Springer Publishing Company, Incorporated, 2007.
- SABOIA, P. C. Simulação de multidões com agentes brownianos e modelo de forças sociais modificado. Dissertação – UNICamp. Campinas, 2010.
- ANDRIOTTI, G K. Modelagem de Motoristas e Cenários de Escolha de Rota em Simulações de Tráfego Veicular Urbano. Dissertação – PPGC UFRGS, Porto Alegre 2004.
- BOLTON, P. A. Managing Pedestrians During Evacuation of Metropolitan Areas. Material preparado para o Federal Highway Administration - U.S. Department of Transportation 2007.
- RIZZO, B. E. Estimativa do quantitativo de indivíduos em multidões. Material preparado para o Centro de Estudos e Pesquisa de Desastres do município do Rio de Janeiro 2006.

Autonomy Development for Unmanned Aerial Vehicles with *Jason* Agents

Marcelo T. Hama¹, Rafael H. Bordini¹

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

mthama@inf.ufrgs.br, r.bordini@inf.ufrgs.br

Abstract. Aerospace technology takes a very important role in several subjects and, in this context, unmanned aerial vehicles (UAV) are evolving into a noticeable theme where the application of agent designs, for autonomy development, has become a much used approach over the last decade. This paper presents an agent-oriented model to develop autonomy and to program UAVs with the use of Jason agents.

Keywords: Intelligent Agents, Unmanned Aerial Vehicle, Autonomy Development, Framework

1. Introduction

In many situations, the development of autonomy addressing particular types of systems is not an easy task due to the *abstract gap* between low-level instructions (more close to hardware level) and high-level operations (more close to application level). In these types of situations, we cannot just assume a direct implementation in view of the fact that it can make the overall system inflexible and hard to extend. In this scenario, a usual plan is to apply or construct interfaces that serve as technologic bridges or use some kind of middleware to mediate the communication between the systems.

In the context of this work, we face an *abstract gap* in an intelligent behaviour development for unmanned aerial vehicles that use an open API firmware which can send and receive data through a universal asynchronous receiver/transmitter (UART) following a particular protocol, and being controlled by a professional through radio transmitters. Generally, to save memory and processing, UAVs firmware is programmed with the use of languages that are close to hardware level (e.g., **C** and **Assembly**), but as drawback it makes the autonomy development technically expensive: autonomy involves high-level operations like intelligent team cooperation and inter-agent communication management; Input/Output management, bit-conversions, and system interruptions are not issues that this type of development should be concerned with.

2. Theoretical Basis

The application of intelligent agents [Wooldridge 2009] in the context of UAV is not in itself a novelty and there are several research projects in this subject, as we can see in [Jakob et al. 2010, Scerri et al. 2008, Sislak et al. 2008], but as far as we have surveyed, our research is the first one that proposes an open-source framework with use of a BDI agent-oriented programming language¹ in the scope of unmanned aerial vehicles. The

¹JACK Intelligent Agents has been used before, but it is a commercial and Java-oriented tool.

agent design can be seen as an interesting way to implement autonomy for UAVs due to the conceptual modeling similarities. The main approach is the application of multiple *Jason* [Bordini and Hubner 2007, Bordini et al. 2007] agents as auto-pilots for a UAV, in a conception that is reminiscent of human pilots inside a manned aircrafts. *Jason* is an *AgentSpeak*² interpretator and runtime IDE where it is possible to develop and simulate Belief-Desire-Intention [Bratman 1987] agents and multi-agent systems.

3. The Model

In an attempt to reduce the *abstraction gap* in software development for UAVs, we proposed an agent-oriented framework using *AgentSpeak* to control a UAV. The proposed model uses a technology bridge between the agent layer and UAV firmware layer, with an action protocol that defines a set of agent capabilities. The protocol actions are treated as capabilities by the agents within the UAV and, as a result, the UAV can implement partial or full autonomy using a BDI approach.

3.1. Abstraction Analogy

The model is based on an analogy between a UAV controlled by agents and common manned aircraft controlled by humans. In this analogy, an agent has capabilities that allow it to communicate with other agents or with remote services/systems while the manned aircraft pilots and tripulations can speak with each other or send/receive messages to remote personnel. The *Jason* agents have *.ASL Files* (a kind of source file) that describe its behaviour and plans, while the manned tripulation has its human intelligence and follows a flight plan. The manned aerial vehicle commands (i.e., accelerate, initiate take-off, activate landing gear, turn on ailerons, change flap angles, and so on) are modeled as a set of capabilities in an action protocol. The final concept is to see the manned aerial vehicle as the UAV with *Jason* virtual environment as the pilot cabin. Table 1 shows the modelling analogies.

Unmanned Aerial Vehicle		Manned Aerial Vehicle
Agents	models	Manned Aircraft Tripulation
AgentSpeak .ASL File	models	Tripulation Intelligence and Flight Plans
Action Protocol	models	Manned Aircraft Commands
Virtual Environment	models	Pilot Cabin
UAV	models	Manned Aircraft

Table 1. Modelling analogies.

3.2. Architecture Conceptual Model

The framework runs on a light operating system within an upgraded UAV prototype. The UAV can send/receive data bytes through the UART which make calls directly to the UAV firmware API and its low-level implementation. These data bytes are converted into readable streams by an **Input/Output Manager**, and processed by a **Converter** which uses a **Firmware Model** to construct valid values of arguments from incoming parameters, defining the framework **Engine**. These values are sent/received by the **Action Protocol**

²An extended version of *AgentSpeak(L)* [Rao 1996].

instance, and called by the implemented **Agent Architecture**. The **Agents** run within a **Jason MAS** deliberative reasoner, performing the interpretation of **.ASL Files** that describe the intelligent behaviours of each of these agents. The conceptual model is shown in Figure 1.

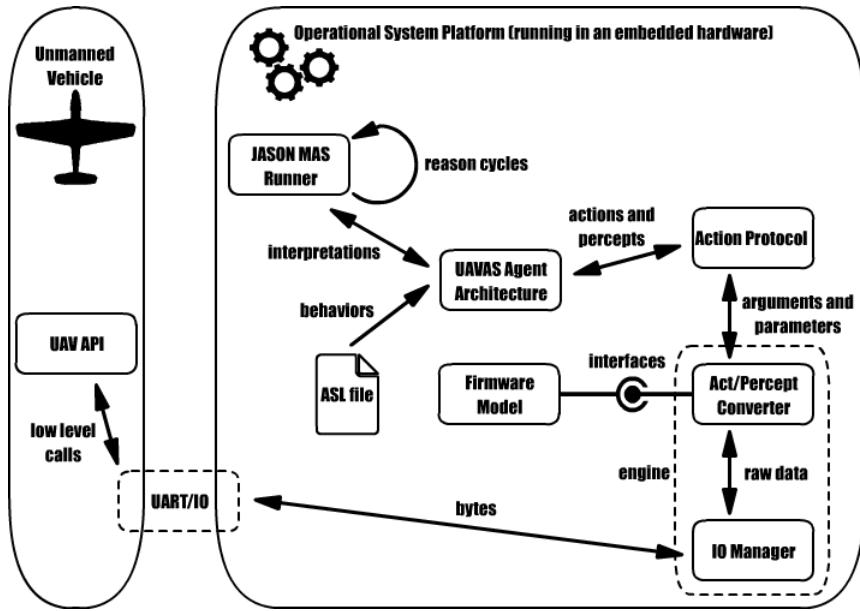


Figure 1. Conceptual model [Hama 2012].

4. Preliminary Results

As preliminary results, we are able to simulate the model infrastructure using Flight Model Simulator³ (FMS). FMS is an RC-Sim used to receive signals from radio transmitters and simulate them in a virtual UAV through a 3D graphic world. Currently, by encoding the generated streams in the same sequence created in transmitters we can simulate a set of operations⁴. In Figure 2, we show a take-off simulation screenshot using FMS. The image shows a quadrotor rising vertically.



Figure 2. Take-off simulation screenshot, using FMS [Hama et al. 2011].

³Project and download page at <http://n.ethz.ch/~mmoeller/fms/>.

⁴We could not simulate percepts because FMS does not provide an open API

5. Current and Future Works

In previous work, [Hama et al. 2011, Hama 2012], a technology bridge has been defined to address the *abstraction gap* in autonomy development for unmanned aerial vehicles. For now, we are seeking to implement a specialised simulator that employs multiple virtual UAV instantiations, inter-UAV and intra-UAV communication, and an API to perceive environment state through query capabilities. Another research direction is towards solving hardware issues that have so far prevented us from implementing the model in real UAV prototypes.

References

- Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley and Sons Ltd, Nova Jersey, USA.
- Bordini, R. H. and Hubner, J. F. (2007). *A Java-based interpreter for an extended version of AgentSpeak*. University of Durham, Universidade Regional de Blumenau.
- Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press, Harvard University.
- Hama, M. T. (2012). Uavas abstraction model - jason agents as pilots and tripulation for unmanned aerial vehicles (to appear). In *Simpósio Brasileiro de Inteligência Artificial*, Curitiba, PR, Brazil.
- Hama, M. T., Allgayer, R. S., Pereira, C. E., and Bordini, R. H. (2011). Uavas: Agentspeak agents for unmanned aerial vehicles. In *II Workshop on Autonomous Software Systems*, page 7, São Paulo, SP, Brazil. Autosoft.
- Jakob, M., Semisch, E., Pavlicek, D., and Pechoucek, M. (2010). Occlusion-aware multi-uav surveillance of multiple uraban areas. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1407 – 1408, Toronto, Canada. ACM Press.
- Rao, A. S. (1996). Agentspeak(l): Bdi agents speak out in a logical computable language. In *Proc. 7th European workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42 – 55, Secaucus, NJ, USA. Springer-Verlag.
- Scerri, P., Gonten, T. V., Fudge, G., Owens, S., and Sycara, K. (2008). Transitioning multiagent technology to uav applications. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 89 – 96, Carnegie Mellon University. International Foundation for Autonomous Agents and Multiagent Systems.
- Sislak, D., Pechoucek, M., Volf, P., Pavlicek, D., Samek, J., Marik, V., and Losiewicz, P. (2008). Agentfly: Towards multi-agent technology in free flight air traffic control. *Defense Industry Applications of Autonomous Agents and Multi-Agent Systems*, pages 73 – 97.
- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley and Sons Ltd, United Kingdom.

