Universidade Federal de Santa Catarina
Centro Tecnológico
DAS410058-41 - **Aprendizado de Máquina -** 2020/3
Departamento de Automação e Sistemas
Professor: Jomi F. Hübner
Trabalho prático - Gold Miners
Acadêmica: Lara Popov Zambiasi Bazzi Oberderfer

Relatório Gold Miners

Objetivo: O objetivo do trabalho é implementar soluções baseadas em grupos de agentes no contexto do Gold Miners (inicialmente proposto em https://multiagentcontest.org/2006/).
Detalhes: A partir do código disponível em http://jacamo.sourceforge.net/tutorial/gold-miners/, que é focado basicamente na dimensão dos agentes,
a) resolva os exercícios;
b) identifique onde artefatos podem ser utilizados para melhorar a solução; implemente tais artefatos e avalie as vantagens/desvantagens;
c) identifique onde uma organização pode ser utilizada para melhorar solução; implemente tal organização e avalie as vantagens/desvantagens; e
d) melhore o desempenho do time.

Introdução

O objetivo do trabalho é implementar soluções baseadas nos recursos de Jason, JaCaMo e com utilização de organização de Moise no contexto de um Jogo do tipo Minerador de Ouro (Gold miners).

Wooldridge e Jennings (1997) em sua definição de agentes, listam as seguintes características: reatividade é perceber o ambiente e responder de maneira oportuna (timely fashion) a mudanças que acontecem nele; autonomia é a capacidade de operar sem intervenção direta de humanos ou outros e ter alguma espécie de controle sobre suas ações e estado interno; habilidade social é a capacidade de interagir com outros agentes via algum tipo de linguagem de comunicação de agentes; pró-atividade é a capacidade de exibir comportamentos baseados em objetivos.

O SMA foi programado em uma versão estendida da linguagem AgentSpeak(L), por meio do interpretador **Jason** é baseado em sistemas de crenças, desejos e intenções (BDI - *Beliefs, Desires and Intentions*); **CArtAgO** é um framework e infraestrutura para programação e execução de ambientes baseados em artefato e **Moise** é um modelo organizacional para sistemas multiagentes baseado em noções como funções, grupos e missões. Ele permite que um MAS tenha uma especificação explícita de sua organização.

Segundo Boissier et al. (2011), a plataforma JaCaMo decorre de várias propostas, pois permite o desenvolvimento de sistemas multiagentes que além de envolvem muitas entidades autônomas, elas também interagem de formas complexas por meio de estruturas sociais de coordenação e normas que regulam o comportamento social geral em um ambiente compartilhado. JaCaMo é uma plataforma que integra as três dimensões fundamentais da programação de sistema multiagente: dimensão de agente; dimensão de ambiente; e dimensão de organização, realizando assim o modelo AEO (Agent, Environment, Organization.

Nessa direção a importância da combinação das três dimensões trazem muitos benefícios (BOISSIER et al., 2011):

- a base de ações dos agentes é dinâmica porque depende dos artefatos disponíveis;
- as ações ganham uma semântica baseada em processos, o que permite definir ações de longo prazo, bem como coordenar ações;
- uma noção explícita e bem definida de sucesso/falha para as ações é fornecida;
- a interação entre os agentes e as organizações é obtida uniformemente usando os mesmos mecanismos que permitem a interação agente-artefato;
- as organizações podem ser remodeladas dinamicamente agindo nos artefatos ;
- a delegação de objetivos organizacionais aos agentes é facilitada

Desenvolvimento do Gold-Miner

Inicialmente, para implementação do projeto, seguimos o tutorial disponível em: http://jacamo.sourceforge.net/tutorial/gold-miners/

