RAFAEL MANCINI SANTOS

ALGORITMOS HEURÍSTICOS PARA O PROBLEMA DE ROTEAMENTO E AGENDAMENTO DE TRABALHADORES COM TAREFAS DEPENDENTES

LAVRAS – MG

2019

RAFAEL MANCINI SANTOS

ALGORITMOS HEURÍSTICOS PARA O PROBLEMA DE ROTEAMENTO E AGENDAMENTO DE TRABALHADORES COM TAREFAS DEPENDENTES

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. Dilson Lucas Pereira
Orientador

LAVRAS – MG
2019

RAFAEL MANCINI SANTOS

ALGORITMOS HEURÍSTICOS PARA O PROBLEMA DE ROTEAMENTO E AGENDAMENTO DE TRABALHADORES COM TAREFAS DEPENDENTES

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

APROVADA em 16 de Dezembro de 2019.

Prof. Dr. Dilson Lucas Pereira	UFLA
Prof. Dr. Mayron César de Oliveira Moreira	UFLA
Prof. Dr. André Vital Saúde	UFLA

Prof. Dr. Dilson Lucas Pereira
Orientador

LAVRAS – MG
2019

SUMÁRIO

Algoritmos heurísticos para o problema de roteamento e agendamento de trabalhadores com tarefas dependentes

Rafael M. Santos, Dilson L. Pereira

[1]Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 – 37200-000 – Lavras – MG – Brasil

rmancini@estudante.ufla.br, dilsonlucas@gmail.com

Abstract. *This work proposes two constructive heuristics and an adaptation of ACO (Ant Colony Optimization) for the Multiperiod Workforce Scheduling and Routing Problem with Dependent Tasks (MWSRPDT). The designed heuristics, denoted as Heurística Padrão and Heurística Tabu, were submitted to experiments in which they were used with two Ant Colony variations: MMAS and ACS. These experiments were performed using artificial instances divided in three classes. For both heuristics, the variation which gave best results was the MMAS. Futhermore, the results obtained by the algorithms stdMMAS and tabuMMAS were compared to the ones obtained by the exact model and by the ACO based algorithm defined by Pereira, Alves e Moreira (2019).The tabuMMAS has shown itself to be the most appropriate algorithm, among the proposed ones, to solve the MWSRPDT. The ACO based algorithm defined by Pereira, Alves e Moreira (2019) has shown to be the most appropriate of all algorithms presented in this work.*

Resumo. *Este trabalho propõe duas heurísticas construtivas e uma adaptação do ACO(Ant Colony Optimization) para o MWSRPDT(Multiperiod Workforce Scheduling and Routing Problem with Dependent Tasks). As heurísticas desenvolvidas, nomeadas Heurística Padrão e Heurística Tabu, foram submetidas a experimentos com duas variações do ACO: MMAS e ACS. Estes experimentos foram realizados em instâncias artificiais divididas em três classes. Para ambas as heurísticas, o uso do MMAS possibilitou a obtenção de melhores resultado. Assim, comparou-se os resultados obtidos pelos algoritmos stdMMAS e tabuMMAS com o modelo exato e com o algoritmo baseado no ACO propostos por Pereira, Alves e Moreira (2019). O tabuMMAS mostrou-se o algoritmo mais apropriado, dentre os propostos, para resolver o MWSRPDT. O algoritmo algACO mostrou-se o mais apropriado de todos algoritmos apresentados neste trabalho.*

1. Introdução

Um problema presente em sistemas de transporte e distribuição é o problema de roteamento de veículos (VRP, do inglês *Vehicle Routing Problem*), que começou a ser estudado por Dantzig e Ramser (1959), com o nome de Problema de Despacho de caminnões(TDP, do inglês *Truck Dispatching Problem*). O problema consiste em um conjunto de clientes, cada um possuindo uma demanda de produto; um conjunto de veículos, com capacidade limitada de carga de produto, responsáveis por atender a demanda dos clientes; um

depósito, do qual os veículos se originam e devem retornar ao completar suas rotas. Este problema requer o planejamento de rotas para a entrega de produtos, de tal forma que a distância percorrida por estes veículos seja a menor possível. Uma generalização do VRP é o Problema de roteamento de veículos com janelas de tempo (VRPTW, do inglês *Vehicle Routing Problem with Time Windows*) em que, apesar de compartilhar o mesmo objetivo do VRP, possui a diferença de que a entrega de produtos para um cliente deve satisfazer um período de tempo especificado pelo próprio cliente; caso um veículo chegue em um cliente antes desse período especificado pelo mesmo, deve esperar até que o período comece (KALLEHAUGE et al., 2005).

Neste trabalho, estudou-se uma classe de problemas chamada de Problema de Roteamento e Agendamento de Trabalhadores com Tarefas Dependentes (MWSRPDT, do inglês *Multiperiod Workforce Scheduling and Routing Problem with Dependent Tasks*), que possui características semelhantes ao VRPTW. O MWSRPDT contempla um horizonte de tempo composto de vários dias, sendo que, neste trabalho, considera-se o dia com uma duração de 8 horas. Clientes solicitam serviços, compostos por tarefas que podem ou não ser interdependentes. Essas tarefas são realizadas por equipes, inicialmente localizadas em um depósito ou armazém. Diferentes equipes podem realizar uma mesma tarefa em tempos diferentes – algumas equipes podem ser mais eficientes que outras em determinadas tarefas. Para cada dia do horizonte de tempo, deve ser desenvolvido um conjunto de rotas a serem percorridas por equipes para a realização de tarefas. Essas tarefas não podem ser executadas parcialmente. Não existe a exigência de que cada cliente seja atendido todos os dias e nem que todas as suas tarefas sejam executadas em um único dia, mas sim que, ao fim do horizonte de tempo, todas as tarefas do serviço requisitado tenham sido realizadas. Os clientes podem ser atendidos mais de uma vez em um dia. O planejamento desenvolvido das rotas e das tarefas deve ser tal que todos os clientes tenham sido completamente atendidos no menor número de dias possível.