SISTEMA MULTIAGENTES PARA INDEXAÇÃO E RECUPERAÇÃO DE OBJETOS DE APRENDIZAGEM

Ronaldo Lima Rocha Campos¹, Ricardo Azambuja Silveira¹

¹Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

{rcampos,silveira}@inf.ufrgs.br

Abstract. *With increased production and dissemination of digital resources, a range of repositories have emerged to meet the demands required for archiving, dissemination and access. So have a scenario where the digital resource, there is stored and access is available. But his recovery is compromised by factors such as different types of description, quantity of resources or inefficiency in information retrieval techniques. Within this scope it is proposed, based on the techniques and adaptability of multiagent systems, a model search and retrieval of information.*

Resumo. *Com crescimento da produção e disseminação dos recursos digitais, uma gama de repositórios surgiram para atender demandas necessárias para seu arquivamento, divulgação e acesso. Assim tem-se um cenário onde o recurso digital, existe, é armazenado e está disponível ao acesso. Porém sua recuperação está comprometida por alguns fatores, tais como, diferentes tipos de descrição, grande quantidade de recursos ou ineficiência nas técnicas de recuperação da informação. Nesse escopo propõe-se, baseado nas técnicas e adaptabilidade de sistemas multiagentes, um modelo de busca e recuperação de informação.*

1. Introdução

Os esforços e investimentos provenientes de educadores e instituições de ensino, para produzir materiais didáticos de qualidade, são consideráveis. Nos moldes atuais de educação a distância, a maior parte do conteúdo e do conhecimento devem estar presentes na forma de materiais didáticos digitais (DOWNES, 2001). E o termo utilizado para designar estes materiais, conteúdos educacionais compartilháveis e reutilizáveis, é objetos de aprendizagem (OA). Diante disto, o desenvolvimento de mecanismos para facilitar a reutilização de materiais didáticos digitais tem atraído o interesse de grupos de pesquisa, organizações e instituições educacionais em todo o mundo.

A recuperação de objetos de aprendizagem é heterogênea, devido a diferentes padrões e especificações para a produção e armazenamento, e a grande distribuição de repositórios através da web. Além disso, é clara a falta de ferramentas eficientes de busca especializadas em OA, o que não permite reutilizar o leque de objetos de aprendizagem produzidos.

As ferramentas existentes utilizadas para recuperar informações sobre objetos de aprendizagem são geralmente baseadas em uma busca sintática. Esse tipo de pesquisa não é uma forma eficiente para se recuperar objetos de aprendizagem, como costuma ser

na recuperação de documentos comuns da web. Nos repositórios, os OAs estão catalogados com uma série de informações, tais como, o tipo de recurso digital, área de conhecimento, entre outros. Diante disso a proposta deste trabalho é o desenvolvimento de um modelo de busca inteligente com base em sistemas multiagentes (SMA) que facilite a recuperação de objetos de aprendizagem disponíveis em um cenário de repositórios heterogêneos.

2. A Recuperação de Objetos de Aprendizagem

Objetos de aprendizagem são recursos educacionais que podem ser usados no processo de aprendizagem suportada pela tecnologia. E que possuem as seguintes características: acessibilidade, interoperabilidade, adaptabilidade, reusabilidade, durabilidade, recuperação e avaliação (MCGREAL, 2004). O objeto de aprendizagem é um recurso educacional descrito formalmente por seus metadados, armazenado em um repositório, e que pode ser combinado com outros objetos de aprendizagem para criar objetos maiores, como aulas e cursos (NASH, 2005). Para assegurar as características acima citadas os objetos de aprendizagem são armazenados em repositórios digitais de conteúdo.

Repositórios digitais de conteúdos são softwares desenvolvidos com a finalidade de armazenar e organizar recursos digitais, dessa forma provêem mecanismos de busca e recuperação dos conteúdos (DOWNES, 2001; NASH, 2005). Os repositórios dispõem de interfaces para submissão, ou catalogação, de conteúdo, fazendo uso de um ou mais padrões de metadados, interfaces de disseminação e coleta, de protocolos de comunicação, e interfaces de recuperação e busca.

Este cenário justifica uma proposta de pesquisa com base em sistemas multiagentes (SMA) para elaborar um modelo de busca inteligente que facilite a recuperação de objetos de aprendizagem disponíveis em um cenário de repositórios heterogêneos. Os sistemas multiagentes são caracterizados pela capacidade de modelar agentes inteligentes capazes de se adaptar ao ambiente, para atuar de forma autônoma, de cooperar e se comunicar uns com os outros para atingir um objetivo comum (WOOLDRIDGE, 2002).

3. O Modelo Proposto

O modelo de sistema multiagente proposto foi desenvolvido para ser capaz de indexar, classificar e recuperar objetos de aprendizagem em diferentes repositórios. O framework desenvolvido em Vian e Silveira (2010) foi revisto e ampliado, dando origem a um modelo, para garantir, além de uma boa cobertura e recuperação por área de conhecimento, priorizar os resultados relevantes. Conforme descrito em Gil, De La Prieta e Rodríguez (2011) as abordagens colaborativas têm trazido bons resultados na indexação de documentos digitais, e trabalhos como MCCALLA (2004), Recker, Walker e Lawless(2003) e Manouselis (2007) justificam tal abordagem.

Gil, De La Prieta e Rodríguez (2011) defendem o uso de duas variáveis para classificar (ranquear) os resultados de busca. A primeira faz uma estimativa de qualidade do objeto, usando informações de usuários e estatísticas de acesso. Já a segunda leva em conta o perfil de quem faz a busca, para auxiliar o sistema a inferir o melhor conteúdo. Essas idéias foram incorporadas ao modelo, que após revisto, se comporta conforme a Figura 1.

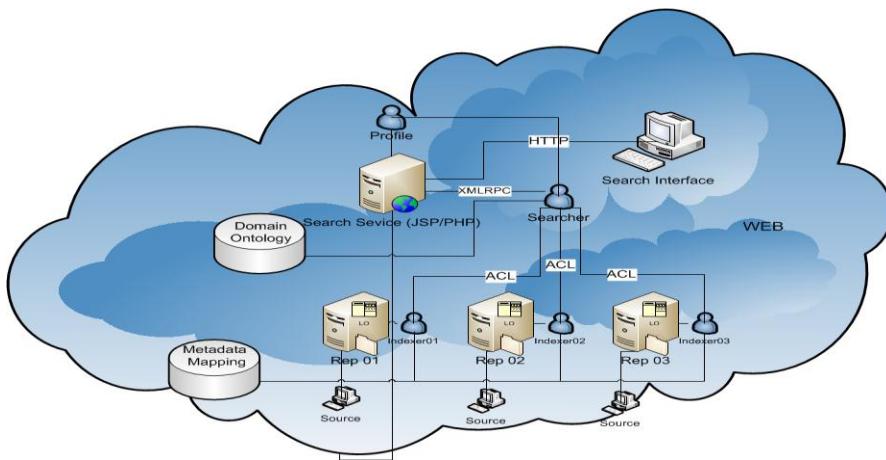


Figura 1: Arquitetura do sistema multiagente
Fonte: Adaptada de Vian, 2010

De acordo com a Figura 1, os principais componentes são: o conjunto de repositórios de objetos de aprendizagem, o sistema multiagente e o serviço web de busca. Os repositórios podem ser totalmente diferentes, tanto em seu conteúdo quanto na tecnologia em que foi desenvolvido, e para cada tipo de repositório, tem-se um agente indexador. Esse agente faz uso de um banco de dados especializado em correlacionar diferentes elementos de diferentes padrões de metadados, levando em conta as relações entre eles, para garantir ao máximo sua semântica.

Em outra parte do processo, o serviço web é responsável pela interface com o usuário, podendo ser desde um sistema de busca que indexa uma variedade de repositórios, até um serviço de busca inserido em outra aplicação, como por exemplo, um ambiente virtual de ensino e aprendizagem (AVEA). Um agente de perfil é responsável por coletar dados, ou até mesmo construir um perfil, sobre o usuário que irá solicitar uma pesquisa, com informações de outros sistemas como um AVEA, ou utilizando ferramentas de desambiguação de termos e coleta de informações como desenvolvido em D'Agostine (2009). Fazendo a ligação entre essas duas pontas, tem-se um agente buscador ligado a uma base de conhecimento, um sistema de ontologias de domínio. Esse agente recebe uma solicitação com informações do usuário, termos para busca, e utiliza essa base de conhecimento para expansão dos termos, desambiguação e para realizar o cálculo de relevância dos resultados obtidos. Para isso o agente buscador comunica-se com os agentes indexadores, podendo ele escolher os agentes que indexem bases de dados relevantes à área do conhecimento pesquisado, e solicita os objetos encontrados.

4. Considerações Finais

No atual estágio do desenvolvimento o sistema já se encontra funcional, tendo os seguintes agentes implementados: Searcher (coordenador), ldap e oai-pmh (agentes protocolos). O Searcher também está integrado ao framework JENA (JENA, 2010) sendo capaz de expandir e estabelecer relações e sinônimos entre os termos a ele submetidos.

A interface homem-máquina dá-se através de uma aplicação web que se comunica com o SMA via protocolo XML. Já é possível realizar buscas nos repositórios CESTA

(CESTA, 2010), LUME (LUME, 2010) e UNASUS-UFSC (UNASUS-UFSC, 2011). Por ora somente uma ontologia de domínio foi criada (segurança da informação).

Ao se utilizar o modelo de classificação dos objetos baseado em informações do usuário (agente perfil), avaliações colaborativas e estatísticas dos objetos, mesmo em fase inicial, apresentam resultados melhores dispostos, isso segundo validação realizada com usuários da área da informática.