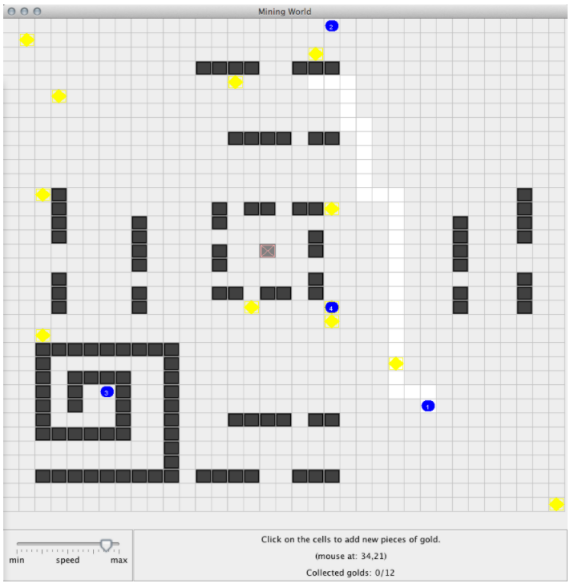


Figura 1: Print do Mapa 6 do Gold Mining

O ambiente é um mundo é a uma grade (Figura 1), tem vários mapas disponíveis para utilizar, podendo ser criado outros, onde os agentes podem se mover de uma célula para outra célula vizinha, caso ela não contenha nenhum agente ou um obstáculo.

Os agentes, operando como uma equipe, devem explorar o ambiente, evitar obstáculos e coletar o máximo de ouro possível para ser levado a um depósito. Cada agente pode carregar um ouro por vez (um agente que não carrega ouro é livre).

Os agentes têm apenas uma visão local de seu ambiente porque eles só podem ver peças de ouro nas células adjacentes; mas, eles podem se comunicar com os outros agentes para compartilhar suas descobertas.

A equipe é composta por **duas** funções desempenhadas por cinco agentes. A função de **minerador** é desempenhada por quatro agentes que terão como objetivo encontrar ouro e transportá-lo para o depósito.

A equipe também tem um agente desempenhando o papel de **líder**; seus objetivos são alocar quadrantes de agentes e alocar agentes livres para uma peça de ouro que foi encontrada.

O agente líder ajuda a equipe, mas cada equipe deve ter exatamente quatro agentes mineradores que se conectam ao servidor de simulação como competidores oficiais, para que o líder não seja registrado no servidor.

Para rodar no eclipse o projeto requer algumas bibliotecas (classpath) que estão na pasta /lib do projeto, precisa ser incluído (Configure Build Path >> Classpath >> Add External JARs >>)

- bin/cartago-2-5.jar
- gradle/wrapper/gradle-wrapper.jar
- /bin/jason-2.5.1.jar
- /bin/search.jar
- /bin/twitter4j-async-4.0.2.jar
- /bin/twitter4j-core-4.0.2.jar
- /bin/twitter4j-media-support-4.0.2.jar
- /bin/twitter4j-stream-4.0.2.jar

Modelagem do Gold-Miner com melhorias

Segundo Ricci et al. (2011) a noção de artefato em MAS foi introduzida pela primeira vez em 2008 no contexto da coordenação de Sistemas Multiagentes, em particular para definir as propriedades básicas das abstrações de coordenação de primeira classe, habilitando e gerenciando o agente interação, generalizando a noção de meios de coordenação. O conceito foi então generalizado além do domínio de coordenação, levando à definição do metamodelo A&A (Agentes e Artefatos) e ao desenvolvimento de um

framework computacional - CArtAgO - para apoiar o desenvolvimento e execução de ambientes projetados e programados sobre a noção de artefato.

O ambiente é concebido como um conjunto dinâmico de entidades computacionais chamadas artefatos, representando em geral recursos e ferramentas que os agentes que trabalham no mesmo ambiente podem compartilhar e explorar. O conjunto geral de artefatos pode ser organizado em um ou vários espaços de trabalho, possivelmente distribuídos em nós de redes diferentes. Um espaço de trabalho representa um lugar, o locus de uma ou várias atividades envolvendo um conjunto de agentes e artefatos (RICCI et al. 2011).

Artefatos

Do ponto de vista do agente, as operações de artefato representam ações externas fornecidas aos agentes pelo ambiente: este é um aspecto principal do modelo. Assim, em ambientes baseados em artefatos, o repertório de ações externas à disposição de um agente - além daquelas relacionadas à comunicação direta - é definido pelo conjunto de artefatos que povoam o ambiente (RICCI et al. 2011).

Conforme a Figura 1, pode-se perceber que o Agente Leader possui o artefato MiningArtifact, WorldMapArtifact e MiningPlanetArtifact, sendo que os dois primeiros artefatos foram desenvolvidos, e o MiningPlanet já fazia parte do sistema, e os Agentes mineradores usam esses artefatos.