O objetivo deste trabalho é contribuir com algoritmos que proporcionem soluções de qualidade para o MWSRPDT, assim como um melhor entendimento desse problema a partir da exploração de diferentes abordagens para resolvê-lo. Esses algoritmos podem, futuramente, serem utilizados em cenários reais que encaixem no modelo do MWSRPDT. Para esse objetivo, foram desenvolvidas duas heurísticas construtivas para algoritmos baseados na meta-heurística Colônia de formigas (ACO, do inglês *Ant Colony Optimization*) (DORIGO; GAMBARDELLA, 1997). Essas duas heurísticas foram nomeadas como Heurística Padrão e Heurística Tabu (Seção 2). Os algoritmos desenvolvidos baseados no ACO são o stdACS, stdMMAS, tabuACS e tabuMMAS (Seção 3). Também foram realizados experimentos computacionais com estes algoritmos (Seção 4), comparando o desempenho destes com o modelo exato e o algoritmo baseado no ACO desenvolvido por Pereira, Alves e Moreira (2019).

1.1. Definição do problema

O MWSRPDT foi modelado a partir de um grafo completo $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ em que cada vértice $v \in V$ representa um cliente; $|V| = n$. O número 0 foi escolhido para representar o vértice origem, ou depósito, do problema. Um cliente representado por um vértice v será descrito pelas letras i ou j daqui em diante. É possível ver, na Figura 1.1, um exemplo de um grafo de clientes de um problema contemplando $n = 3$ clientes, sendo esses representados pelos vértices a, b e c.

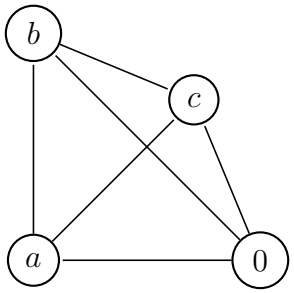

Figura 1.1. Exemplo de um grafo de clientes G

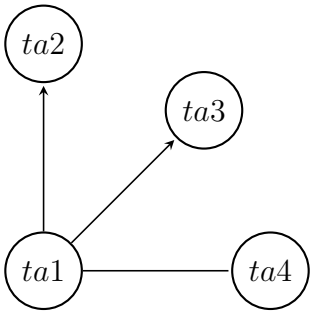

Figura 1.2. Grafo de serviço S_1

Considerando uma empresa que ofereça um conjunto de serviços ζ, tomando $S \in \zeta$ um serviço pertencente a este conjunto, temos que, no problema, cada cliente do grafo G requisita um serviço S da empresa. Cada serviço é representado como um grafo direcionado $S = (V_S, E_S)$, em que o conjunto de vértices V_S representa as tarefas que compõem o serviço S e o conjunto de arestas E_s representa as dependências entre as tarefas. Se uma aresta $(a, b) \in E_S$, então a tarefa b só poderá ser executada após a execução da tarefa a. Pode-se ver exemplos de grafos de serviço nas Figuras 1.2 e 1.3, em que as tarefas são representadas pelos vértices $ta1, ta2, ta3$ e $ta4$. Nestas Figuras, utiliza-se como exemplo dois serviços arbitrários denotados como S_1 e S_2. S_1 possui 4 tarefas, enquanto S_2 possui 3 tarefas.

As tarefas vão sendo executadas por um conjunto de equipes heterogêneas $\mathcal{E}$ em uma sequência de dias. Cada dia tem uma duração $D = 8$ horas, de forma que uma equipe $e \in \mathcal{E}$ só pode durar uma quantidade de tempo $D_e \leq 8, D_e \in \mathbb{R}_+$. Assume-se que todas

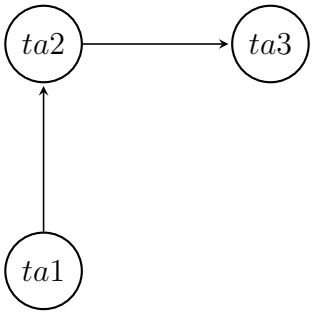

Figura 1.3. Grafo de serviço S_2

equipes utilizem o mesmo meio de transporte o tempo todo, e o tempo $d_{ij} = d_{ji} \in \mathbb{R}_+$ que uma equipe e leva pra ir de i para j é dado em horas. As equipes são heterogêneas no sentido em que equipes diferentes executam uma mesma tarefa em tempos diferentes, sendo $t^e_{ia} \in \mathbb{R}_+$ o tempo que uma equipe e leva para executar uma atividade a do cliente i.

1.2. Uma breve revisão da literatura

Os problemas classificados como Problema de Roteamento e Agendamento de Trabalhadores (WRSP, do ingles *Workforce and Scheduling Routing Problem*) surgem em diferentes tipos cenários, assim como foi mostrado pela pesquisa feita por Castillo-Salazar, Landa-Silva e Qu (2012). Exemplos de cenários são enfermeiras atendendo pacientes em suas casas (CHENG; RICH, 1998) e técnicos realizando reparos e instalações para clientes residindo em diferentes localizações (CORDEAU et al., 2010). Em Goel e Meisel (2013), temos um exemplo do cenário de manutenção em redes elétricas com o objetivo de minimizar o tempo de inatividade da rede.

Castillo-Salazar, Landa-Silva e Qu (2012) constataram que uma estratégia que tem se demonstrado útil para resolver o WRSP é o *Branch and Price*. Esta estratégia se refere ao uso da abordagem *Branch and Bound* com geração de colunas (BARNHART et al., 1998). Também descrevem características comuns entre problemas classificados como WRSP, sendo estas: períodos de tempo, modalidades de transporte, locais de começo e fim, trabalhadores com habilidades diversas, tempo para realização e relações de precedência de tarefas.

Em Cheng e Rich (1998), o problema pode ser caracterizado como um VRPTW com múltiplos depósitos, sendo estes depósitos as casas de cada uma das enfermeiras a trabalho. Também, neste problema, tem-se que cada cliente solicita apenas uma tarefa para ser realizada, que pode ser variada. Abordou-se o problema através da modelagem exata com programação inteira mista e do desenvolvimento de uma heurística baseada em algoritmo guloso aleatório e busca local.

