A Priorização de Tarefas em uma Alocação Dinâmica de Tarefas em Times de Agentes Heterogêneos

Derick P. Garcez[1], **Túlio L. Baségio**[1], **Rafael H. Bordini**[1]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul

derick.garcez;tulio.basegio<@edu.pucrs.br>,rafael.bordini@pucrs.br

Resumo. *As tarefas do nosso dia a dia podem por muitas vezes gerar um complicador na hora de organiza-las, e se as mesmas tem prioridades uma sobre as outras o problema pode ficar ainda mais complexo, da mesma forma se pensarmos em um conjunto de tarefas realizadas por um conjunto de pessoas a complexidade é alta, o mesmo acontece quando temos que definir um conjunto de tarefas que um robô ou vários robôs devem realizar, este problema se chama Alocação de Tarefas e é um problema NP-Difícil, mas podemos colocar mais complicadores para este problema e se as tarefas tem prioridades uma sobre as outras? A solução para este problema de prioridade sobre as tarefas é o que vamos tentar solucionar neste trabalho, utilizando uma implementação para a solução da alocação de tarefas como base.*

Palavras Chave: Alocação de Tarefas, Sistemas MultiAgente, Alocação de Tarefas com Prioridade.

1. Alocação de Tarefas com Prioridade

A cada dia o número de tarefas que devemos realizar em nosso dia a dia vem aumentando, a complexidade destas tarefas exigindo competências maiores da nossa parte também é algo frequente nas tarefas. Desta forma uma maneira de alocar de maneira eficiente estas tarefas, tanto em tempo como em qualidade é um grande desafio, as tarefas podem apresentar dependências entre elas, prioridade uma sobre as outras ou até competências que ainda não temos para executalas. Pensando desta forma como realizamos essa alocação das tarefas para um agente autônomo ? E se pensarmos em um sistema onde vários agentes precisam cumprir tarefas que estão em um ambiente? Estas respostas se enquadram em um problema já conhecido na computação, o problema da Alocação de Tarefas em um Sistema Multiagente, este problema está na categoria de problemas NP-Difíceis mostrando toda sua complexidade para possíveis soluções. Uma nova proposta para a alocação de tarefas em sistemas multiagente foi apresentada na tese de doutorado de Túlio Lima Basegio intitulada como ”Alocação Dinâmica de Tarefas em Times de Agentes Heterogêneos”[8], onde o professor Doutor Rafael Heitor Bordini foi orientador, o trabalho que será apresentado é uma contribuição para este trabalho, e conta com o Professor Doutor Rafael H. Bordini como orientador e Túlio L. Basegio como co-orientador. Na tese de doutorado as tarefas que estavam sendo alocadas não tinham prioridades, era apenas um conjunto de tarefas que poderiam ser classificadas em três tipos, como podemos ver na Tabela 1, a contribuição para o trabalho será criar um sistema de prioridades para cada tarefa, e depois realizar a alocação das tarefas por ordem de prioridade.

Com este contexto podemos então exemplificar de forma mais fácil qual o problema que estaremos tentando solucionar pensando em um contexto de desastres, este

Tabela 1. Tipos de Tarefas

Nome	Descrição
Atômica	Uma tarefa sem subtarefas, realizada por um único agente.
Simples	Uma tarefa com 1 ou mais subtarefas realizada por um único agente.
Composta	Uma tarefa com 1 ou mais subtarefas realizada por 1 ou k agentes

mesmo contexto foi utilizado na tese anteriormente.Em uma cidade onde uma enchente aconteceu podem existir diversas tarefas a serem realizadas pelos agentes, como resgatar pessoas em determinado local, realizar uma varredura em determinada região para identificação de sobreviventes, entregar um kit-médico para alguma equipe de resgate ou sobreviventes em situação de difícil acesso, tirar fotos de determinadas regiões, entre muitas outras tarefas que podem ser realizadas em uma situação de desastre, mas podemos pensar que algumas tarefas precisam receber uma prioridade maior na hora dos agentes realizarem, como resgatar uma pessoa tem uma prioridade maior do que tirar fotos de determinadas regiões, assim precisamos primeiramente realizar o resgate destas pessoas, desta forma vamos apresentar a solução para priorização nas tarefas.