Referências

- CESTA. Collection of Entities Support the use of Technology in Learning. Center for Interdisciplinary Studies in New Technologies in Education (CINTED), 2010. Disponível em <http://www.cinted.ufrgs.br/CESTA/> Acessado em 2 Abril 2010.
- D'AGOSTINE (10 de novembro de 2011), D'AGOSTINE, Caio S. Captura e Gerência de Informações de Contexto Semântico para Recuperação de Informação. 2009. 114 f. Dissertação (Mestrado em Ciências da Computação) - Programa de Pós-graduação em Ciências da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2009.
- Downes, S (2001) Learning Objects: Resources for distance education worldwide. [online] The International Review of Research in Open and Distance Learning, v. 2, n. 1, DSpace (2010). Available at <http://www.dspace.org/> [Accessed: 10 April 2010].
- Gil A.B., De La Prieta F. and Rodríguez S. (2011) Automatic Learning Object Extraction and Classification in Heterogeneous Environments. In:Highlights in Practical Applications of Agents and Multiagent Systems 9th International Conference on Practical Applications of Agents and Multiagent Systems, 2011, Volume 89/2011, 109-116.
- Jena - A Semantic Web Framework for Java, 2010. Disponível em <<http://jena.sourceforge.net/>> Acessado em 3 out 2010.
- LUME (2010). Lume - Repository of the Federal University of Rio Grande do Sul. Disponível em <Available at <http://www.lume.ufrgs.br>> Acessado em 10 Fev 2010
- Manouselis, N., Vuorikari, R., and Van Assche, F. (2007).Simulated Analysis of Collaborative Fil-tering for Learning Object Recommendation. SIRTEL Workshop, EC-TEL 2007.
- McCalla, G. (2004) The Ecological Approach to the Design of E-Learning Environments: Purpose-based Capture and Use of Information about Learners. Journal of Interactive Media in Education, 2004 (7) Special Issue on the Educational Semantic Web. Volume 1, p. 18.
- McGreal, R. (2004) Online Education Using Learning Objects. London: Routledge.
- Nash, SS (2005) Learning Objects, Learning Object Repositories, and Learning Theory: Preliminary Best Practices for Online Courses. Interdisciplinary Journal of Knowledge and Learning Objects, New York, 2005.
- Recker, M., Walker, A., and Lawless, K.(2003). What do you recommend? Implementation and ana-lyses of collaborative information filtering of web resources for education. Instructional Science, 31(4-5), pp. 299-316.
- UNASUS-UFSC (2011). Repositório UNASUS-UFSC. Disponível em <<http://repositorio.unasus.ufsc.br/>> Acessado em 19 jan 2012.
- Vian J., Silveira R.A. (2011) Multiagent System for Indexing and Retrieving Learning Objects. In: Highlights in Practical Applications of Agents and Multiagent Systems

Advances in Intelligent and Soft Computing, 2011, Volume 89/2011, 53-60, DOI:
10.1007/978-3-642-19917-2_7
Wooldridge, M. An Introduction to Multiagent Systems. England: John Wiley, 2002.

Modelo Hospedeiro-Parasitóide Baseado em Sistema Multiagente*

Érica Nicolao Lunardi[†], Aline Brocker do Amaral Velho, Igor Kimieciki[†],
Fabio Y. Okuyama, Celson R. Canto Silva

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Câmpus Porto Alegre, RS, Brasil

nicolao.lunardi@gmail.com, aline.brocker@gmail.com,
igorkimieciki@hotmail.com, fabio.okuyama@poa.ifrs.edu.br,
celson.silva@poa.ifrs.edu.br

Abstract. *Multiagent Systems based simulations allow the investigation of natural facts in ecology. This project aims to implement a computer simulation of the interaction between host insects and its parasitoids, which can be found in scenarios of biological pest control. With this work we aim to replicate the results of CANTO-SILVA (2003) in order to extend it with sensibility tests of simulation parameters. Thus, we believe that this work may contribute to ecology, agriculture and computing research areas.*

Resumo. *Simulação Baseada em Sistemas Multiagente possibilitam a investigação de fenômenos naturais da ecologia. Este projeto visa à implementação de uma simulação computacional da interação de insetos hospedeiros e parasitóides, encontrados em cenários de controle biológico de pragas. Neste trabalho pretendemos recriar os resultados obtidos em CANTO-SILVA (2003), para extendê-lo com a realização de testes de sensibilidade para os parâmetros da simulação. Desta forma, acreditamos poder contribuir para as áreas de pesquisa em ecologia, agricultura e ciência da computação.*

1. Introdução

Sistemas Multiagentes (SMA) são sistemas compostos de uma coleção de componentes autônomos, com objetivos particulares que se inter-relacionam de acordo com uma organização, interagindo para resolver tarefas. Os SMA formam uma área de pesquisa dentro da Inteligência Artificial Distribuída (IAD) que se preocupa com todos os aspectos relativos a computação distribuída em sistemas de inteligência artificial (Bordini et al, 2001).

Esses sistemas possibilitam incorporar à simulação características individuais dos organismos e características relacionadas à estrutura espacial. Além de permitirem avaliar quais são e como agem os mecanismos que afetam a dinâmica das populações. Este tipo de cenário é bastante propício para aplicação de simulação baseada em agentes, pois temos múltiplos indivíduos agindo autonomamente sobre o ambiente.

Segundo a terminologia dada em [PARRA et al. 2002], o controle biológico clássico é a importação e colonização de parasitóides ou predadores, visando ao controle de pragas exóticas (eventualmente nativas). De maneira geral, as liberações são

* Trabalho parcialmente financiado por FAPERGS.

† Bolsista do PIBITI/IFRS/CNPq – Brasil.

realizadas com um pequeno número de insetos, por uma ou mais vezes no local; por isso, o controle biológico é visto como uma medida de controle a longo prazo. E isso se comprova quando, na presente simulação populacional, os parasitóides, que são inseridos em menor número, se multiplicam para combater naturalmente os percevejos, que já habitam o ecossistema. Ecologicamente, esses inimigos naturais atuam no controle biológico de pragas, neste caso do percevejo-cinzento-do-fumo, sem a utilização de insumos químicos por parte dos agricultores.

Nesta primeira parte do projeto estamos reimplementando a simulação apresentada em CANTO-SILVA (2003). A simulação realizada em 2003 utilizava como base a plataforma SeSam (*Shell for Simulated Agent Systems*) [KLÜGL, 1998]. A modelagem realizada anteriormente na versão 1.0.2 do SeSam, não é mais compatível com as versões correntes do SeSam, o que exigiria adaptações do código anterior. Desta forma, como a equipe não tinha conhecimento da ferramenta, optou-se pelo NetLogo [WILENSKY, 2012] com a qual já havia contato anterior e apresentava as funcionalidades requeridas. O NetLogo apresenta uma interface amigável para a programação do SMA e apresentou uma rápida curva de aprendizado no desenvolvimento da simulação. A nova simulação está sendo implementada utilizando o estudo de campo e a simulação anterior como bases, e a partir dela terá início o processo de validação de dados.

Através da validação será possível verificar a veracidade do resultado alcançado pelas simulações, buscando evitar eventuais erros da simulação antiga e legitimar a nova. Após isso serão realizados os testes de sensibilidade dos parâmetros do modelo. No modelo foram utilizados parâmetros provenientes da literatura, os quais pretendemos testar através das simulações.

2. Modelagem da Simulação

No modelo implementado no NetLogo, a plantação de fumo é representada por uma grade composta por 270 células (intituladas *patches* na linguagem Logo). Cada *patch* representa uma planta de fumo. Além disso, foi adotado um valor representativo da temperatura média diária do sistema baseado em dados meteorológicos obtidos através da pesquisa de campo presente na tese de CANTO-SILVA (2003).

A interação hospedeiro-parasitóide é feita por dois seres: os hospedeiros e os parasitóides. Nela os parasitóides se alimentam de seus hospedeiros até alcançando a maturidade, de forma que o hospedeiro morre no final da interação. No caso específico implementado, o papel de hospedeiro é feito pelo percevejo *Spartocera dentiventris* e o de parasitóide pelo *Gryon gallardoi*.

A quantidade de seres simulados e a data da sua inserção na simulação é fiel ao que foi realizado em campo por CANTO-SILVA (2003). Primeiramente são inseridas dez fêmeas de percevejos adultos no 1º dia da simulação e mais 5 no 16º. Posteriormente, inserem-se os parasitóides, divididos em 3 grupos: 2 parasitóides no 30º dia da simulação, 7 no 49º e 15 no 66º.

2.1. Parâmetros do Hospedeiro

O ciclo de vida do hospedeiro *S. dentiventris* gira em torno de cento e quatorze dias distribuídos em três fases de vida. A primeira fase é a Fase Ovo que dura até quatorze dias, a segunda é a Fase Ninfa que dura até trinta e quatro dias e por fim a Fase Adulta, que dura até sessenta e seis dias. A longevidade do hospedeiro é afetada por taxas de mortalidade que variam de acordo com sua fase.

Em sua fase adulta, a taxa de probabilidade de morte do percevejo será dada em função da sua idade, eles também morrem (são excluídos do sistema) caso sua idade alcance um limite pré estabelecido (66 dias). Para sua fase de ovo, a probabilidade de morte é de 2% ao dia (excluindo-se a chance de parasitismo) e para os percevejos na fase ninfa, ela varia entre 6%, para percevejos colonizadores, e 9% ao dia para os demais gerações do percevejo.

A movimentação dos percevejos varia conforme o sexo: para os machos ela ocorre se a idade for múltipla de 5 e para as fêmeas se for múltipla de 7. Nos dias em que as fêmeas não se movimentam elas ovipositam, a quantidade de ovos ovipositados é calculada também em função da idade e a probabilidade sexual dos ovos é de 50% para macho ou fêmea. A distância percorrida em cada movimento obedece uma matriz de probabilidades elaborada a partir de dados obtidos em literatura. A direção dos movimentos é aleatória.