Em Cordeau et al. (2010), foi abordado o Problema de Agendamento de Tarefas e Técnicos (TTSP, do inglês (*Technician and Task Scheduling Problem*). Esse problema pertence à classe de problemas WRSP e foi introduzido como o assunto do desafio de 2007 proposto pela *French Operational Research Society*(ROADEF) em colaboração com a *France Télécom*. Neste problema, dado um conjunto de técnicos, cada um com diferentes domínios de habilidade, o objetivo é atribuir tarefas à grupos de técnicos de forma a minimizar o custo da mão de obra, satisfazendo as prioridades de tempo de término das tarefas. Neste trabalho, experimenta-se resolver o problema com a meta-heurística ALNS (*Adaptative Large Neighborhood Search*).

No problema de manutenção em redes elétricas (GOEL; MEISEL, 2013), as tarefas realizadas pelos trabalhadores podem ser interdenpendentes; os trabalhadores são homogêneos, no sentido em que possuem habilidade para realizar todas as tarefas e os tempos de viagem e realização de tarefa são os mesmos. A abordagem utilizada para a busca de soluções combina métodos exatos para o modelo de programação inteira mista do problema com a meta-heurística LNS (*Large Neighborhood Search*).

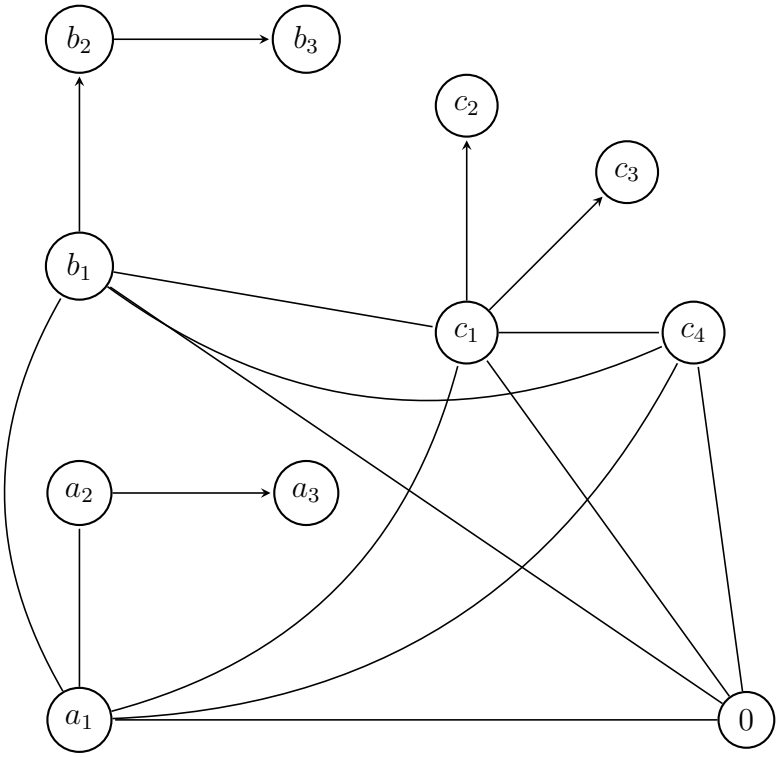

Figura 2.1. Um exemplo de um grafo estendido G'

2. Heurísticas Construtivas

2.1. Descrição geral

Começaremos por descrever as características em comum entre as duas heurísticas construtivas desenvolvidas.

Optou-se, neste trabalho, por modelar o problema como um grafo direcionado dinâmico denominado grafo estendido, dado por $\mathrm{G}' = (\mathrm{K}, \mathrm{A})$, em que $K = \{(i, a) | i \in V \text{ e } a \in V_S\}$, sendo $S \in \zeta$ o serviço requisitado pelo cliente i, e A é um conjunto dinâmico cuja natureza será explicada mais adiante. Diante disso, um par (i, a) é considerado um vértice no grafo estendido. Para simplicidade de notação, $(i, a) \equiv a_i$ e $(j, b) \equiv b_j$, representando, respectivamente, uma tarefa a do cliente i e uma tarefa b do cliente j; frequentemente ao decorrer deste trabalho, a_i será referenciada como uma tarefa ou um vértice de G'.

Considerando um cenário exemplo, com base no grafo de clientes da Figura 1.1, em que os clientes a e b solicitam o serviço S_2 (Figura 1.3) e o cliente c solicita o serviço S_1 (Figura 1.2), pode-se ver na Figura 2.1 o que seria um grafo estendido desse cenário.

As arestas do conjunto A tem a forma (a_i, b_j), sendo que percorrer uma aresta significa que a equipe já realizou a tarefa a_i, foi do cliente i ao cliente j, e realizou a tarefa b_j. O conjunto A é dinâmico e depende do conjunto de vértices já visitados durante a execução das heurísticas. Para inicialmente preservar a relação de precedência entre as tarefas temos que, para um cliente i que solicitou um serviço S, $(a_i, b_i) \in A$ se $(a, b) \in E_S$.

Na definição do problema MWSRPDT, temos que as equipes são heterogêneas,

pois gastam tempos diferentes para executar uma mesma tarefa. Assim, o peso de uma aresta (a_i, b_j) deve depender de qual equipe está utilizando-a. Diante disso, a partir da definição do grafo estendido G', define-se o peso $w(a_i, b_j, e)$ de uma aresta (a_i, b_j) percorrida por uma equipe e pela equação:

$$w(a_i, b_j, e) = \begin{cases} t^e_{jb} + d_{ij}, \text{ se } i \neq j; \\ t^e_{jb}, \text{ se } i = j. \end{cases} \tag{1}$$

Para as heurísticas desenvolvidas, são utilizadas listas de adjacência, denotadas aqui como $\delta(a_i)$ para um vértice a_i. Estas listas, assim como o conjunto A, também são dinâmicas. Como um vértice $b_j \in \delta(a_i)$ se, e somente se, a aresta $(a_i, b_j) \in A$, um vértice b_j pertence à lista de adjacência do vértice a_i se, e somente se, todas as tarefas, das quais a tarefa b_j é dependente, foram realizadas.