2. Trabalhos Relacionados

Os trabalhos realacionados a este trabalho estão ligado diretamente a alocação de tarefas em um sistema multiagente, a base para essa contribuição é a tese de doutorado do Túlio, citada anteriormente, em cima desta implementação realizamos a priorização das tarefas. Pórem podemos utilizar outros trabalhos que tiveram uma pequena contribuição para este, o meu trabalho de replicação para a tese ”Uma Replicação do Trabalho ’Alocação Dinâmica de Tarefas em Times Agentes Heterogêneos’”[5] teve como objetivo conhecer e aprofundar o conhecimento na tese do Túlio, e o trabalho de Henrique Carvalho de Almeida Soares, ”Um estudo sobre o Problema de Alocação”[4] serviu como base de conhecimento para o problema de alocação de tarefas.

2.1. Alocação Dinâmica de Tarefas em Times de Agentes Heterogêneos

De forma simples podemos pensar que o problema de locação de tarefas é ordenar as tarefas que temos durante um dia, algumas destas tarefas apenas nós podemos realizar por que só nós temos as aptidões para realizar, porém existem outras que um conjunto de pessoas podem realizar, assim podemos organizar nosso tempo junto com outras pessoas.O problema de alocação de tarefas soluciona isso, a melhor forma que podemos alocar tarefas para agentes em um sistema baseado em tarefas. A tese de Doutorado de Túlio Lima Basegio serviu como base para a contribuição presente neste trabalho, abordando uma nova forma de realizar a alocação de tarefas, com um desempenho superior principalmente na etapa de trocas de mensagens entre os agentes presentes no ambiente, a proposta que foi orientada pelo Doutor Professor Rafael Heitor Bordini também fruto do meu estudo no trabalho ”Uma Replicação do Trabalho ’Alocação Dinâmica de Tarefas em Times Agentes Heterogêneos’”, utilizando a programação multiagente através do Jacamo[1] e a

programação linear cmo GLPK[7] para desenvolver uma nova proposta para o problema de alocação de tarefas em um ambiente multiagente. A tese realiza comparações com alguns algoritmos conhecidos da área como SSIA, ICBAA, entre outros, para então mostrar que a nova proposta de fato tem números iguais em alguns aspectos e muito melhores em outros.O foco da tese é trabalhar com robôs que vão atuar em uma situação de desastre, operando resgates, entregando kits, dentre outras tarefas.O sistema que foi criado para a alocação de tarefas inclui um pacote chamado *dataGenerator* o mesmo em linhas gerais realiza a criação dos agentes, das tarefas do ambiente, e com estes componentes cria os arquivos necessários para a execução do processo de alocação das tarefas.

3. A Solução

O processo de solução para a prioridade de tarefas foi pensando em cima da tese citada anteriormente, desta forma utilizando a estrutura que já hávia sido implementada foi pensanda uma nova estrutura para solucionar o problema.Foi utilizada a linguagem de programação JAVA[2] na sua versão 8, para a codificação e compilação foi utilizada a IDE NetBeans[3] na sua versão 8.2, todos os códigos fontes estão disponibilizados em um repositório do GitHub[6].

Pensando em linhas gerais devemos ter a prioridade de tarefa estabelecida em sua criação, ou seja deve vir como parâmetro ou já introduzida no sistema como uma tarefa de prioridade alta ou baixa, assim como o sistema não abordava esta situação e criava as tarefas de forma aletatória, também atribuímos de forma aleatória as prioridades das tarefas. Assim devemos pensar que cada prioridade de tarefa deve ser armazenada em algum lugar, criando então uma ordem para executalas. Pensando em um ambiente com **T** tarefas, realizamos a ordenação ou classificação destas tarefas em pequenas listas de acordo com suas **P** prioridades, ou seja vamos ter um ambiente com tarefas `T1,T2,T3,...`, cada tarefa tem uma prioridade `P1,P2,...` devemos organizar em listas de prioridades estas tarefas do ambiente, como podemos ver na Figura 1, podemos também ter prioridades que não tenham tarefas, ou seja uma prioridade muito alta que deve ser executada na hora, podemos ter um ambiente sem esta prioridade no determinado momento daquela alocação. Uma observação a ser feita, neste capítulo podemos utilizar a nomeclatura *Tasks*, a mesma é outra forma de representar Tarefas, a estrutura utilizada pelo Túlio na tese é de *Tasks*, porém nossa implementação fala sobre Tarefas, ambas são a mesma coisa.

A partir desta colocação das tarefas em listas de acordo com a sua prioridade, ordenamos as listas de prioridades e realizamos o processo de geração dos arquivos para a alocação de tarefas para cada lista de prioridades, caso a lista esteja vázia pulamos para a próxima lista, assim geramos os arquivos que serão rodados no Jacamo, e no GLPK.