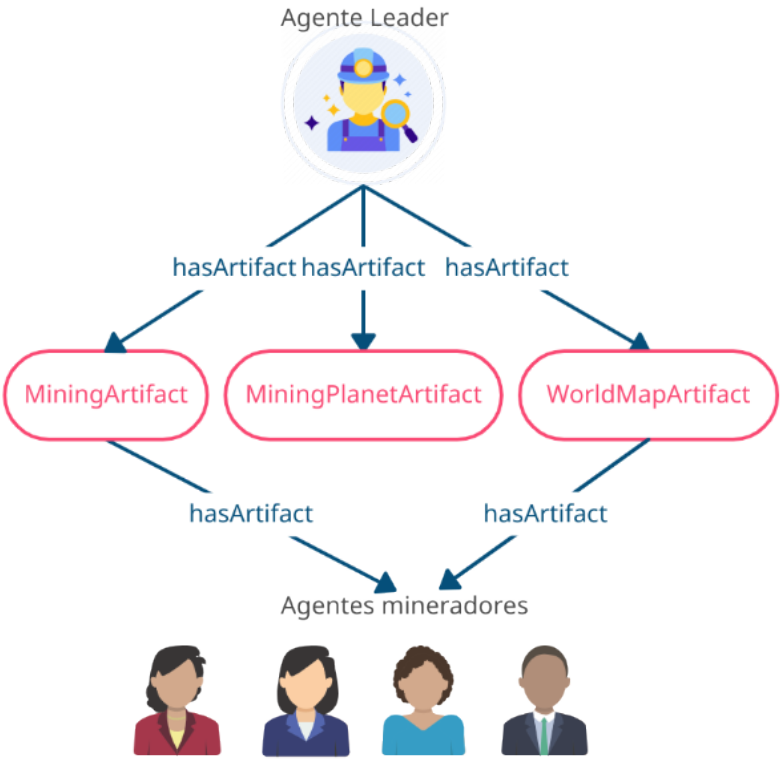


Figura 2: Diagrama de agentes e artefatos

Para a formação dos agentes pode-se observar no código abaixo que o agente lider segue o código do leader.asl e os agentes mineradores rosa e laranja seguem os codigos miner_new.asl e os agentes vermelho e verde seguem os códigos base do miner.asl.

```
mas gold_miners {
    agent lider : leader.asl
    agent minerador_rosa : miner_new.asl{
        focus: mining.m1view
        focus: mining.pinkMinerMap
    }
    agent minerador_laranja : miner_new.asl{
        focus: mining.m2view
        focus: mining.orangeMinerMap
    }
    agent minerador_vermelho : miner.asl{
        focus: mining.m3view
        focus: mining.redMinerMap
    }
    agent minerador_verde : miner.asl{
        focus: mining.m4view
        focus: mining.greenMinerMap
    }
...
```

Nas Figuras 3, 4 e 5 temos a modelagem dos diagramas dos artefatos MiningPlanet, WorldMap e MiningArtifact.

O artefato WorldMap tem como objetivo abstrair as ações do agente miner como pode ser visto na Figura 4. E o artefato MiningArtifact foi desenvolvido para abstrair as ações do agente leader, como pode ser visto na Figura 5