Durante a execução de ambas heurísticas desenvolvidas o conjunto A e a lista de adjacência $\delta(a_i)$ são atualizados na medida que os vértices de G' são visitados pelas equipes. Tomando $U \subseteq K$ como o conjunto de vértices já visitados e $\mathcal{D}_{bj}$ como o conjunto de tarefas das quais b_j é dependente, temos $(a_i, b_j) \in A$ e $b_j \in \delta(a_i)$, $\forall a_i \in K; a_i \neq b_j$, quando $\mathcal{D}_{bj} \subset U$.

A partir das listas de adjacência, são construidas listas de componentes de solução, denotadas como $\mathcal{C}$. Os componentes de solução são representados como uma tupla $c \equiv (a_i, b_j, e)$; $c \in \mathcal{C}$, a qual mostrou-se uma melhor alternativa de acordo com os testes feitos por Pereira, Alves e Moreira (2019), em que foram obtidos melhores resultados com esse tipo de componente de solução utiizando uma heurística baseada em DES (*Discrete Event Simulation*) em um algoritmo baseado no ACO.

Outro aspecto presente em ambas as heurísticas é sobre a organização das equipes. Todas as equipes partem do vértice de origem, que é representado no grafo estendido por $(0, 0)$, para construir as rotas. As equipes trabalham de forma dessincronizada, de forma que é preciso manter, para cada equipe, o dia k_e em que uma equipe e está e o horário h_e em que a equipe se encontra em relação ao dia. A ordem com que as equipes constroem suas rotas é aleatório, e cada equipe só pode escolher um componente de solução por iteração do algoritmo.

2.2. Heurística Padrão

Em cada iteração, cada equipe e escolhe um componente c de uma lista de componentes de solução disponíveis, construídos a partir da lista de adjacência, que é atualizada após cada escolha. O dia de uma equipe é finalizado quando não é possível que mais um componente seja escolhido sem que seja violado o tempo limite de trabalho diário.

É importante notar que a heurística permite que uma equipe espere quando não há escolhas possíveis, enquanto ainda há vértices para serem visitados. A espera é feita da seguinte forma: quando não há tarefas disponíveis para uma equipe e, enquanto existe pelo menos uma equipe $e' \neq e$ tal que $k_{e'} \leq k_e$, busca-se o menor horário $h_{e'}$ satisfazendo $h_{e'} > h_e$ e $k_{e'} = k_e$, para fazer com que $h_e \leftarrow h_{e'}$, de forma que o tempo de espera será igual a $h_{e'} - h_e$. Esta ação de espera se repete até que $h_e \geq h_{e'}$ para todo $e' \in \mathcal{E}$ e $k_{e'} = k_e$. No caso de $k_{e'} > k_e$ para toda equipe $e' \in \mathcal{E}$ tal que $e \neq e'$, a equipe e finaliza seu dia, pois não há possibilidade de novas atividades se tornarem disponíveis naquele

dia. A ação de espera de uma equipe é representada no algoritmo como a escolha de um componente de solução especial $\Delta = (a_i, a_i, e)$.

De forma a evitar uma preferência na escolha de tarefas, as equipes são escolhidas de uma lista de equipes que é sorteada após cada iteração

A solução é finalizada quando todos os vértices já foram visitados. Esse algoritmo foi detalhado no pseudocódigo 1.

2.3. Heurística Tabu

De forma a criar oportunidades de escapar de convergência a mínimos locais, a Heurística Tabu trabalha com a proibição do uso de todos os componentes de uma solução já aperfeiçoada na construção de novas soluções. Consegue-se, primeiro, uma solução executando o ACO utilizando o Heurística Padrão como construção de soluções com um determinado número de iterações (o número de iterações utilizado neste trabalho é especificado na seção 4.2). Após esse passo, é criada uma lista com todos os componentes pertencentes à solução gerada. Esta lista será usada como lista Tabu em uma nova construção de soluções, idêntica ao Heurística Padrão exceto pela seguinte alteração: sempre que a lista de adjacência é atualizada (em todas as iterações), certifica-se que os componentes na lista Tabu não sejam inseridos na lista de componentes disponíveis. Uma consequência desse método é o cenário em que a construção de uma solução torna-se impossível devido aos únicos componentes de solução possíveis estarem na lista Tabu. Quando isso ocorre, ignora-se a lista Tabu.

3. A meta-heurística Colônia de Formigas

Escolhemos a meta-heurística Colônia de Formigas (ACO, do inglês *Ant Colony Optimization*) (DORIGO; CARO, 1999) devido ao seu sucesso em resolver problemas reais de roteamento de veículos (RIZZOLI et al., 2007). Essa técnica de otimização baseia-se na inteligência coletiva emergente do comportamento de algumas espécies de formigas. Essas possuem um mecanismo de depósito de feromônios. Elas sentem a presença desse e tendem a escolher os caminhos com maior concentração de feromônios. Investigando esse mecanismo, em um experimento conhecido como ”experimento de dupla ponte”(BECKERS; DENEUBOURG; GOSS, 1992), em que o ninho de uma colônia de formigas Argentinas foi conectada a uma fonte de comida por duas pontes de tamanhos distintos observou-se que, apesar de inicialmente as formigas escolherem as pontes de forma aleatória, a menor ponte sempre acumulava mais feromônios do que a outra, e, assim, atraía mais formigas a ela, o que aumentava ainda mais sua concentração de feromônios. Como resultado, após um tempo, toda a colônia passa a utilizar apenas uma ponte. Dessa forma, esse mecanismo pode ser usado, também, para encontrar o menor caminho entre o ninho e a fonte de alimento, como foi observado em uma variante do experimento de dupla ponte (GOSS et al., 1989), em que o tamanho de uma ponte era consideravelmente maior do que o tamanho da outra. As formigas que escolhiam a ponte menor, faziam o caminho com maior frequência, por ser bem menor, e consequentemente depositavam mais feromônios, o que provocava a convergência da colônia a utilizar o menor caminho entre o ninho e a fonte de alimento.