2.2. Parâmetros do Parasitóide

No mundo artificial, são introduzidos vinte e quatro parasitóides fêmeas adultas. Esses agentes, quando adultos, possuem um ciclo de vida curto, que é limitada tanto pela idade (15 dias) quanto pela quantidade de ovos parasitados por ele (68 ovos). Em cada dia de sua vida eles percorrem no máximo 8 passos, estimados de acordo com as distâncias em metros registradas em campo.

A movimentação e a oviposição dos parasitóides estão muito ligadas. Isto porque durante a movimentação as fêmeas procuram por plantas que contenham percevejos adultos, pois quanto mais percevejos adultos estiverem em uma planta, mais ovos poderão estar no mesmo. O número de ovos ovipositados por um parasitóide varia conforme a quantidade de ovos e o número de parasitóides concorrentes na planta. Quando um ovo do hospedeiro é ovipositado, ele se transforma em um parasitóide imaturo, e sua probabilidade sexual é de 80% para fêmea.

O parasitóide imaturo apresenta uma taxa de mortalidade fixa de 4% por dia. Nessa fase, seu desenvolvimento é inversamente proporcional à temperatura do ambiente. Caso sobreviva até atingir seu desenvolvimento completo, o parasitóide torna-se adulto.

3. Simulação da Interação Populacional

O estudo de populações e suas relações com o meio implica em um trabalho complexo decorrente de todas as regras biológicas intrínsecas ao sistema. Primeiramente deve-se observar o meio em que se encontram os parasitóides e seus hospedeiros nas modelagens. Por exemplo, não é possível dois parasitóides ovipositarem em um mesmo ovo de hospedeiro. Pois regras como o teste de residência, restringirão o acesso de muitos parasitóides e hospedeiros à mesma planta. Além de nos próprios procedimentos conter uma regra que impeça o ato da oviposição por um segundo parasitóide. Desta forma será simulada a oviposição do parasitóide, como se houvesse uma disputa pelo mesmo ovo entre vários parasitóides. Sendo assim, deve-se observar qualquer tipo de perturbação, ou seja, detalhes que possam existir no meio natural para aplicá-los corretamente no modelo.

Posteriormente, deve-se atentar para a dinâmica do sistema que tem como resultante uma interação total entre indivíduos destas populações. As populações são uma mistura complexa e inconstante de genótipos (GIACOMINI, 2007) e por isso da importância em se ter uma modelagem baseada em indivíduos. E através dessa

interação populacional, será possível prever, por exemplo, a dispersão dos hospedeiros na cultura de fumo e quantos parasitóides deveriam ser inseridos naquele meio a fim de controlar a dispersão do hospedeiro. Desta forma, SMA se torna uma ferramenta extremamente útil para auxiliar no planejamento de programas de controle biológico de pragas, a partir do momento que possibilita simular no mundo artificial não só a dinâmica temporal da interação, mas também a sua dispersão no espaço.

4. Considerações

Este trabalho é referente a um projeto de pesquisa ainda em estágios iniciais. Trata da replicação de simulação realizada anteriormente com outra ferramenta. Temos como objetivo dar continuidade e estender o desenvolvimento da simulação. Entre as extensões pretendidas, temos (i) realização de testes de sensibilidade com os parâmetros da simulação; (ii) uso da simulação como ferramenta pedagógica para estudo de interação de populações; (iii) adaptação da simulação para outros cenários de simulação de controle de pragas (*e.g.* soja).

Assim, ao realizarmos a implementação do sistema, validação e extensões, acreditamos que este trabalho contribua para as áreas de pesquisa de ecologia, agricultura e ciência da computação. Pois possibilitará o uso prático da simulação computacional, permitindo uma melhor compreensão do cenário de controle biológico de pragas e seus fenômenos associados.

5. Referências Bibliográficas

- Alvares, L. O.; Sichman, J. S. Introdução aos sistemas multiagentes. In: MEDEIROS, C. M. B. (Ed.). Jornada de Atualização em Informática (JAI'97). Brasília: UnB, 1997. cap. 1, p. 1–38.
- Bordini, R. H.; Vieira, R.; Moreira, A. F. Fundamentos de sistemas multiagentes. In: Ferreira, C. E. (Ed.). Jornada de Atualização em Informática (JAI'01). Fortaleza, Brasil: SBC, 2001. v. 2, cap. 1, p. 3–44.
- Canto-Silva, C. R., 2003. Parâmetros bioecológicos de (*Brethes*) (Hymenoptera: Scionidae) e modelagem da dinâmica espaço-temporal da sua interação com *Spartocera dentiventris* (Berg) (Hemiptera: Coreidae) através da simulação de múltiplos agentes. Tese de doutorado, UFRGS, PPGBAN. Porto Alegre, 210p.
- Giacomini, H. C., 2007. Sete motivações teóricas para o uso da modelagem baseada no indivíduo em ecologia. Acta Amaz. v. 37 no.3. Manaus.
- Klügl, F. e Puppe, F. The multi-agent simulation environment SeSAM. In: Proceedings of the Workshop “Simulation and Knowledge-based systems”. H. Kleine Büning (Ed.), University Paderborn. 1998.
- Neumann, R. I. Anuário brasileiro do fumo 1998. Porto Alegre. Gazeta Grupo de Comunicações, 1998, 103p.
- Parra, J.R.P.; Botelho, P.S.M.; Corrêa-Ferreira, B.S.; BENTO, J.M.S. Controle biológico: uma visão inter e multidisciplinar. In: Parra, J.R.P.; Botelho, P.S.M.; Corrêa-Ferreira, B.S.; BENTO, J.M.S. Controle biológico no Brasil: parasitóides e predadores. São Paulo: Manole, 2002. p.125-142.
- Wilensky, U., 2012. NetLogo Home Page. Disponível em: <<http://ccl.northwestern.edu/netlogo/>> Acesso em: 15/03/2012.

An Agent-Based Enrichment System for Genetic Diversity Analyses

**Giordano B. Soares-Souza¹, Guilherme P. G. Kingma², Eduardo Tarazona-Santos¹,
Maíra R. Rodrigues¹**

¹Instituto de Ciências Biológicas – Universidade Federal de Minas Gerais (UFMG)

²Instituto de Ciências Exatas e Informática – Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte – MG – Brazil

{jwojwo, guilherme.kingma}@gmail.com, {edutars, maira}@icb.ufmg.br

Abstract. *Different evolutionary forces produce genetic variants among individuals and populations. One of the most frequently studied variants is the Single Nucleotide Polymorphism (SNP). In some cases, population-specific genetic variants might be associated to diseases, resulting in a difference in the frequency of a disease among human populations. To achieve relevant genetic and biomedical knowledge, these variants need to be associated with other types of biological data, such as biochemical pathways, pharmacogenetics, and genome-wide associations. However, these data are fragmented in different databases. Therefore, it becomes necessary to develop tools to integrate distinct sources of biological data in order to enrich genetic diversity analyses. In this context, we propose an Agent-based enrichment system to integrate different sources of biologically relevant data in order to improve the genetic diversity analysis of native and Latin-American populations in particular.*

Resumo. *Diferentes fatores evolutivos produzem variantes genéticas entre indivíduos e populações. Uma das variantes mais estudadas é o Polimorfismo de Base Única (SNP). Em alguns casos, variantes específicas de determinadas populações podem estar associadas a doenças, resultando na diferença de frequência de uma doença entre populações humanas. Para alcançarem conhecimento genético e biomédico relevantes, essas variantes precisam ser associadas a outros tipos de informações biológicas, tais como vias bioquímicas, farmacogenética e estudos de associação por varredura genômica. No entanto, tais dados encontram-se fragmentados em diferentes bases de dados na Web. Com isso, torna-se necessário o desenvolvimento de ferramentas de integração de fontes distintas de dados biológicos, de forma a enriquecer as análises de diversidade genética. Nesse contexto, é proposto um sistema baseado em Agentes para enriquecimento de dados através da integração de diferentes fontes de dados biológicos relevantes, visando, em particular, melhorar as análises de diversidade genética de populações nativas e latino-americanas.*

1. Introduction

Although our genetic code is highly similar, different evolutionary forces produce genetic variants among individuals and populations. One of the most frequently studied variants is the Single Nucleotide Polymorphism (SNP). These polymorphisms are distributed across

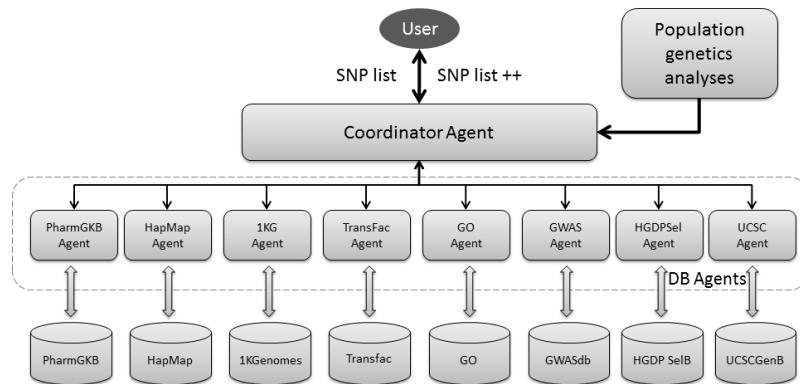


Figure 1. Multiagent enrichment system's overview.

the genome and can influence the expression of genes and, consequently, protein function. This influence is one of the causes of the different phenotypes in individuals from distinct populations, such as skin colour and hair texture. However, they sometimes lead to the loss of protein function and may cause the development of a disease. In addition to that, population-specific genetic variants might be associated to diseases or specific health-related conditions, leading to a difference in the frequency of a disease among human populations. This is the case, for example, of lactose intolerance, a condition that has higher occurrence in individuals from African and Asian populations. Therefore, the identification of variants with significant difference in frequency from one population to another is one of the main goals of genetic research and have a major biomedical interest [Myles et al. 2008].