3.1. DataGenerator

DataGenerator é o nome do pacote que está incluso na tese do Túlio, este pacote realiza a geração das tarefas, agentes, subtarefas, coloca aptidões para os agentes, as aptidões necessárias paras as tarefas serem realizadas, ele cria os dados para que a alocação seja realizada. No final de toda a geração ele cria arquivos de saída nos formatos corretos para a execução do Jacamo e do GLPK. Todas as definições de locais de saída dos arquivos, quantos agentes, quantas tarefas, quais os tipos de tarefas, quantas subtarefas , todas as definições são feitas através das variáveis globais presentes na classe principal *dataGenerator*.O pacote contém outras classes que são utilizadas pela *dataGenerator*, são elas:

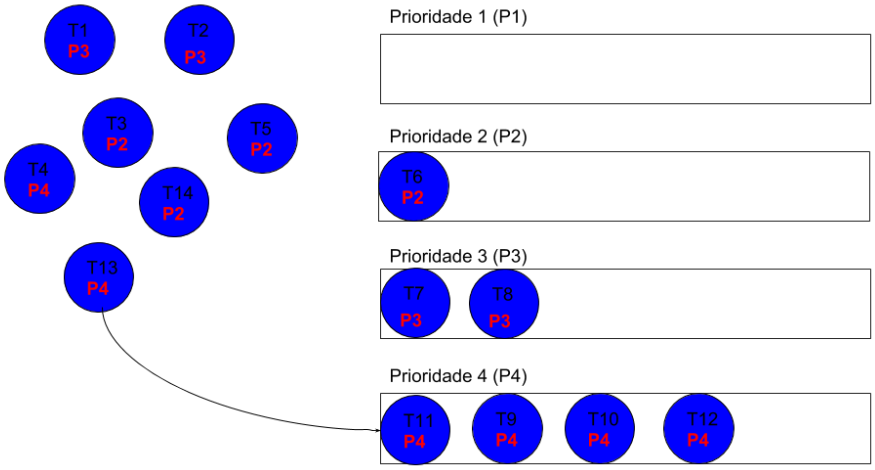

Figura 1. Exemplo de classificação de priorioridade nas Tarefas.

AppendToFile que trata a saída dos arquivos, *Tasks* que cuída da estrutura das tarefas, *Role* que são as aptidões, e por último *Agent* que é a estrutura dos agentes, as únicas classes que foram alteradas foram as *Tasks* e *dataGenerator*.

3.1.1. Priorização de Tarefas

Com o conhecimento do Problema da Alocação de Tarefas, podemos pensar que a ordenação ou criação das prioridades das tarefas devem ser executadas antes da alocação, assim devemos primeiro dizer quem são as tarefas prioritarias, para ai então podemos realizar a alocação das mesmas conforme a sua ordem. Nesta seção vamos abstrair os tipos de tarefas que podemos ter no sistema, como vimos na Tabela 1, e vamos apenas tratar as tarefas com tarefas simples, pois os tipos de tarefas não inteferem na prioridade das mesmas.Vamos utilizar um sistema de inteiros para ordernar a prioridade de cada tarefa, onde '1' seria a prioridade de mais alto nível e quanto maior o número menor a prioridade da tarefa.

A prioridade que cada tarefa tem é algo que deve ser definido de alguma forma, pode ser algo já definido que entra no sistema, ou algo que precisa ser mapeado e definido por algum tipo de critério, como nosso foco é apenas realizar a alocação destas tarefas de forma organizada, vamos atribuir a cada tarefa uma prioridade randômica utilizando a biblioteca *Math* do Java 8, mais especificamente o método *Random*, atribuímos a prioridade a cada tarefa. Esta etapa foi implementada no método *populateTasks*, a cada nova tarefa, que no método chamamos de *tasks*, atribuímos uma prioridade, como a estrutura criada utiliza um tipo de objeto chamado *Task* e também utliza uma *String* para armazenar as tarefas, realizamos a alterações nas duas estruturas para receberem a sua prioridade. A estrutura *String* apenas recebeu a adição de um inteiro dizendo qual o número da prioridade que está tarefa está atribuída. Já na estrutura do objeto *Task* foi adicionado ao seu construtor um novo parâmetro que recebe um inteiro como atribuição para a prioridade, além claro de um *get* para a consulta do valor atribuído.