Inspirado nesse modelo de comportamento de algumas espécies de formigas, a técnica de otimização ACO utiliza o conceito de ”formigas artificiais”, as quais constroem

Algorithm 1 StandartHeuristic

procedure STANDARTHEURISTIC(G') ▷ G' é o grafo estendido de um problema
Seja U o conjunto de vértices já visitados
Seja e uma equipe
Seja k_e o dia em que uma equipe e se encontra
Seja h_e o horário
Seja $\mathcal{S}$ o conjunto de componentes utilizados na solução
Seja l_e o vértice em que uma equipe e se encontra
Inicializar U e $\mathcal{S}$ como $\emptyset$
Inicializar k_e e h_e como 0 para todo $e \in \mathcal{E}$
Inicializar l_e como $(0, 0)$ para todo $e \in \mathcal{E}$
while $K \neq U$ **do**
for e na lista sorteada de equipes **do**
Seja $\delta(l_e)$ a lista atualizada de vértices adjacentes
if $\delta(l_e) = \emptyset$ **then**
if $K \neq U$ **then**
if Existe pelo menos uma equipe e' tal que $k_{e'} \leq k_e$ **then**
if Existem $h_{e'} > h_e; e' \in \mathcal{E}, e' \neq e$, e $k_{e'} = k_e$ **then**
Seja $h_{e'}$ o menor desses horários.
$h_e \leftarrow h_{e'}$
$\Delta = (l_e, l_e, e)$
$\mathcal{S}.inserir(\Delta)$
Pular execução para próxima equipe $e \in \mathcal{E}$
else
if $l_e = (0, 0)$ para todo $e \in \mathcal{E}$ **then return** $\mathcal{S}$.
else
É preciso adicionar componentes de retorno $(l_e, (0, 0), e)$ para todas as equipes tais quais $l_e \neq (0, 0)$ e fazer com que, para estas equipes, $l_e \leftarrow (0, 0)$
Seja $\mathcal{C}$ a lista de componentes construída a partir de $\delta(l_e)$
Seja c um componente vazio
if $C = \emptyset$ **then**
$c \leftarrow (l_e, (0, 0), e)$
$k_e \leftarrow k_e + 1$
else
$c \leftarrow Decisao(\mathcal{C})$ ▷ A decisão pode ser ACS ou $MMAS$(Seção 3)
$\mathcal{S}.inserir(c)$
$h_e \leftarrow h_e + w(l_e, c.a_i, e)$
$l_e \leftarrow c.a_i$ ▷ $c.a_i$ sendo a tarefa que foi realizada
if $l_e \neq (0, 0)$ **then**
$U.inserir(l_e)$

return $\mathcal{S}$

soluções para o problema de otimização em questão. Aspectos relacionados à qualidade das soluções são comunicados entre as formigas através de um mecanismo análogo ao depósito de feromônios feito por formigas reais. O AS (*Ant System*) (DORIGO et al., 1996) é o primeiro dos algoritmos com essas características, e seu modelo do cálculo da probabilidade de uma formiga escolher, a partir de um vértice de origem v_o, um vértice de destino v_d, é dado pela seguinte equação:

$$p(v_o, v_d) = \begin{cases} \dfrac{[\tau(v_o, v_d)]^\alpha \cdot [\eta(v_o, v_d)]^\beta}{\sum\limits_{v_i \in \delta(v_o)} [\tau(v_o, v_i)]^\alpha \cdot [\eta(v_o, v_i)]^\beta}, & \text{se } v_d \in \delta(v_o); \\ 0, & \text{senão,} \end{cases} \tag{2}$$

em que τ é o feromônio; $\eta = 1/w$ é o inverso do custo w da aresta (v_o, v_d); $\delta(v_o)$ é a lista dos vértices adjacentes ao vértice v_o; α e β são parâmetros que representam as significâncias dadas aos termos τ e η, respectivamente. A técnica em questão foi formalizada em uma metaheurística para seu uso na resolução de problemas de otimização combinatória. De forma mais geral, o feromônio é aplicado a cada componente de solução.

As heurísticas construtivas descritas na seção 2 são utilizadas pelo ACO de acordo com o Pseudocódigo 2. Optou-se pelas variações do ACO: ACS (*Ant Colony System*) (DORIGO; GAMBARDELLA, 1997) e MMAS (*Max-Min Ant System*) (STÜTZLE; HOOS, 2000). Como é possível notar, a variação AS não foi utilizada. Isso é motivado pela melhor perfomance que o MMAS já demonstrou sobre o AS em alguns problemas (STÜTZLE; HOOS, 2000).

Nas descrições à seguir das variações ACS e MMAS, adaptou-se as equações ao modelo do grafo estendido, de forma que uma aresta (v_o, v_d) foi trocada pelo componente de solução (a_i, b_j, e) e a lista de adjacência $\delta(v_o)$ foi trocada por $\delta(a_i)$.

Algorithm 2 Colonia de formigas

```
1: procedure ACO(G′)                ▷ Sendo G′ o grafo estendido do problema
2:     Atribuir parâmetros e inicializar feromônios.
3:     while número de iterações não foi atingido do
4:         ConstruirSolucaoDasFormigas(G′)        ▷ com Heurística Padrão ou Heurística Tabu
5:         AtualizarFeromonios()
```

3.1. *Ant Colony System*

Nesta variação, um vértice, a partir dessa regra, é escolhido de acordo com a seguinte equação:

$$b_j = \begin{cases} \arg\max\limits_{b_k \in \delta(a_i)} \{\tau(a_i, b_k, e)^\alpha \cdot \eta(a_i, b_k, e)^\beta\} & \text{se } q \leq q_o \\ b_j \text{ escolhido probabilisticamente de acordo com 4} & \text{senão,} \end{cases} \tag{3}$$

em que q é uma variável aleatória tal que $0 \leq q \leq 1$ e q_0 é um parâmetro.

Quando $q \leq q_0$, a escolha de b_j é determinística e dada direta pela equação. Se esse não for o caso, a escolha é feita de forma probabilística, e a probabilidade do componente de solução (a_i, b_j, e) ser escolhido é dado pela equação:

$$p(a_i, b_j, e) = \begin{cases} \dfrac{[\tau(a_i, b_j, e)]^\alpha \cdot [\eta(a_i, b_j, e)]^\beta}{\sum\limits_{b_k \in \delta(a_i)} [\tau(a_i, b_k, e)]^\alpha \cdot [\eta(a_i, b_k, e)]^\beta}, & \text{se } b_j \in \delta(a_i); \\ 0, & \text{senão.} \end{cases} \tag{4}$$

3.2. *Max-Min Ant System*

Nesta outra variação, o MMAS (*Max-Min Ant System*) escolhe-se o vértice de forma probabilística de acordo com a Equação 4. A partir de um vértice origem, calcula-se as probabilidades das possíveis arestas a serem utilizadas partindo desse vértice.