In the field of genetic research, population genetics studies, in particular, have been used to explain human genetic diversity patterns in terms of population history (e.g., intercontinental migration history) and to understand the genetic bases of phenotypic adaptation (e.g., the genetic bases behind altitude adaptation of some Andine populations). Therefore, data generated by population genetics studies are the starting point to genetic diversity analyses of complex diseases. Some large-scale studies in this area include the HapMap project, SNP500Cancer and the on-going 1000 genomes project.

Despite the great amount of genetic diversity data available, the study of variants with biomedical interest requires their association with other types of biological data, such as biochemical pathways, pharmacogenetics, genome-wide association, and so on. Today, these heterogeneous data are fragmented in different databases. Therefore, it becomes necessary to develop tools to integrate different sources of biological data in order to enrich genetic diversity analyses and to move forward with genetic studies of particular diseases or populations. The agent-oriented paradigm has been advocated as a natural way to design and implement systems that are distributed and heterogeneous [Foster et al. 2004]. Therefore, given the heterogeneity of different types of genetic and biological data and their distribution over distinct sources over the Web, the development of a genetic analyses enrichment system points to an agent-based approach.

Moreover, there are several types of genetic diversity studies with requirements that vary from simple to more complex information integration processes. Therefore, each study will require a different enrichment process and, thus, a different combination

Database	Biological Information	Interface
PharmGKB	Diseases, Pharmacogenetics, Pathways	API to local scripts
HapMap	Populational genetic variation, Genomic annotation	Web service
1000Genomes	Populational genetic variation, Structural variation, Genomic annotation	Web service
Transfac	Transcription factors	API to local scripts, mySQL connection
Gene Ontology	Biological categories, Term enrichment	API to local scripts, mySQL connection
GWASdb	Gene expression, Association studies, Evolution, AA substitutions, Genomic mapping	Web service
HGDP	iHS, XP-EHH, Fst	Web Service
UCSC	Nucleotide conservation	mySQL connection, DAS server, Local database

Table 1. Databases with relevant biological information.

of sources of information to be integrated. This requires that the computational entities in the system have the ability to make decisions about which process to follow, and to communicate in order to coordinate the enrichment process.

In this context, we propose an Agent-based enrichment system to integrate different sources of biologically relevant data in order to improve the genetic diversity analysis of Native and Latin-American populations, in particular, and to facilitate the understanding of the effects of these diversity patterns in the development of complex diseases in Amerindian populations.

2. MAS Architecture Overview

The user interacts with the Multiagent system by selecting one of the pre-defined genetic analyses enrichments. These enrichments vary in complexity. For example, simple analyses require the access to only one or two databases. This is the case of the basic genetic variation enrichment, which consists in finding information such as the gene, chromosomal region, chromosomal coordinate, and strand of a given SNP or list of SNPs, which can all be found in the dbSNP database. Complex enrichment analyses involve several databases. This is the case of false-positive enrichment analysis for case-control association studies, which requires the retrieval of information such as population variation and molecular function and localization of a SNP, its possible effects on gene expression and regulation, and its association to diseases, phenotypes and metabolic pathways.

The basic architecture of the Multiagente system for genetic analyses enrichment is shown in Figure 1. It is composed of two types of agents: a Coordinator agent and Database (DB) agents. The Coordinator agent has knowledge about a pre-defined number of population genetics and genetic epidemiology analyses. Each analysis requires specific types of biological data. According to a selected analysis, the Coordinator agent interacts with the corresponding DB agents to request information about a variant. Once all information is returned, the Coordinator agent combines them in a report. Agent communication will be supported by specific biological ontologies, such as the SNP-Ontology and Gene Ontology.

Currently, there are eight (8) DB agents, one for each biological database identified with relevant information for the genetic diversity analyses of Native and Latin-American populations. This is necessary because the selected databases are very heterogeneous in terms of both the type of biological information and their data access interface, as shown in Table 1. Therefore, each DB agent has the knowledge of the data access mode and data format accepted and returned by the database that it encapsulates.

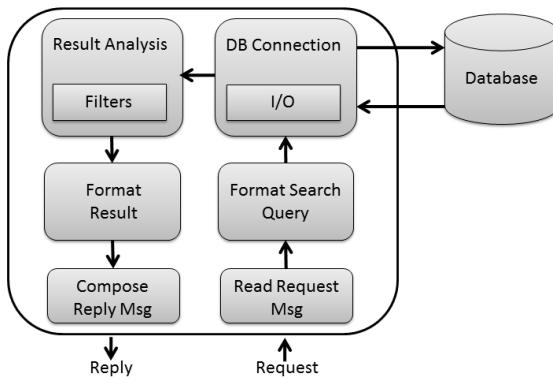


Figure 2. Internal architecture of a DB agent.

In order to search a biological database for specific information, a DB agent has the internal architecture shown in Figure 2. The Read Requested Msg module recognised the information that is being requested as compatible for search or not, depending on its knowledge of the database accepted biological data types. If the information is compatible, it is passed on to the Format Search Query module, which formats the query according to the type of data access interface and accepted input data. After that, the DB Connection module establishes the connection with the database in order to send the search query. Once the result arrives, it is passed on to the Result Analysis module, which filters out undesirable data or data with low statistical confidence or low quality values. Finally, the search results that passed the filtering process are formatted appropriately (module Format Result) and sent back to the Coordinator agent (Compose Reply Msg module). If the search query has a timeout, returns an error or no results, an error message is send to the Coordinator agent instead.

3. Conclusions

We propose an Agent-based enrichment system to improve genetic diversity analyses through the integration of different sources of biologically relevant data. This system will be used to analyse genetic diversity in Native and Latin-American populations and will be available to analyse variant data from other populations after the former is concluded. We believe that an agent-based approach is suited to implement such enrichment system given the heterogeneity of different types of genetic and biological data and their distribution over distinct sources over the Web. Moreover, Multiagent systems can cope with dynamic applications and, thus, can easily accommodate the incorporation of new data sources for complementary biological data that might be available in the future.

References

- Foster, I., Jennings, N., and Kesselman, C. (2004). Brain meets brawn: Why grid and agents need each other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 8–15. IEEE Computer Society.
- Myles, S., Tang, K., Somel, M., Green, R., Kelso, J., and Stoneking, M. (2008). Identification and analysis of genomic regions with large between-population differentiation in humans. *Ann Hum Genet*, 72(Pt 1):99–100.

Integrando Agentes Inteligentes e Valor Agregado para o Monitoramento e Controle de Processos de Software

Leandro L. C. de Souza, Emmanuel S. S. Freire, Mariela I. Cortés, Gustavo L. Campos

Centro de Ciências e Tecnologia – Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – Campus do Itaperi - Fortaleza – CE – Brasil

{leocadiodesouza, savio.essf}@gmail.com, {mariela, gustavo}@larces.uece.br

Abstract. This paper presents an agent-based approach to realize the monitoring and control of software process. For this to happen, information about cost and time associated with the project should be perceived and analyzed. The methodology used combines the notion of intelligent agents and the technique of Earned Value Management. Therewith, project managers and their staff can assess and measure performance and progress of the project, and to detect possible deviations (cost and time) in relation to what was planned.

Resumo. Este artigo apresenta uma abordagem baseada em agentes inteligentes destinados a realizar o monitoramento e controle de processos de software. Para que isso aconteça, informações relativas a custo e tempo associadas ao projeto, devem ser percebidas e analisadas. A metodologia utilizada combina a noção de agentes inteligentes e a técnica do Gerenciamento do Valor Agregado. Com isso, gerentes de projetos e sua equipe poderão avaliar e medir o desempenho e progresso do projeto, e detectar possíveis desvios (custo e tempo) em relação ao que foi planejado.

1. Introdução

A idéia de aplicar a tecnologia de agentes para resolver problemas de monitoramento e controle de processos de software veio da necessidade dos gerentes de projetos de acompanhar, de uma forma mais efetiva, a evolução do projeto. Esse acompanhamento está diretamente relacionado ao cumprimento de metas relativas a custo, tempo e escopo. Essas variáveis consistem de estimativas para o projeto, onde gerentes obtêm informações sobre o valor gasto e o tempo para a realização das atividades.

Segundo o *Project Management Body of Knowledge* (PMBoK) [PMI, Inc. (a) 2008] os processos de monitoramento são responsáveis por acompanhar, revisar e regular o progresso e o desempenho do projeto, identificar todas as áreas nas quais serão necessárias mudanças no plano e iniciar as mudanças correspondentes. Neste contexto, a técnica do Gerenciamento do Valor Agregado (GVA) [PMI, Inc. (b) 2005] é um método amplamente utilizado para monitorar o desempenho integrando as medidas de escopo, custos e tempo. Considerando a importância do monitoramento destas variáveis como forma de atingir a qualidade esperada [Sommerville 2007] e a complexidade dos processos envolvidos, o presente trabalho apresenta uma abordagem baseada em agentes inteligentes onde os agentes têm a função de observar e mensurar de forma periódica e uniforme o desempenho do projeto para identificar variações em relação ao plano de gerenciamento do mesmo e

recomendar ações preventivas em antecipação a possíveis problemas. As atividades do projeto são monitoradas em relação ao plano de gerenciamento e à linha de base de desempenho do mesmo. Adicionalmente, esses agentes são projetados para, a partir dos desvios detectados, propor ações de controle e, se possível, um novo plano de atividades.

Os agentes propostos foram concebidos empregando-se a arquitetura do agente reativo baseado em modelos (estado interno) e regras heurísticas [Russel e Norvig 2003].

Este artigo é estruturado como segue. A Seção 2 apresenta um referencial teórico sobre as técnicas e modelos utilizados na elaboração da proposta. A Seção 3 aborda os trabalhos relacionados. A Seção 4 apresenta uma descrição sobre o que realmente consiste esta abordagem. Na Seção 5, a abordagem é apresentada em um nível mais técnico, onde a operacionalização e formulação da proposta é melhor discutida. E por último, a Seção 6 apresenta as considerações finais.