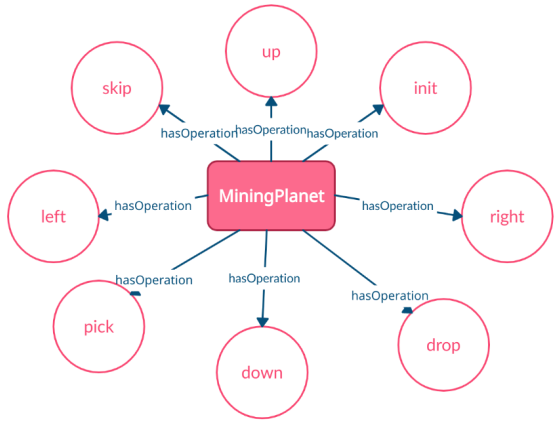


Figura 3:Diagrama do artefato MiningPlanet

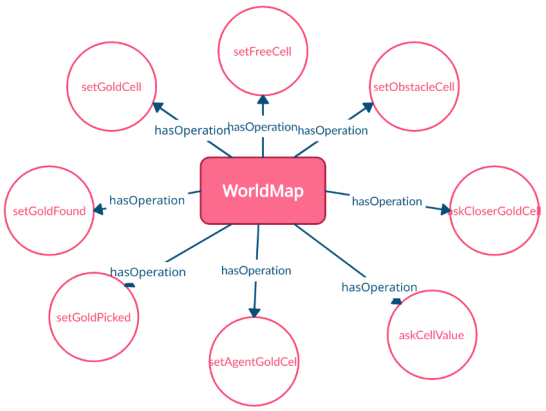


Figura 4: Diagrama do artefato WorldMap

Figura 5: Diagrama do artefato MiningArtifact

Para a criação do ambiente, workspace mining para os artefatos alguns trechos do código-fonte abaixo:

```
workspace mining {
        artifact m1view : mining.MiningPlanet(6,0)
        artifact m2view : mining.MiningPlanet(6,1)
        artifact m3view : mining.MiningPlanet(6,2)
        artifact m4view : mining.MiningPlanet(6,3)
        artifact pinkMinerMap   : mining.WorldMap("MinerPink")
        artifact orangeMinerMap : mining.WorldMap("MinerOrange")
        artifact redMinerMap    : mining.WorldMap("MinerRed")
        artifact greenMinerMap  : mining.WorldMap("MinerGreen")
}
```

Organização

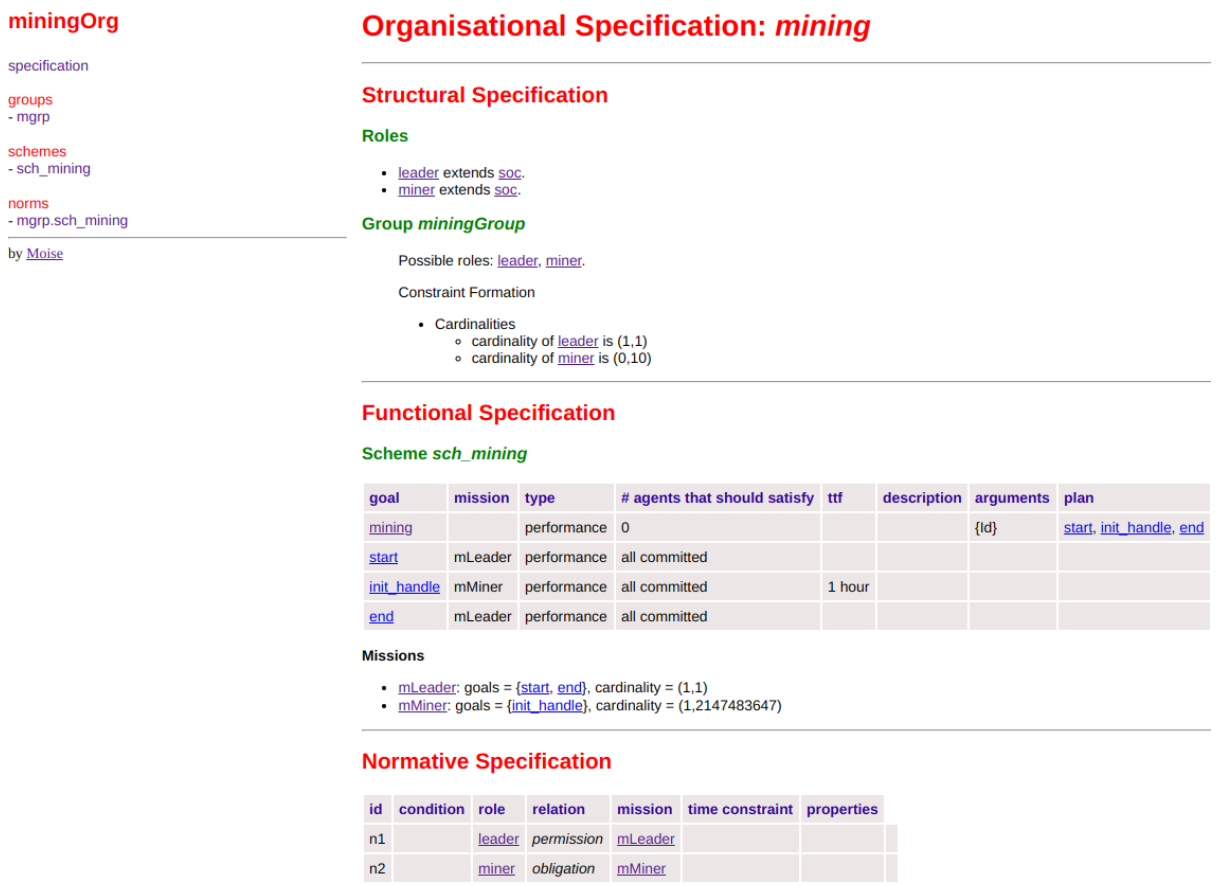
Segundo Hübner (2002), a Especificação Funcional (FS) descreve como um SMA geralmente atinge seus objetivos globais, decompostos em planos (objetivos) e distribuídos aos agentes em missões. Esse esquema pode ser visto como uma árvore de decomposição de objetivos, no qual, a raiz é um objetivo global e as folhas são objetivos que podem ser alcançados por um agente.