3.3. Atualização de feromônios

Ambas as variações da ACO implementam atualização global de feromônios, que é feita após a construção de soluções. No entanto, o ACS também implementa atualização local de feromônios, que é feita no momento de escolha de um componente de solução. Esses dois métodos de atualização serão descritos a seguir.

O objetivo da atualização de feromônios é direcionar a busca de soluções, aumentando a probabilidade das formigas optarem por caminhos mais promissores. Seja $\mathcal{M}$ o conjunto de todas as soluções geradas pela construção de soluções; seja $\mathcal{S} \in \mathcal{M}$ uma solução. O mecanismo da atualização de feromônios global, tanto do ACS quanto to MMAS, é dado pela seguinte equação:

$$\tau_c \leftarrow (1 - \rho) \cdot \tau_c + \sum_{\mathcal{S} \in \mathcal{M}, c \in \mathcal{S}} \frac{Q}{f(\mathcal{S})}, \tag{5}$$

em que ρ é um parâmetro que representa a taxa de evaporação do feromônio, $f(\mathcal{S})$ é o valor da solução $\mathcal{S}$, Q é um parâmetro. A atualização é feita para todas os componentes c pertencentes a $\mathcal{S}$. No caso da variação MMAS, existem limites superior e inferior para a quantidade de feromônios, fazendo com que a equação fique da seguinte forma:

$$\tau_c \leftarrow \Big[(1 - \rho) \cdot \tau_c + \sum_{\mathcal{S} \in \mathcal{M}, c \in \mathcal{S}} \frac{Q}{f(\mathcal{S})}\Big]_{\tau_{min}}^{\tau_{max}} \tag{6}$$

em que τ_{max} e τ_{min} são, respectivamente, os limites superior e inferior de quantidade de feromônio que um componente de solução pode ter.

Considerando o ACS, como já foi dito, existe também uma atualização local de feromônios. Essa é feita de acordo com a equação

$$\tau_c \leftarrow (1 - \phi) \cdot \tau_c + \phi \cdot \tau_0, \tag{7}$$

em que $0 < \phi < 1$ é um parâmetro.

Instâncias		StdACS		StdMMAS	
n	matv	mval	mt	mval	mt
A					
10	29.4	1.47	62.4	1.43	45.6
20	58.2	3.4	202.9	3.21	190.9
30	83.4	4.2	405.7	3.76	429.1
B					
10	29.4	2	58.5	1.74	55.1
20	58.2	3.71	199.5	3.57	209.5
30	83.4	6.45	415.5	6.03	445.3
C					
10	27	7.93	105.5	7.92	115.44
20	57	11.31	359.3	11.42	369.7
30	87	13.77	832.4	13.76	896.4

Tabela 1. Comparação do desempenho da heurística Heurística Padrão com as variações MMAS e ACS

4. Experimentos computacionais

Nesta seção, são apresentados os experimentos computacionais que foram realizados com as heurísticias desenvolvidas utilizando as variações ACS e MMAS. Primeiro, são descritas as características das instâncias utilizadas para os experimentos e, logo em seguida, são informados os valores utillizados para os parâmetros do ACO. Os resultados dos experimentos feitos com as heurísticas Heurística Padrão e Heurística Tabu, utilizando tanto o ACS quanto o MMAS, são apresentados na sequência. Esses algoritmos (heurística construtiva com o ACO) são mencionados nessa seção com os seguintes nomes: stdACS e stdMMAS para a Heurística Padrão com as variações ACS e MMAS, respectivamente; tabuACS e tabuMMAS para a Heurística Tabu com as mesmas respectivas variações. Por fim, são apresentados os resultados das heurísticas desenvolvidas utilizando a variação que melhor produziu resultados, comparando estes com os obtidos pelo modelo exato e algoritmo baseado no ACO desenvolvidos por Pereira, Alves e Moreira (2019).

4.1. Instâncias

Os experimentos foram feitos nas mesmas instâncias geradas artificialmente utilizadas por Pereira, Alves e Moreira (2019). Essas instâncias foram divididas em três tipos: A, B e C. Para cada tipo de instância, considera-se o número de clientes $n \in \{10, 20, 30\}$. Em todos os tipos de problemas, escolheu-se um número de equipes $|\mathcal{E}| = 3$.

Nas instâncias do tipo A, todas as equipes podem executar todas as tarefas. Essas são divididas entre 3 serviços S_1, S_2 e S_3 que possuem, respectivamente, 1, 3 e 5 tarefas. Esses serviços são atribuídos, aleatoriamente, a n clientes.

Instâncias		TabuACS		TabuMMAS	
n	matv	mval	mt	mval	mt
A					
10	29.4	1.47	49.5	1.4	110.9
20	58.2	3.4	230.5	3.17	379.3
30	83.4	4.2	453.9	3.72	894.7
B					
10	29.4	2	65.5	1.73	119.9
20	58.2	3.71	223.23	3.46	401.6
30	83.4	6.5	462.9	5.8	925
C					
10	27	7.92	117.6	7.92	179.4
20	57	11.32	399.5	11.41	616.7
30	87	13.77	889.6	13.56	1402.1

Tabela 2. Comparação do desempenho da heurística Heurística Tabu com as variações MMAS e ACS

As instâncias do tipo B são semelhantes às do tipo A. Apenas se diferenciam pelo fato de que algumas equipes podem ser incapazes de realizar algumas tarefas. No entanto, em cada instância do tipo B, pelo menos uma equipe e é capaz de realizar uma atividade a.

Já nas instâncias do tipo C, uma atividade a só pode ser realizada por uma equipe e. Nesse caso, considera-se apenas um serviço S_0 composto de 3 tarefas e 3 equipes disponíveis, sendo que cada equipe é capaz de realizar apenas 1 das 3 tarefas do serviço S_0.