2. Referencial Teórico

2.1. Gerenciamento do Valor Agregado (GVA)

O GVA pode ser brevemente definido como uma técnica de gerenciamento que compara as informações planejadas e realizadas de escopo, tempo e custo para medir o desempenho e estabelece uma tendência para o projeto [PMI, Inc. (b) 2005]. Essa técnica requer as informações de uma linha de base integrada a partir da qual, o desempenho possa ser medido durante o desenvolvimento do projeto. Essa linha de base nada mais é do que o (re)planejamento aprovado para todos os itens que compõem o projeto. Na abordagem aqui proposta, o escopo, o custo e o tempo formam o conjunto de itens do projeto.

2.2. Agentes Inteligentes

Um agente inteligente é uma entidade autônoma que percebe seu ambiente através de sensores e age sobre o mesmo utilizando atuadores [Russel e Norvig 2003]. Os agentes são usados para apoiar o desenvolvimento de sistemas de software em que os dados, controle, conhecimentos e/ou recursos são distribuídos. Além disso, os agentes fornecem uma metáfora natural para apoio em ambientes de equipe, onde esses agentes podem monitorar e coordenar mudanças e opiniões dentro da equipe de desenvolvimento [Balasubramanian et al. 2001].

3. Trabalhos Relacionados

A solução definida por [Golarath e Galorath 2006] tem como objetivo estimar e monitorar projetos de software, gerando índices de acompanhamento e métricas relevantes ao gerenciamento. Essa solução utiliza um modelo proprietário de estimativa e implementa o chamado *Parametric Project Monitoring and Control* (PPMC). Esse modelo utiliza o GVA para realizar o monitoramento, o qual é responsável por calcular os indicadores de desempenho do projeto de software. O controle é realizado através da metáfora “sinal de trânsito” para uma determinada medida ou métrica. Apesar das ações preventivas auxiliarem os gerentes de projeto no monitoramento e controle, elas não são suficientes para inibir todos os possíveis desvios em relação à duração e custo das atividades.

O Modelo de Gerenciamento de Projetos de Software apoiado por Agentes de Software (SPMSA), desenvolvido por [Nienaber 2008], tem como objetivo melhorar os processos no ambiente de Gerenciamento de Projetos de Software (SPM), abordando seus

aspectos intrínsecos. Esse modelo tem como foco o apoio às equipes e membros individuais das equipes em ambientes SPM durante a execução das tarefas. Para que esse auxílio seja possível, o monitoramento e controle dessas tarefas são realizados utilizando a tecnologia de agentes de software. Uma desvantagem desse modelo é que nenhuma proposta de correção de desvios (escopo, custo e tempo) ou notificação dos mesmos é apresentada.

4. Abordagem Proposta

A abordagem proposta neste artigo visa contribuir no planejamento contínuo do projeto de software com base na análise das informações obtidas a partir dos processos de monitoramento e controle. A partir de um planejamento inicial, agentes de software monitoram o desempenho das principais variáveis do projeto (escopo, tempo e custo) no intuito de avaliar e medir o desempenho geral do projeto. Este monitoramento é realizado avaliando o desempenho real do projeto em relação ao que foi planejado, determinando possíveis desvios levando em conta limites de variação pré-estabelecidos e demais indicadores propostos na teoria do valor agregado. Estes indicadores são incorporados através de regras implementadas na estrutura dos agentes de forma a auxiliar na detecção de desvios em relação à linha de base do projeto.

A partir das constatações levantadas durante o processo de monitoramento, o agente responsável pelo controle irá agir no sentido de propor ações que visem manter o projeto dentro de limites de variação aceitáveis. Caso essas variações ultrapassem os limites aceitáveis, esta abordagem propõe um replanejamento através da geração de novas estimativas. A princípio, vamos considerar que os gerentes de projeto entrem com essas estimativas (custo e tempo) e, a partir daí, a planejamento contínuo possa ser realizado através da estratégia de priorização de requisitos [Brasil 2011].

Diante deste contexto, esta abordagem envolve o desenvolvimento de dois agentes, um agente monitor (AMon) e um agente de controle (ACon), que podem funcionar de forma reativa ou pró-ativa. A Seção 5 apresenta os agentes propostos.

5. Detalhamento Técnico

A Estrutura Analítica do Projeto (EAP) consiste da decomposição hierárquica das entregas do trabalho a ser executado pela equipe para atingir os objetivos do projeto e criar as entregas requisitadas [PMI, Inc. (a) 2008]. A abordagem proposta considera que no início do processo o gerenciamento do plano do escopo e a EAP foram gerados e que o AMon e o ACon percebem estas informações. Os agentes serão descritos como segue:

Agente Monitor: Com base nas informações da EAP e linhas de base de orçamento e cronograma, o agente AMon é capaz de detectar diferenças entre estas informações correntes e situações desejadas estabelecidas previamente no projeto (metas), e, sempre que necessário enviar para o gerente mensagens que visam indicar a presença de riscos ao cumprimento das metas. No AMon, as regras heurísticas, conforme mencionado na Seção 1, consistem de um conjunto de associações comuns observadas entre certas condições extremas (positivas ou negativas), estabelecidas a partir das descrições em E_i (estado interno) e certas mensagens e suas consequências descritas em ações: *se condição_extrema(E_i) então reportar_mensagem(A)*.

Agente de Controle: A partir das mensagens emitidas pelo agente AMon para o gerente das informações de linhas de base de orçamento e cronograma, e do feed-back do

projetista, o agente ACon é capaz de gerar um conjunto de ações que indiquem ao gerente alguns meios de eliminar ou reduzir as diferenças existentes entre a situação corrente e a situação desejada. No agente ACon as regras associam diferenças, calculadas a partir das descrições de estado corrente interno e de estados desejados (metas) oriundos do planejamento e incorporados em *Ei* (estado interno), e certas ações de controle em *Ação* que promete eliminar as diferenças: ***se diferenças(Ei) então enviar_sinal(A)***.

6. Considerações Finais

Este trabalho esboçou uma abordagem baseada na integração de agentes inteligentes e GVA para auxiliar gerentes de projeto e suas equipes no processo de monitoramento e controle dos custos, do escopo e do tempo das atividades do projeto.

Os agentes propostos foram esboçados de forma a detectar desvios, medir o desempenho e progresso do projeto e informar, em tempo real, a situação do projeto durante o processo de desenvolvimento e sugerir ações de controle para minimizar efeitos negativos. Esta abordagem está em construção e os resultados preliminares, ainda envolvendo casos bastante simplificados, são motivadores indicando que o trabalho é relevante e deve ser continuado, visto que a idéia promete minimizar custos e tempo, e abrirá portas para outros problemas característicos do processo de desenvolvimento.

Referências

- Balasubramanian, S., Brennan R. W., e Norrie, D. H. (2001) An architecture for metamorphic control of holonic manufacturing systems, Computer in Industry, vol. 46.
- Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clarck, B. K., Horowitz, E., Madachy, R., Reifer, D. J., Steele, B. (2000) Software cost estimation with COCOMO II, Prentice-Hall, Upper Saddle River, NJ.
- Brasil, M. M. A. (2011) Planejamento de *releases* de software através da aplicação de técnicas de busca multiobjetivas, MACC - Universidade Estadual do Ceará.
- Galorath, D. D., Galorath, J. (2006) Achieving software development success - using best practice planning, estimation, tracking and control. Em: Software Measurement European Forum, Roma. Proceedings. Roma: SMEF, p. 293-304.
- Nienaber, R. C. (2008) A model for enhancing software project management using software agent technology. Em: Universidade da África do Sul. Trabalho apresentado para o título de Doutor do Departamento de Ciência da Computação.
- Project Management Institute (PMI) (a) (2008) Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBoK), 14 Campus Boulevard Newtown Square, Pennsylvania 19073-3299, Estados Unidos, 4º Edição.
- Project Management Institute (PMI) (b) (2005) Practice Standart for Earned Value Management, 1º Edição. USA: Project Management Institute, Inc..
- Russel, S. e Norvig, P. (2003) Artificial Intelligent: A modern approach, Nova Jersey, Estados Unidos, 2º Edição.
- Sommerville, L. (2007) Engenharia de Software, Pearson Addison – Wesley. São Paulo.

Uma metodologia para modelagem de sistemas multiagentes

Daniela Maria Uez, Jomi F. Hübner

¹Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

{dmuez, jomi}@das.ufsc.br

Resumo. A área de engenharia de software orientado a agentes ainda não conta com uma metodologia completa, na qual seja possível modelar e projetar de forma integrada os vários componentes de um sistema multiagentes. Este artigo apresenta uma proposta para o desenvolvimento de uma metodologias na qual seja possível a modelagem dos sistemas multiagentes, integrando o agente, o ambiente, a interação e a organização. Essa metodologia será desenvolvida utilizando técnicas da engenharia de métodos situacional (SME).

1. Introdução

Nos últimos anos, muitas metodologias foram propostas para a área de Sistemas Multiagentes (SMA). De um modo geral, essas metodologias ainda não estão suficiente maduras e completas para que possam ser consideradas um padrão [Weyns et al. 2009]. Um exemplo disso é o fato de que muitas metodologias permitem somente a modelagem do agente e das interações entre os agentes e oferecem pouco ou nenhum suporte à modelagem do ambiente e da organização [Deloach 2005, Molesini et al. 2009]. Assim, um estudo que leve à definição de uma metodologia que permita a modelagem explícita desses quatro componentes do sistema multiagente (a saber, agente, interações, ambiente e organização) é uma contribuição importante para aumentar a maturidade da área. Este artigo apresenta o projeto para o desenvolvimento de uma metodologia que visa permitir a modelagem integrada desses quatro componentes, além de fornecer uma ferramenta capaz de gerar o código fonte do sistema modelado.