Essas decomposições podem ser definidas pelo projetista do MAS, que especifica sua experiência no esquema, ou pelos agentes que armazenam suas (melhores) soluções anteriores.

Para o esquema organizacional o nome da organização foi identificado como miningOrg para acessar o arquivo mining_os.xml, que possui a especificação do grupo mgrp: miningGroup que possui um leader com o jogador lider e tambem um miner com quatro jogadores: minerador_laraja, minerador_rosa, minerador_verde e minerador_vermelho que pode ser visualizado no código abaixo e na Figura 6.

```
organisation miningOrg : mining_os.xml {
        group mgrp : miningGroup {
            players: lider              leader
```

```
                    minerador_rosa     miner
                    minerador_laranja  miner
                    minerador_vermelho miner
                    minerador_verde    miner
        }
}
```



miningOrg

specification

groups
- mgrp

schemes
- sch_mining

norms
- mgrp.sch_mining

by Moise

Organisational Specification: *mining*

Structural Specification

Roles

- leader extends soc.
- miner extends soc.

Group *miningGroup*

Possible roles: leader, miner.

Constraint Formation

- Cardinalities
  - cardinality of leader is (1,1)
  - cardinality of miner is (0,10)

Functional Specification

Scheme *sch_mining*

goal	mission	type	# agents that should satisfy	ttf	description	arguments	plan
mining		performance	0			{Id}	start, init_handle, end
start	mLeader	performance	all committed				
init_handle	mMiner	performance	all committed	1 hour			
end	mLeader	performance	all committed				

Missions

- mLeader: goals = {start, end}, cardinality = (1,1)
- mMiner: goals = {init_handle}, cardinality = (1,2147483647)

Normative Specification

id	condition	role	relation	mission	time constraint	properties
n1		leader	*permission*	mLeader		
n2		miner	*obligation*	mMiner		

Figura 6: Esquema Organizacional da especificação mining gerado

Grupo

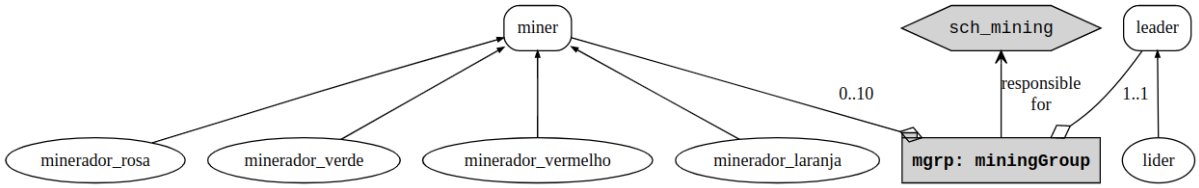
Para a formação do grupo, a especificação foi projetada e identificada por miningGroup, foram definidos dois papéis. O papel miner, com a cardinalidade de 0 até no máximo 10. e o papel leader com cardinalidade 1, com os jogadores lider jogando com o papel de leader, e com o papel miner os jogadores: minerador_laraja, minerador_rosa, minerador_verde e minerador_vermelho, como pode-se ver na Figura 7 e no código-fonte abaixo do arquivo xml.

mgrp (group)

created from specification miningGroup (*root* group)

Formation:

ok



Players

- lider plays leader
- minerador_laranja plays miner
- minerador_rosa plays miner
- minerador_verde plays miner
- minerador_vermelho plays miner

Responsible for the following schemes:

- sch_mining

Figura 7: Esquema Organizacional da especificação do Grupo miningGroup gerado

```
<structural-specification>
        <role-definitions>
                  <role id="miner" />
                  <role id="leader" />
            </role-definitions>

        <group-specification id="miningGroup">
            <roles>
                  <role id="leader" min="1" max="1"/>
                <role id="miner" min="1" max="10" />
            </roles>
        </group-specification>
    </structural-specification>
```

A especificação de esquema é uma árvore de decomposição de objetivo global. Essa decomposição é realizada por planos, portanto os principais elementos na especificação de um esquema são os planos e as metas. O plano para o esquema mining é a raiz, seguido das ações start e end que podem ser realizadas pelo mLeader assumido pelo agente lider, e a ação init_handle que pode ser realizada pelo mMiner assumida pelos agentes jogadores minerador_laraja, minerador_rosa, minerador_verde e minerador_vermelho.