4.2. Parâmetros do ACO

Utilizou-se os valores para os parâmetros do ACO que foram sugeridos por Pereira, Alves e Moreira (2019): para o ACS, usou-se $\alpha = 5.51$, $\beta = 5.59$, $\rho = 0.27$, $Q = 3.21$, $\tau_0 = 5.23$, $\phi = 0.27$ e $q_0 = 0.48$; para o MMAS, usou-se $\alpha = 2.57$, $\beta = 2.3$, $\rho = 0.04$, $Q = 4.67$, $\tau_0 = 5.11$, $\tau_{min} = 0.22$, $\tau_{max} = 9.57$. O ACO foi executado, em todos os testes, com 100 iterações e 100 formigas. No caso da heurística Heurística Tabu, obtém-se, antes, uma solução $\mathcal{S}_o$ através da execução do stdMMAS com os parâmetros já especificados, para depois ser executado, com 100 iterações e formigas, utilizando os componentes $c \in \mathcal{S}_o$ como lista tabu.

4.3. Resultados e discussão

Todos os algoritmos utilizados para a comparação de resultados nessa seção foram implementados em *python3*. O modelo exato desenvolvido por Pereira, Alves e Moreira (2019) é um modelo de programação linear inteira mista que foi implementado utilizando o pacote CPLEX, obtido através da iniciativa acadêmica da IBM. Os experimentos com os algoritmos desenvolvidos neste trabalho foram realizados em um notebook utilizando Linux Mint 19 Tara, com um processador de dois núcleos Intel Core i7-4500U 1.80GHz e 8GB de memória RAM. Já os experimentos relacionados ao modelo exato e ao algoritmo baseado no colônia de formigas desenvolvidos por Pereira, Alves e Moreira (2019) foram realizados em um computador utilizando o sistema operacional Ubuntu 14.04, com um processador de 8 núcles Intel Xeon 3.5Ghz e 8GB de memória RAM. Medimos a performance do computador utilizado neste trabalho através dos indicadores dados pelo software *PassMark*© (encontrado em https://www.cpubenchmark.net/), em que foi obtido um valor *CPUMark* de 1558 para o indicador *single thread*. Quanto maior o valor *CPUMark* de um computador, mais rápido ele é, sendo que um computador com o dobro desse valor consegue processar, aproximadamente, o dobro da quantidade de dados.

Os resultados dos experimentos feitos com o objetivo de comparar o desempenho das heurísticas desenvolvidas utilizando as variações MMAS e ACS do ACO são mostrados nas Tabelas 1 e 2. Para essas Tabelas, para cada número n de clientes, utilizou-se 5 instâncias para obter um desempenho médio de cada algoritmo. A média do número de tarefas dos problemas é dada por ”matv”, enquanto a média do tempo gasto por cada algoritmo, em segundos, é dada por ”mt”. A média dos números de dias obtidos pelas soluções das 5 instâncias está sob ”mval”.

Para as Tabelas 3, 4 e 5, foram escolhidos os algoritmos que obtiveram melhor desempenho, de forma que fosse possível comparar os resultados obtidos por esses, pelo modelo exato, e pelo algoritmo baseado no ACO desenvolvidos por (PEREIRA; ALVES;

Instância			stdMMAS		tabuMMAS		algACO		Modelo		
n	id	tarefas	ls	t	ls	t	ls	t	ls	li	t
10	1	23	2	38	2	82.6	2	10.7	2	2	8.7
	2	35	2	49.7	2	132.7	2	13.2	2	2	8.7
	3	31	2	48	2	119.2	2	13.4	2	2	7.0
	4	35	2	59.2	2	143	2	15.7	2	2	30.5
	5	23	1	33.3	1	67.4	1	8.4	1	1	0.5
	6	23	2	34.1	2	60.2	2	8.8	2	2	184.9
	7	29	2	56	2	101.1	2	13.6	2	2	20.5
	8	29	2	47.6	2	83.3	2	11.1	2	1	*
	9	31	2	64.4	2	110.8	2	17	2	2	16.9
	10	19	1	25.6	1	43.4	1	7.6	1	1	0.3
média		27.8	1.8	45.6	1.8	94.4	1.8	12	1.8	1.7	387.8
20	1	63	5	276.6	5	534.7	5	59.1	5	4	*
	2	63	4	192	3	384.8	3	41.9	3	3	1075.7
	3	59	4	172.1	4	341.3	4	37.7	4	4	243.0
	4	55	2	158.3	2	319.9	2	40.1	2	2	31.3
	5	51	3	155.5	3	315.6	3	36.3	3	3	224.5
	6	61	3	243	3	421.8	3	55.1	3	3	128.9
	7	55	3	202	3	343.5	3	45.9	3	3	353.7
	8	63	4	219.2	4	376	4	47.9	4	3	*
	9	43	2	125.8	2	209.8	2	31.2	2	2	11.8
	10	61	3	226.2	3	390.8	3	52.9	3	3	936.3
média		57.4	3.3	197.1	3.2	363.8	3.2	44.8	3.2	3	1054.9
30	1	79	4	441.5	3	888.3	4	96.6	4	3	*
	2	83	4	370.5	4	768.7	4	73.2	4	2.8	*
	3	79	4	350.9	4	747.3	4	66.8	4	3	*
	4	101	4	594.3	4	1241.6	4	112.8	4	4	1785.6
	5	75	6	388.5	6	827.8	5	76.2	5	3.9	*
	6	75	4	397.3	4	676.8	4	80.5	4	4	557.8
	7	101	7	638.8	7	1149.6	7	127.2	7	2.3	*
	8	99	5	567.7	5	1167.5	5	111.3	5	2	*
	9	79	5	468.2	5	910.3	5	89	5	3.1	*
	10	83	3	463	3	852.1	4	89.7	3	3	850.4
média		85.4	4.6	468.1	4.5	923	4.6	92.3	4.5	3.1	2839.3