Como mencionado anteriormente vamos ignorar o fato da existência de tipos de

tarefas, porém é válido comentar que como existem três tipos de tarefas, o método realiza a criação das tarefas de forma separada, assim a adição da atribuição randômica de prioridade foi feita para cada um dos três tipos de tarefas presentes.

A forma de armazenamento que foi vista na Figura 1, é mantida nesta etapa do processo, porém foi criada uma estrutura chamada 'Mariz de Tasks', ou Matriz de Tarefas, essa estrutura é uma matriz, que no eixo **X** identificamos qual é a prioridade,e no eixo **Y** navegamos pelas Tarefas que estão atribuídas para a aquela prioridade identificada por **X**, podemos entender melhor a estrutura pela Figura 2.

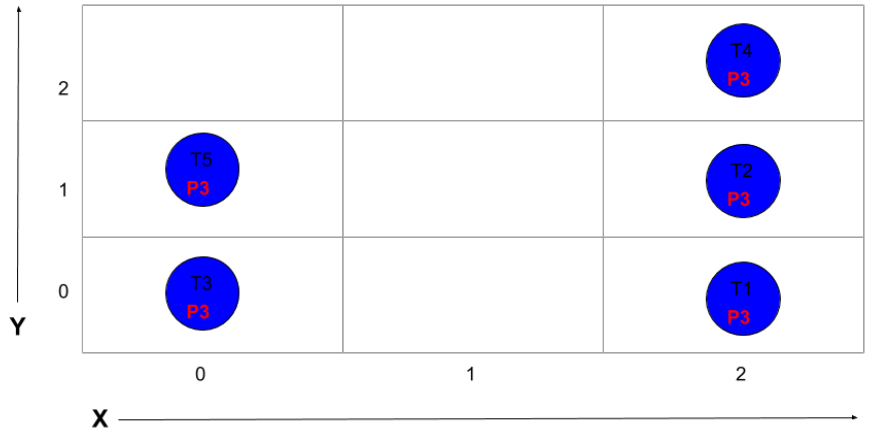

Figura 2. Exemplo de Matriz de Tarefas

Com esta estrutura, cada vez que o processo de criação de Tarefas cria uma nova tarefa, seja ela de qualquer tipo, definimos uma prioridade aleatória para esta tarefa e colocamos a mesma dentro desta estrutura, assim temos armazenado dentro de cada tarefa sua prioridade, mas ao mesmo tempo que criamos já classifícamos elas colocando em sua respectiva estrutura, assim não é necessára uma ordenação de um conjunto de tarefas e sim apenas a adição delas em seu devido lugar.

A forma aleatória que definimos é feita pela biblioteca *Math* utilizando o método *random*, assim gerando um número aleatório que é atribuído para a tarefa como prioridade.Utilizamos um parâmetro para a o método, onde colocamos um intervalo de inteiros que pode ser o resultado do método, nosso intervalo foi definido [1 a `nPrioridade`], sendo `nPrioridade` o número de prioridades dentro desta geração, esse número é difinido pela variável que tem o mesmo nome.A decisão de utilizar uma forma aleatória para atribuir prioridades para as tarefas é dada por que a prioridade das tarefas teria que ser algo vindo de fora, ou definido anteriormente a criação destas tarefas no sistema, assim como não temos esta definição foi preferido gerar de forma aleatória a prioridade das tarefas, até por que estamos preocupados com a alocação das tarefas, e não quais são as tarefas que estamos alocando por este mesmo motivo não utilizamos nomes nas tarefas, e sim apenas Tarefa 1, Tarefa 2.

3.2. Geração dos arquivos

Com a estrutura da Matriz de Tarefas, vista na seção anterior, a geração dos arquivos são realizadas nos métodos *dataAgents_toFile, dataAnnouncer_toFile,dataPlanner_toFile,*

dataGLPK_toFile, dataAnnouncer_toFile, esses métodos todos realizam a geração de todos os arquivos necessários para a execução da alocação.Assim o procedimento que realizamos altera um pouco a estrutura anterior do *dataGenerator*, antes esses métodos eram executados apenas uma vez, agora na nova estrutura para cada prioridade esse método é executado, então se temos 3 prioridades, vamos executar estes processos 3 vezes.Então cada vez que vamos chamar esse conjunto de métodos atribuímos nas variáveis as quais eles identificam as tarefas, as tarefas daquela prioridade, fazendo com que o método rode exatamente como foi feito, mas agora com tarefas da prioridade certa.Os arquivos que estes processos geram são salvos em pastas de acordo com a prioridade, ou seja a prioridade 2, tem uma pasta chamada 'Prioridade2' dentro do caminho que o foi identificado nas variáveis do *dataGenerator*.