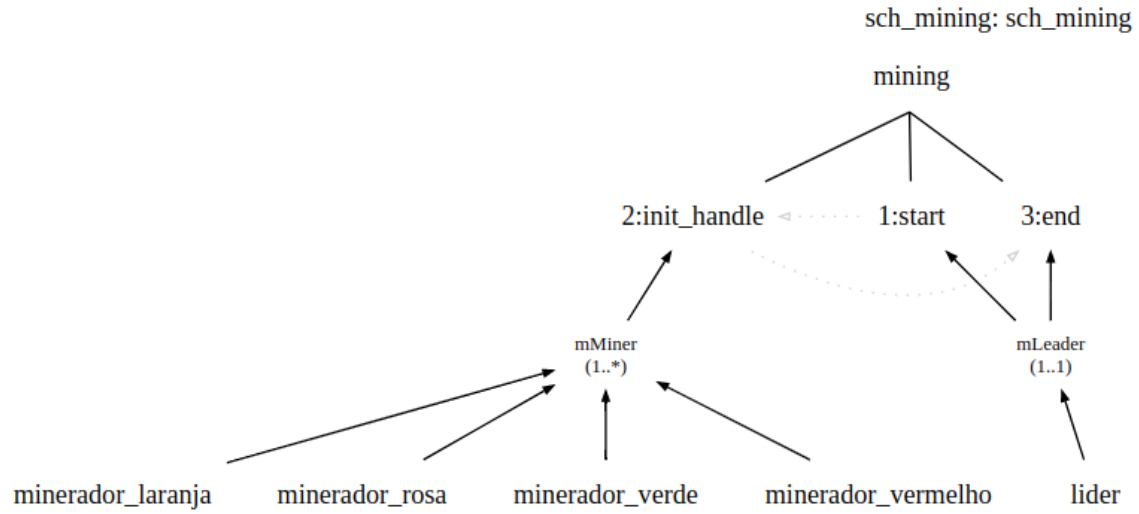
sch_mining (scheme instance)

created from specification sch_mining , owner is lider

Formation:

ok

Responsible groups: mgrp



Players

- lider committed to mLeader
- minerador_laranja committed to mMiner
- minerador_rosa committed to mMiner
- minerador_verde committed to mMiner
- minerador_vermelho committed to mMiner

Figura 8: Esquema Organizacional da especificação do esquema sch_mining gerado

Nesse esquema, **mining** é a meta raiz, e essa meta é alcançada por um plano composto recursivamente por uma sequência de cumprimento de outras metas. O atributo min de uma meta significa o número de agentes que devem satisfazer a meta de forma que ela seja considerada globalmente alcançada. O valor padrão para min é 'todos', o que significa que todos os agentes comprometidos com esta meta devem ser definidos como alcançados para o estado como globalmente alcançados. Cada objetivo tem, portanto, uma identificação única no esquema e uma descrição. Como podemos ver no código abaixo:

```
<functional-specification>
        <scheme id="sch_mining" >
            <goal id="mining">
            <argument id="Id" />
                 <plan operator="sequence">
                   <goal id="start" />
                   <goal id="handle" ttf="1 hour" />
                   <goal id="end" />
                 </plan>
               </goal>
```

```
            ...
        </scheme>
</functional-specification>
```

Uma missão é um conjunto de metas para o comprometimento de um agente no contexto da execução de um esquema. As missões são definidas da seguinte forma:

```
<functional-specification>
        ...
            <mission id="mLeader" min="1" max="1">
                  <goal id="start" />
                <goal id="end" />
            </mission>
            <mission id="mMiner" min="1">
                  <goal id="handle" />
            </mission>
        </scheme>
</functional-specification>
```