2. Metodologias Orientadas a Agentes

Um sistema multiagentes é formado por um conjunto de agentes, um ambiente, um conjunto de possíveis interações e uma organização [Demazeau 1995]. A modelagem dos sistemas multiagentes necessita de metodologias apropriadas, que auxiliem o programador a lidar com a complexidade desse tipo de sistema [Giorgini and Henderson-Sellers 2005, Bergenti et al. 2004]. Assim, desde o início dos anos 90, diversas metodologias foram propostas para permitir a modelagem desses sistemas. Apesar disso, a área de engenharia de software para sistemas multiagentes ainda carece de uma metodologia que permita a modelagem dos quatro componentes dos sistemas multiagentes definidos por Demazeau [Casare 2011]. Se por um lado as metodologias geralmente permitem que os aspectos relacionados ao agente e às interações entre os agentes sejam modelados, poucas são as que permitem a modelagem do ambiente e dos aspectos organizacionais [Deloach 2005, Molesini et al. 2009]. Na Tabela 1 são apresentadas algumas das principais metodologias existentes para

a modelagem de sistemas orientados a agentes e qual é a abrangência de cada metodologia com relação à modelagem do ambiente e da organização. As metodologias apresentadas são Adelfe [Beron et al. 2002], Gaia [Wooldridge et al. 2000], Ingenias [Pavón and Gómez-Sanz 2003], Message [Caire et al. 2002], O-MaSE¹ [DeLoach and García-Ojeda 2010], Passi [Cossentino and Potts 2002], Prometheus [Winikoff and Padgham 2004] e Tropos [Mylopoulos et al. 2001].

Table 1. Componentes dos sistemas multiagentes contemplados pelas metodologias. Adaptado de [Casare 2011] e [Molesini et al. 2009]

	Ambiente	Organização
Gaia	+-	+
O-MaSE	+-	+
Prometheus	+-	-
Tropos	-	-
ADELFE	++	-
PASSI	+-	-
MESSAGE	+-	+
Ingenias	+-	+

A modelagem explícita do ambiente, incluindo os objetos que o compõem e as interações entre os agentes e o ambiente, permite que se identifique como os agentes afetam os objetos existentes no ambiente, como o ambiente influencia no comportamento dos agentes e como se dá a interação direta e indireta entre eles através do ambiente [DeLoach and Valenzuela 2006]. Além disso, permite que sejam atingidos alguns conceitos computacionais desejáveis, como a separação dos conceitos relativos ao ambiente, a reusabilidade dos modelos de ambiente, o desenvolvimento de modelos genéricos do ambiente e a possibilidade de alteração dinâmica do ambiente [Ricci et al. 2011]. Para modelagem do ambiente, é necessária a modelagem de duas abstrações principais: as abstrações de ambiente e as abstrações de topologia. Uma abstração de ambiente é uma entidade que encapsula funções ou serviços para os agentes. Já uma abstração de topologia é vista como uma coleção de conjuntos vizinhos que provém uma noção estruturada de localidade para o sistema [Molesini et al. 2009]. Na Tabela 1, as metodologias classificadas com ++ são aquelas nas quais é possível modelar tanto as abstrações do ambiente quanto de abstrações de topologia; +- indica que a metodologia permite somente a modelagem de abstrações de topologia; +- indica somente a modelagem de abstrações do ambiente e - indica que a metodologia não permite a modelagem do ambiente.

A modelagem dos aspectos organizacionais, possibilita que agentes heterogêneos trabalhem em conjunto sem terem sido especificamente desenvolvidos para esse fim [Deloach 2005]. Apesar da autonomia dos agentes ser importante num sistema multiagente, esta autonomia pode provocar perda da coerência global do sistema. Assim, a organização procura estabelecer um comportamento global coeso que objetiva levar o sistema a atingir seus objetivos [Hübner and Sichman 2003]. Por isso, atualmente algumas metodologias foram alteradas para incluir a modelagem de conceitos organizacionais. Na

¹A metodologias O-MaSE é uma extensão da metodologia MaSE [Deloach 1999] que inclui o aspecto organizacional dos sistemas multiagentes. Por isso, optou-se por tratar somente a metodologia O-MaSE por ser mais completa.

Tabela 1 as metodologias que permitem a modelagem dos aspectos computacionais foram classificadas como + indica que a metodologia permite a modelagem da organização e - indica que a metodologia não permite a modelagem deste componente.

3. Uma solução possível

A proposta apresentada neste artigo pretende criar uma metodologia que permita a modelagem e o projeto de sistemas multiagentes que contemplem os quatro componentes - o agente, o ambiente, a interação e a organização. Essa metodologia será desenvolvida através da extensão de uma metodologia já existente, integrando nesta os modelos e componentes faltantes.

A pesquisa será desenvolvida em três etapas. Durante a primeira etapa, de revisão bibliográfica, serão estudados e avaliados os aspectos envolvidos no desenvolvimento de sistemas multiagentes levando-se em conta não só a modelagem destes quatro componentes. Na segunda etapa, será definida qual metodologia será estendida e como será feita a integração dos componentes faltantes. Também será desenvolvida a ferramenta que permita a utilização desta metodologia estendida e a geração do código fonte para o sistema modelado. Por fim, na terceira etapa, a metodologia será testada e avaliada. Para tanto, a metodologia será utilizada para modelagem de um sistema multiagente que contemple os aspectos dos agentes, do ambiente, da organização e da interação entre agentes. Após, serão feitos testes com alunos de cursos de graduação ou pós-graduação com o objetivo de avaliar se a metodologia atende as necessidades de modelagem dos sistemas multiagentes.

Basicamente, a pesquisa será guiada pelas seguintes questões:

- Como os componentes podem ser integrados de forma a permitir que o sistema multiagentes seja modelado e projetado em todas as suas dimensões, mantendo a coerência entre os componentes?
- Quais são as vantagens do uso dessa metodologia?

References

- Bergenti, F., Gleizes, M.-P., and Zambonelli, F. (2004). Coordination infrastructures in the engineering of multiagent systems. In Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter 14, pages 273–296. Kluwer Academic Publishers.
- Bernon, C., Glize, P., Picard, G., and Glize, P. (2002). The ADELFE methodology for an intranet system design. In *PAOIS 2002*, pages 27–28.
- Caire, G., Coulier, W., Garijo, F. J., Gomez, J., Pavón, J., Leal, F., Chainho, P., Kearney, P. E., Stark, J., Evans, R., and Massonet, P. (2002). Agent oriented analysis using message/uml. In *Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering II*, AOSE '01, pages 119–135, London, UK. Springer.
- Casare, S. J. (2011). *Medee: a method framework for multiagent systems*. PhD thesis, Universidade de São Paulo.

- Cossentino, M. and Potts, C. (2002). Passi: a process for specifying and implementing multi-agent systems using UML.
- Deloach, S. A. (1999). Multiagent Systems Engineering: A Methodology and Language for Designing Agent Systems. In *Agent-Oriented Information Systems '99 (AOIS'99)*.
- Deloach, S. A. (2005). Engineering organization-based multiagent systems. In *LNCS*, pages 109–125. Springer.
- DeLoach, S. A. and García-Ojeda, J. C. (2010). O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems. *Int. J. Agent-Oriented Softw. Eng.*, 4(3):244–280.
- DeLoach, S. A. and Valenzuela, J. L. (2006). An agent-environment interaction model. In Padgham, L. and Zambonelli, F., editors, *Agent-Oriented Software Engineering VII, 7th International Workshop, AOSE 2006, Hakodate, Japan, May 8, 2006, Revised and Invited Papers*, volume 4405 of *LNCS*, pages 1–18. Springer.
- Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. In *Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo*, pages 117–132.
- Giorgini, P. and Henderson-Sellers, B. (2005). Agent-oriented methodologies: an introduction. In Giorgini, P. and Henderson-Sellers, B., editors, *Agent-Oriented Methodologies*. Idea Group Publishing.
- Hübner, J. F. and Sichman, J. S. a. (2003). Organização de sistemas multiagentes. In Vieira, R., Osório, F., and Rezende, S., editors, *III Jornada de MiniCursos de Inteligência Artificial JAIA03*, volume 8, pages 247–296. SBC.
- Molesini, A., Omicini, A., and Viroli, M. (2009). Environment in agent-oriented software engineering methodologies. *Multiagent Grid Syst.*, 5(1):37–57.
- Mylopoulos, J., Kolp, M., and Castro, J. (2001). Uml for agent-oriented software development: The tropos proposal.
- Pavón, J. and Gómez-Sanz, J. J. (2003). Agent oriented software engineering with INGENIAS. In *CEEMAS 2003*, volume 2691 of *LNAI*, pages 394–403. Springer.
- Ricci, A., Piunti, M., and Viroli, M. (2011). Environment programming in multi-agent systems: an artifact-based perspective. *AAMAS*, 23:158–192.
- Weyns, D., Parunak, H. V. D., and Shehory, O. (2009). The future of software engineering and multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 2(1):369–377.
- Winikoff, M. and Padgham, L. (2004). *Developing Intelligent Agent Systems: A Practical Guide*. Halsted Press.
- Wooldridge, M., Jennings, N. R., and Kiny, D. (2000). The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3:285–312.

Uma nova abordagem para a integração da interação em um sistema multiagente

Maicon R. Zatelli¹, Jomi F. Hübner¹

¹ Departamento de Automação e Sistemas

Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

{maicon, jomi}@das.ufsc.br

Resumo. *Interação é um dos aspectos chave no projeto de sistemas orientados a agentes. Isso permite que agentes colaborem a fim de realizar tarefas complexas. Neste trabalho fazemos um levantamento de algumas publicações recentes a respeito da interação e propomos a criação de um novo modelo de interação que integre a interação com os demais componentes de um sistema multiagente, como o ambiente, de maneira unificada e coerente.*

1. Introdução

Segundo a abordagem AEIO (*Agent, Environment, Interaction, Organization*), os componentes básicos de um sistema multiagente (SMA) são agentes, ambiente, interação e organização. Para um dado problema a ser resolvido, deve-se escolher o modelo de agentes, o ambiente, a interação e a organização a serem instanciados. Portanto, um SMA não envolve somente os agentes, mas também outros elementos. O desenvolvedor deve ser capaz de diferenciar claramente cada uma dessas partes [Demazeau 1995].