3.3. Execução da alocação

Com os arquivos gerados, devemos realizar o processo de alocação das tarefas, foram gerados 2 arquivo, são eles: *execGLPK* e *execJacamo*. Estes dois arquivos são arquivos no formato *.bat* e então são linhas de comandos que o sistema operacional, no caso Windows, identifica e realiza as ações, as linhas que estão dentro destes arquivos indicam aonde estão os arquivos que precisam ser lidos pelos progamas Jacamo e GLPK, os arquivos indicam aonde estão os arquivos para cada prioridade, e executam para cada prioridade uma alocação de tarefas, então desta forma para cada prioridade começando da de mais alto nível, realizamos a alocação de tarefas com a proposta de solução da tese do Túlio. Para de fato realizar a alocação apenas é necessário executar os arquivos *execGLPK* e *execJacamo*, ambos vão gerar os resultados dentro das pastas de cada prioridade, os resultados do GLPK é necessário utilizar a classe implementada na tese do túlio para ler os resultados de saída, a classe se chama *ReadGLPKResults1* e necessário alterar as variáveis que localizam quais arquivos você deseja fazer a leitura.

4. Resultados

Alguns resultados foram obtidos utilizando a proposta de solução para alocação de tarefas, estes resultados são muito parecidos com os encontrados no trabalho de replicação da tese do Túlio, porém como agora estamos trabalhando com prioridades, e executamos para cada prioridade uma alocação, vamos análisar a execução conforme aumenta a quantidade de prioridades, variando a quantidade de tarefas presentes também. Nestes resultados obtidos estamos ignorando o tempo que o *dataGenerator* leva para gerar os arquivos, pois o tempo é abaixo de 1 segundo praticamente em todas as execuções.

A primeira análise feita leva em conta a variação do número de prioridades como alvo, então para cada 3 quantidades de prioridades de tarefas executamos com a mesma quantidade de tarefas presentes no ambiente, ou seja temos 3 prioridades presentes com 10 tarefas, 5 prioridades e 10 tarefas, e assim por diante, podemos ver os resultas obtidos na Tabela 2.

Olhando os resultados obtidos podemos perceber uma variação grande quando aumentamos a quantidade de prioridades, porém podemos ver que esse aumento é algo praticamente constante, podemos analísar melhor olhando o Gráfico 1. Como podemos ver que quando temos 10 tarefas presentes no ambiente quando ocorre a variação da quantidade de prioridades temos uma linha constante, o mesmo ocorre para 20 tarefas, porém

Tabela 2. Tempo de Execução por Número de Prioridades

Número de Prioridades	Número de Tarefas	Tempo de Execução em Segundos
3	10	24s
5	10	34s
10	10	43s
3	20	50s
5	20	48s
10	20	60s

o desempenho pelo tempo de execução é melhor quando executamos com 5 prioridades do que para 3, mas se analisarmos de forma empírica temos o mesmo tempo para ambas as quantidade de prioridade.