A cardinalidade das missões (os parâmetros mínimo e máximo) afirmam que apenas um agente pode ser comprometido com essas missões. O valor padrão para min é 1 e para max é 'ilimitado'.

goal	state	committed / achieved by	arguments	plan : dependencies
mining	waiting	[]/[]	Id = cartago.OpFeedbackParam@51f56bd9	start, init_handle, end : { *end* }
start	enabled	[lider]/[]		
init_handle	waiting	[minerador_laranja,minerador_rosa,minerador_verde,minerador_vermelho]/[]		{ *start* }
end	waiting	[lider]/[]		{ *init_handle* }

Normative State

state	agent	maintenance condition	aim	deadline	done at	annotations
active	lider	enabled(sch_mining,start)	**O** done(sch_mining,start,lider)	--		created = ["2021-4-1 10:24:3"] norm = [ngoal, [["A",lider],["S",sch_mining],["D","1970-11-4 21:0:0"],["G",start], ["M",mLeader],["What",done(sch_mining,start,lider)]]]

Figura 9: Esquema Organizacional da especificação do Grupo miningGroup gerado (continuação)

Normas

A especificação normativa (NS) declara as funções exigidas para missões e obrigações de missões para funções. Do ponto de vista desta simulação, criamos agentes, atribuímos a eles um papel (por exemplo, leader ou miner) e depois de uma missão (por exemplo, mLeader ou mMiner).

A Figura 10 apresenta o Esquema do Estado Normativo do resultado da simulação do Gold-Miners do projeto, no caso, como estado ativo os agentes mineradores laranja, rosa e verde

Normative State

state	agent	maintenance condition	aim	deadline	done at	annotations
active	minerador_laranja	responsible(mgrp,sch_mining)	**P** committed(minerador_laranja,mMiner,sch_mining)	--		created = ["2021-4-2 10:3:13"] norm = [pn2, [["MMinCard",1], ["A",minerador_laranja], ["S",sch_mining],["V",1],["Gr",mgrp], ["MMaxCard",2147483647]]]
	minerador_rosa	responsible(mgrp,sch_mining)	**P** committed(minerador_rosa,mMiner,sch_mining)	--		created = ["2021-4-2 10:3:13"] norm = [pn2, [["MMinCard",1], ["A",minerador_rosa],["S",sch_mining], ["V",1],["Gr",mgrp], ["MMaxCard",2147483647]]]
	minerador_verde	responsible(mgrp,sch_mining)	**P** committed(minerador_verde,mMiner,sch_mining)	--		created = ["2021-4-2 10:3:13"] norm = [pn2, [["MMinCard",1], ["A",minerador_verde], ["S",sch_mining],["V",1],["Gr",mgrp], ["MMaxCard",2147483647]]]
fulfilled	minerador_vermelho	not (well_formed(sch_mining))	**O** committed(minerador_vermelho,mMiner,sch_mining)	2022-4-3 10:3:13	--	created = ["2021-4-2 10:3:13"] fulfilled = ["2021-4-2 10:3:13"] norm = [n2, [["MMinCard",1], ["A",minerador_vermelho], ["S",sch_mining],["V",0],["Gr",mgrp], ["MMaxCard",2147483647]]]

Figura 10: Esquema do Estado Normativo gerado

```
<normative-specification>
  <norm id="n1"  type="permission"   role="leader"  mission="mLeader" />
  <norm id="n2"  type="obligation"   role="miner"   mission="mMiner" />
</normative-specification>
```

Resultados

Seguem prints dos resultados obtidos:

Figura 11: Tela da compilação do jogo

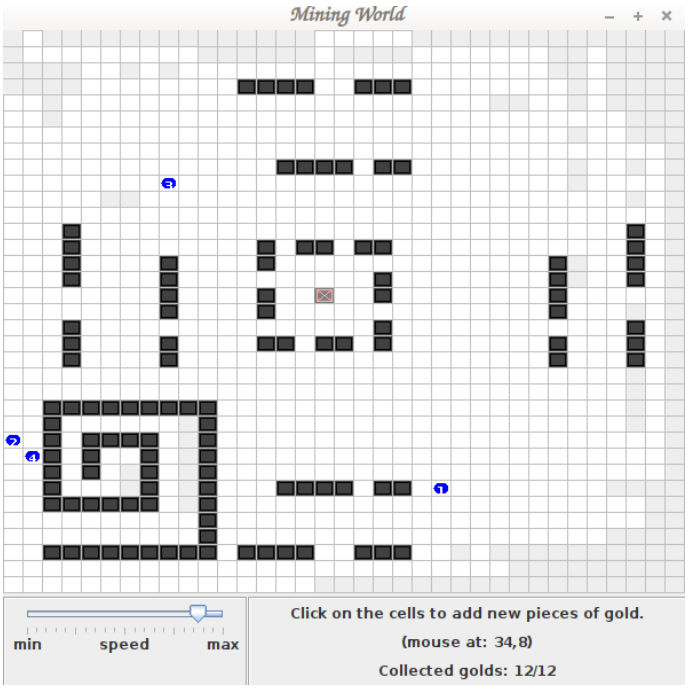


Figura 12: Tela da compilação do final do jogo

Melhora do desempenho

Para melhorias de desempenho e performance sugiro as opções melhora de busca e aprendizado por reforço.

Buscas

Também gostaria de ter realizado testes com os agentes utilizando outros tipos de buscas, o projeto utiliza a busca com A* e Heurística com Manhattan, que é um dos melhores algoritmos, mas gostaria de ter tido tempo para realizar testes com Busca em Largura, Busca em Profundidade, para testar a performance dos agentes.

Aprendizado por reforço

Para melhorar o desempenho dos agentes poderia ser desenvolvido um aprendizado por reforço, na qual o agente aprende por meio de uma política de ação para realizar uma tarefa através das suas interações com o ambiente, o labirinto, no caso. O agente também aprende como satisfazer suas motivações através das suas experiências passadas que produziu no ambiente.

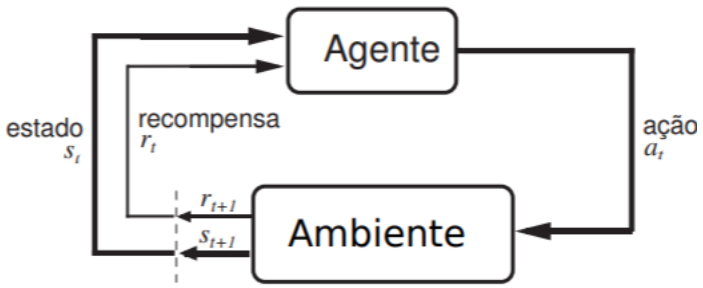


Figura 13: Interação agente-ambiente no aprendizado por reforço
(Adaptado de MITCHELL, 1997)

Para ganhar essas experiências ele interage repetidamente com o ambiente, a partir dessas interações o ambiente retorna um novo estado e uma recompensa numérica, essa recompensa é a entrada para que caminhos o agente deve seguir.

Por falta de tempo essa parte do projeto não foi implementada, mas fica para sugestão de melhoria dos agentes.

Referências Bibliográficas

Boissier, O., Bordini, R. H., Hubner, J. F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo, Science of Computer Programming, 2013, 747 – 761, ISSN 0167-6423, Special section: The Programming Languages track at the 26th ACM Symposium on Applied Computing (SAC 2011) & Special section on Agent-oriented Design Methods and Programming Techniques for Distributed Computing in Dynamic and Complex Environments.

CREATELY. Ferramenta online de criação de diagramas. Disponível em: https://app.creately.com/. Acesso em: mai. 2021.

M. Wooldridge and N. R. Jennings, “Intelligent agents: theory and practice,” Knowl. Eng. Rev., vol. 10, no. 02, p.115, Jun. 1995.

Firtina, U., Tezel, B. T., Challenger, M. and Kardas, G. (2018) “Abstract and Concrete Syntaxes for Software Agent Environment Modeling in CArtAgO Infrastructure”, In proceedings of the 3rd International Conference on Computer Science and Engineering (UBMK 2018), September 20-23, 2018, Sarajevo, Bosnia and Herzegovina, pp 622-626. Disponível em: http://akademik.ube.ege.edu.tr/~kardas/publications/conference_&_workshop_papers/2018_ubmk.pdf. Acesso em: mai. 2021.

MITCHELL, T. M.. Machine Learning. McGraw-Hill, 1997.

RICCI, A., PIUNTI, M. & VIROLI, M. Environment programming in multi-agent systems: an artifact-based perspective. Auton Agent Multi-Agent Syst 23, 158–192 (2011). https://doi.org/10.1007/s10458-010-9140-7