A interação é uma peça importante em um SMA. Tipicamente um agente sozinho não controla todos os recursos ou tem a habilidade de executar todas as ações necessárias para atingir seus objetivos. Para isso ele deve interagir com outros agentes que fazem o controle destes recursos ou que podem realizar tais ações. Desse modo, a interação tornou-se uma questão indispensável que deve ser levada em consideração nas diferentes fases do desenvolvimento de uma aplicação multiagente [Cabri et al. 2002].

Neste artigo nós apresentamos uma revisão bibliográfica sobre alguns trabalhos recentes relacionados à interação e propomos um projeto para o desenvolvimento de um modelo de interação que integre os agentes, a organização e o ambiente de forma unificada e coerente. Essa solução deve ser disponibilizada juntamente com uma plataforma de SMA, o que permite aos desenvolvedores a utilização da nova abordagem.

2. Trabalhos Relacionados

Munindar [Singh 2011] defende a ideia de que a chave para a existência da programação orientada a interação é o tratamento da interação como entidade de primeira classe, que ajuda na criação de sistemas cuja participação de agentes pode ser projetada e operacionalizada de maneira independente. Em sua proposta, todas as informações relevantes, que afetam o estado social de uma interação, estão explicitamente colocadas dentro do protocolo de comunicação, nos valores dos parâmetros das mensagens que estão sendo trocadas. A parte de programação lógica interna de cada agente não é relevante, e isso é a motivação chave para a programação orientada a interação. Outra contribuição da sua

proposta é a composição de protocolos, que facilita o reuso de protocolos já projetados e implementados.

A fim de integrar a interação com a organização [Boissier et al. 2010] propõe utilizar o *framework* Easi [Saunier and Balbo 2009] com o objetivo de estender o Moise [Hübner et al. 2002] para que seja possível, por meio da especificação normativa, induzir os agentes a trabalharem com certos protocolos. A nova dimensão chamada de *communication mode specification* (CS) é definida como (*type*, *direction*, *protocol*), onde *type* identifica o tipo de comunicação (direta ou indireta), *direction*, identificando a direção (unidirecional ou bidirecional) e *protocol*, que identifica qual protocolo está sendo usado (FIPA-request, Publish-Subscribe, etc). A especificação normativa do Moise é definida como (*id*, *c*, *p*, *dm*, *object*), onde *id* é o identificador da norma, *c* é a condição de ativação, *p* o papel em que a modalidade deôntica é suportada, *dm* a modalidade deôntica e *object* que é o conteúdo da norma. O *object* da norma foi definido contendo duas expressões: *do* (*m*), onde a missão *m* precisa ser executada, ou *use* (*l*, *cm*, *a*), onde o modo de comunicação *cm* deve ser usado na ligação *l* no contexto *a*.

Um segundo trabalho [Hübner et al. 2010], desenvolvido com o objetivo de adicionar uma quarta dimensão ao Moise, parte do princípio que, em um sistema multiagente é possível classificar os níveis estruturais em dois níveis distintos: o populacional e o organizacional. O primeiro corresponde aos agentes em si e aos mecanismos relacionados a eles, como a interação e a comunicação. O segundo é relacionado aos papéis que podem ser assumidos pela população do sistema multiagente. Essa proposta visa a introdução de uma nova dimensão no Moise, focada na comunicação entre papéis no nível organizacional, e não na comunicação entre agentes no nível populacional. A fim de integrar essa nova dimensão nas outras do modelo Moise, novas relações foram adicionadas na especificação normativa, que são responsáveis por indicar quais protocolos devem ou podem ser usados para atingir as metas que constituem as missões dos papéis.

Com uma abordagem diferente, [Baldoni et al. 2010] visa integrar a interação com o ambiente. Sua justificativa baseia-se no fato de que a interação era limitada somente a atos de fala e isso não era sempre a forma mais natural e real de se fazer as coisas. Como exemplo, no sistema de votos no mundo real as pessoas frequentemente usam as mãos ao invés de dizer o nome do candidato de sua escolha. Se os ambientes fossem representados explicitamente seria possível usar um número maior de ações, que podem ser percebidas pelos agentes através do ambiente, sendo que estas ações também seriam um meio de se comunicar com outros agentes. Portanto, o projeto MERCURIO objetiva adicionar uma extensão ao padrão FIPA ACL, onde a comunicação, baseada em FIPA ACL, é integrada com formas de interação que são habilitadas e mediadas pelo ambiente. Estas novas formas de interação são relacionadas ao modelo de ações e percepções que os agentes podem efetuar por meio do ambiente no qual estão situados.

3. Avaliação do estado da arte

Após uma visão geral de cada trabalho, nota-se que os mesmos, em relação a interação, não abrangem todos os componentes de um SMA em uma abordagem única. A Tabela 1 compara estas propostas em relação ao que elas cobrem em cada integração da interação com outro componente. Um trabalho que possui (+) indica que este possui algum su-

porte de integração entre a interação e aquele componente identificado pela coluna. Um trabalho que possui (++) indica que este possui um bom suporte de integração. Por fim, um trabalho que possui (–) quer dizer que não há suporte para aquela integração.

Tabela 1. Comparação entre trabalhos de interação.

Trabalho	Int. x Agente	Int. x Organização	Int. x Ambiente
[Singh 2011]	++	+	-
[Boissier et al. 2010]	++	++	-
[Hübner et al. 2010]	++	++	-
[Baldoni et al. 2010]	++	-	++

Partindo deste ponto, pretende-se propor um modelo de interação nos SMA que integre os agentes, a organização e o ambiente de forma unificada e coerente. Unificada no sentido de abranger todos os componentes de um SMA em uma abordagem única, e coerente visando manter na interação os mesmos conceitos utilizados nos outros componentes. Por exemplo, o papel que é utilizado na interação é o mesmo papel que existe no modelo organizacional e não há metas na interação que não estejam especificadas no modelo organizacional.

A fim de exemplificar a problemática existente, pode-se imaginar que um grupo de agentes precise executar uma eleição segura para escolher um coordenador que deve negociar com o coordenador de outro grupo. Uma eleição segura é uma eleição em que não há um agente como responsável pela contabilização de votos, mas sim há uma estrutura segura (urna) para computar os votos e informar o resultado final. Entre os passos do protocolo, tem-se que todos os agentes devem executar uma ação de votar sobre um artefato urna. Ao computar todos os votos dos agentes, a urna faz a apuração e notifica o vencedor para que este assuma o papel de coordenador. Ao assumir esse papel, o agente recebe quais metas deve realizar. Dentre elas, está uma meta de negociar, que está diretamente vinculada a um protocolo de negociação. Tal protocolo informa ao agente para que contate um outro agente com o papel de coordenador. A fim de descobrir quem são os coordenadores, basta o agente ler o modelo organizacional. Não é necessário o agente enviar mensagens a diversos agentes pedindo se esses são coordenadores.

Seguindo as abordagens existentes não é possível representar tal cenário de forma completa. Em [Singh 2011] não há a representação do ambiente e nem mesmo de um modelo organizacional, logo, tanto a ação de votar, como a descoberta de agentes em determinado papel, não são suportados. Em [Boissier et al. 2010] e [Hübner et al. 2010] não são previstas ações no ambiente, logo, a ação de votar não pode ser realizada. Por fim, em [Baldoni et al. 2010] não há coerência com um modelo organizacional e isso impede que agentes descubram quais agentes estão em cada papel e até mesmo qual protocolo deve-se seguir para efetuar a negociação.

4. Considerações finais

A fim de validar o modelo, espera-se integrá-lo na plataforma JaCaMo [Boissier et al. 2012]. O JaCaMo é um projeto que tem o objetivo de permitir ao desenvolvedor separar cada um dos componentes de um SMA. Atualmente ele tem considerado os componentes de agentes, ambiente e organização, porém o componente

de interação ainda não foi adequadamente integrado. O JaCaMo, portanto, mostra-se ser o cenário ideal para a integração da proposta.

Ao término do projeto pretende-se também elaborar critérios e avaliar a solução desenvolvida. Neste último caso serão considerados tanto os aspectos relacionados à organização do código de uma aplicação desenvolvida utilizando a nova abordagem, como também da execução do SMA. Tratando-se de organização de código há critérios como separação de conceitos, ou seja, como estão separados os componentes no código fonte, bem como a capacidade de reuso do código, legibilidade, entre outros. Quanto a execução de SMA, pode-se verificar quesitos de escalabilidade, desempenho e robustez. A escalabilidade e desempenho serão medidos variando-se a quantidade de agentes e a intensidade do uso da interação, enquanto que a robustez pode ser medida construindo diversos protocolos, desde mais simples até outros mais elaborados. Outro objetivo da avaliação é comparar a nova proposta em relação às demais existentes. Nesse caso será elaborada uma tabela comparativa mostrando os componentes atendidos pela solução desenvolvida e as outras propostas.

Referências

- Baldoni, M., Baroglio, C., Bergenti, F., Boccalatte, A., Marengo, E., and Martelli, M. (2010). Mercurio: An interaction-oriented framework for designing, verifying and programming multi-agent systems. In *Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010)*, pages 134–149.
- Boissier, O., Balbo, F., and Badeig, F. (2010). Controlling multi-party interaction within normative multi-agent organizations. In *Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010)*, pages 17–32.
- Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2012). Jacamo project. Disponível em <http://jacamo.sourceforge.net/>.
- Cabri, G., Leonardi, L., and Zambonelli, F. (2002). Separation of concerns in agent applications by roles. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 430–438, Washington, DC, USA. IEEE Computer Society.
- Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. In *1st. European Conference on Cognitive Science, Saint-Malo*, pages 117–132.
- Hübner, A., Dimuro, G. P., Costa, A. C. R., and Mattos, V. L. D. (2010). A dialogic dimension for the moise+ organization model. In *Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010)*, pages 21–26.
- Hübner, J. F., Sichman, J. S., and Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, pages 118–128.
- Saunier, J. and Balbo, F. (2009). Regulated multi-party communications and context awareness through the environment. In *Journal on Multi-Agent and Grid Systems*, pages 75–91.
- Singh, M. P. (2011). Information-driven interaction-oriented programming: Bspl, the blindingly simple protocol language. In *10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 491–598.