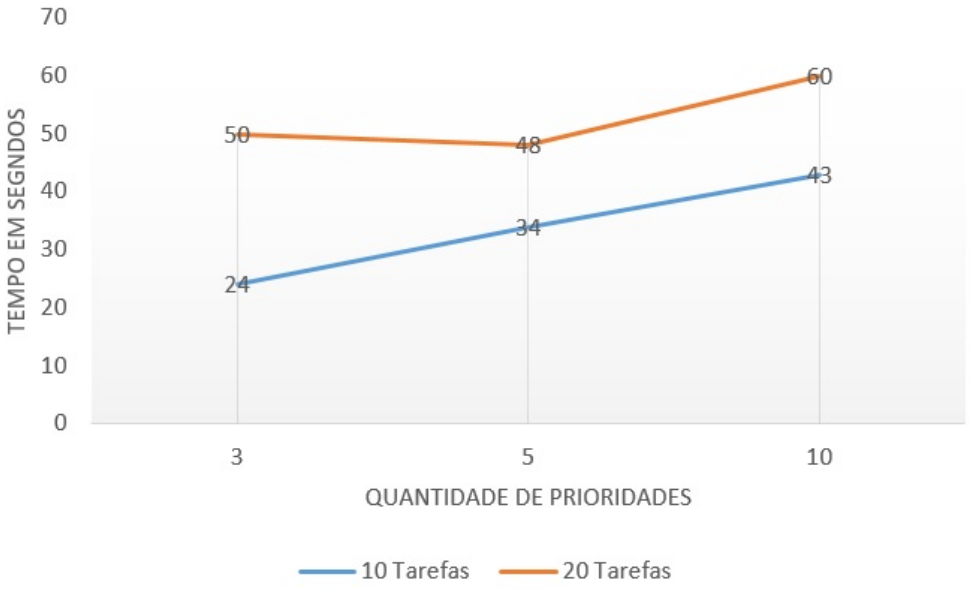

Gráfico 1. Variação de tempo conforme aumento de prioridades.

Nossa comparação dos resultados obtidos serão com os resultados de tempo obtidos no trabalho de replicação da tese do Túlio, desta forma podemos análisar o desempenho através do tempo de antes da prioridade, ou apenas 1 prioridade, com o desempenho da prioridade de tarefas implementada. Primeiro reproduzimos as mesmas condições dos dados que já tínhamos anteriormente, ou seja executamos para 10 subtarefas e 10 agentes no ambiente e apenas 1 prioridade no sistema, o tempo de execução aumentou em 5 segundos, o aumento se dá pela etapa de chamada de cara prioridade, e por ter que acessar a pasta da prioridade durante a execução, o que antes não era feito pelo sistema, podemos análisar os resultados obtidos na Tabela 3.A segunda comparação realizada é feita com 20 subtarefas, pelos mesmo motivos o tempo de execução com a prioridade é maior, mas nada tão significante.

Tabela 3. Comparação pelo tempo de execução da Replicação com a Prioridade de Tarefas

Número de Tarefas	Tempo de Execução em Segundos
10	7s
10	12s
20	8s
20	14s

5. Conclusão

Após entendermos o problema complexo que a alocação de tarefas nos trás, podemos ver que a solução apresentada para a prioridade de tarefas na alocação de tarefas não é um grande problema, os tempos comparados com o trabalho de replicação com os resultados obtidos após a inclusão da prioridade mostram que os dados são muito parecidos, porém o tempo aumenta de forma considerável quanto maior for a quantidade de prioridade de tarefas presentes no sistema, porém o resultado a curto prazo continua sendo uma ótimo solução para o problema.Esta contribuição é um grande passo para a Tese do Túlio, pois com ela abre a possibilidade de comparações com outros algoritmos, além disso realizando uma análise com o estudo proposto na replicação onde foi levantado que a grande melhora da proposta seria na área de troca de mensagens, podemos imaginar que com a prioridade de tarefas o número de troca de mensagens fica mais baixo ainda, pois cada alocação trabalha com menos tarefas, pois as mesmas são divididas de forma aleatória pela quantidade de prioridades no sistema. Assim podemos ter a certeza que este trabalho serve como uma grande contribuição, e mostra que a Tese do Tùlio está muito bem proposta e que realmente funciona em diversas variantes, inclusive na variante de prioridade nas tarefas.

Referências

[1] Jacamo project. *Disponível em:< http://jacamo.sourceforge.net/>, Acesso em*, 09, 2018.

[2] Java. *Disponível em:<https://www.java.com/pt_BR/>*, $Acessoem, 11, 2018$.

[3] Netbeans. *Disponível em:<https://netbeans.org/>, Acesso em*, 11, 2018.

[4] H. C. de Almeida Soares. Um estudo sobre o problema de alocação. *Disponível em:<https://goo.gl/gt8pAx>, Acesso em*, 11, 2018.

[5] R. H. B. DERICK PRADIÉ GARCEZ, TÚLIO LIMA BASÉGIO. Uma replicação do trabalho ”alocação dinâmica de tarefas em times de agentes heterogêneos”. *Disponível em:<https://goo.gl/3adygB>, Acesso em*, 09, 2018.

[6] D. P. GARCEZ. Integradoraii. *Disponível em:<https://github.com/derickpg/IntegradoraII>, Acesso em*, 11, 2018.

[7] A. Makhorin. Glpk (gnu linear programming kit). *<http://www.gnu.org/s/glpk/glpk.html>*, 2008.

[8] R. H. B. TÚLIO LIMA BASÉGIO. Alocação dinâmica de tarefas em times de agentes heterogêneos. *Disponível em:<https://goo.gl/uQ6hC2>, Acesso em*, 09, 2018.