Tabela 3. Resultados para as instâncias da classe A

Instância			stdMMAS		tabuMMAS		algACO			Modelo	
n	id	tarefas	ls	t	ls	t	ls	t	ls	li	t
10	1	23	2	39.7	2	87.4	2	9.5	2	2	3.2
	2	35	3	68.6	3	150.4	3	11.9	3	3	42.6
	3	31	2	61.1	2	135	2	13	2	2	14.3
	4	35	3	72.6	3	158.4	3	12	3	3	81.3
	5	23	1	33.2	1	68.3	1	8.8	1	1	0.4
	6	23	2	43.8	2	91.7	2	8	2	2	1.3
	7	29	2	67.3	2	131.5	2	10.9	2	2	6.8
	8	29	6	76.1	6	141.8	6	10.9	6	3.1	*
	9	31	2	81.3	2	143.9	2	17.3	2	2	4.6
	10	19	1	33.2	1	57.3	1	6.5	1	1	0.2
média		27.8	2.4	57.7	2.4	116.6	2.4	10.9	2.4	2.1	375.5
20	1	63	5	285.9	5	566.2	4	63.2	4	4	241.6
	2	63	6	233.4	6	446.3	6	43.7	6	4.6	*
	3	59	3	169.2	3	315.9	3	37.9	-	-	-
	4	55	2	181.6	2	342.7	2	35.1	2	2	29.3
	5	51	3	177.6	3	337.1	3	31.7	3	3	54.4
	6	61	4	246.4	4	523.9	4	44.3	4	3	*
	7	55	4	258.4	4	535.1	3	41.1	3	3	272
	8	63	4	253.4	4	451.2	3	42.5	3	3	726.3
	9	43	3	152.4	3	265.9	3	27.4	3	3	113.5
	10	61	7	263.1	7	525.9	7	44.1	7	4.6	*
média		58.2	4.1	222.1	4.1	431	3.8	41.5	3.8	3.3	1359.7
30	1	79	7	517.9	7	998.4	6	91	6	4.3	*
	2	83	4	341.4	4	776.2	4	70.3	4	2.6	*
	3	79	3	299.7	3	662.7	3	57.9	3	3	1382.3
	4	101	13	663.1	12	1445.6	12	117.9	12	2.7	*
	5	75	6	404.5	5	742.2	5	77.4	5	3.7	*
	6	75	6	406.6	6	816	6	75.4	6	3.8	*
	7	101	26	847.7	26	1692.8	26	112.7	-	-	-
	8	99	5	568.5	5	1161	5	108.8	5	1.7	*
	9	79	4	416.8	4	842.7	4	72.5	4	2	*
	10	83	4	414.3	4	844.9	4	78.7	4	2	*
média		85.4	5.8	448.1	5.6	921.1	5.4	83.4	5.4	2.9	3353.5

Tabela 4. Resultados para as instâncias da classe B

Instância			stdMMAS		tabuMMAS		algACO		Modelo		
n	id	tarefas	ls	t	ls	t	ls	t	ls	li	t
10	1		4	95.5	4	146	4	11	3	3	12.6
	2		10	135.6	10	212	10	12	10	5	*
	3		9	82.7	9	127.7	9	14.9	9	3.9	*
	4		10	140.7	10	212.1	10	14.4	10	8.9	*
	5	27	9	122.8	9	199.1	10	17.5	9	4.7	*
	6		3	81	3	161.9	2	12.5	2	2	2.8
	7		3	90.8	3	185.3	3	11.8	-	-	-
	8		6	94.1	6	187	5	10.1	5	4	*
	9		6	96.6	6	193.6	5	13.2	5	3.1	*
	10		2	80	2	141.5	2	16.4	2	2	311.3
média		27	6.6	103.2	6.6	175.7	6.3	13.6	6.1	4.1	2436.3
15	1		4	204.2	4	341.3	4	20.9	4	3	*
	2		15	212.3	15	409.5	15	28.6	14	6.7	*
	3		14	150.7	14	304	14	31.5	14	5.6	*
	4		6	177.4	6	335.3	5	18.2	5	3.7	*
	5	42	3	153.2	3	308.3	4	23.8	-	-	-
	6		14	261.2	14	541.6	14	15.8	14	5.8	*
	7		16	230.3	16	521.7	15	19.8	15	5.8	*
	8		15	235.8	15	484	15	21.6	14	6	*
	9		8	214.6	8	437	7	20.8	7	5.4	*
	10		5	189	5	373.4	6	20.5	5	3.7	*
média		42	10.8	208.4	10.8	416.4	10.4	22	10.2	5.1	3600

Tabela 5. Resultados para as instâncias da classe C

MOREIRA, 2019). Os algoritmos que foram escolhidos são o stdMMAS e tabuMMAS, sendo essa escolha motivada pela média menor de nº de dias das soluções obtidas por esses algoritmos. As primeiras 5 instâncias utilizadas para obter os resultados presentes nesta Tabela foram as mesmas utilizadas para obter o desempenho médio nas Tabelas 1 e 2. Cada instância é identificada por uma "id", dada por um número. Sob "tarefas"está o número total de tarefas de cada instância. O número de dias dada por uma solução de um algoritmo está sob "ls"e o tempo gasto para encontrar uma solução está sob "t", dado em segundos. Para o modelo exato, representado nas Tabelas como "Modelo", o número de dias de uma solução é dado em limites inferior e superior, que são representados nas Tabelas, respectivamente, como "li"e "ls". O tempo gasto pelo modelo em cada instância é dado sob "t", também dado em segundos. Para o algoritmo baseado no ACO desenvolvido por Pereira, Alves e Moreira (2019), representado nas Tabelas como "algACO", tem-se também o número de dias das soluções sob "ls"e o tempo gasto para encontrar essas soluções sob "t". Nas instâncias em que o modelo falhou em encontrar uma solução ótima dentro de um tempo limite de 3600 segundos, marcou-se com símbolo "*". Em uma única instância (com $n = 20$ e $id = 3$ na Tabela 4), não se obteve resultado, pois de acordo com Pereira, Alves e Moreira (2019), o modelo excedeu o limite de uso de memória durante a execução; representou-se essa ausência de solução com o símbolo -".

Na Tabela 3, nota-se que para instâncias com $n \leq 20$, o modelo, o algoritmo tabuMMAS e o algoritmo algACO conseguiram obter soluções ótimas em 17 das 20 instâncias, sendo que o stdMMAS conseguiu obter soluções ótimas em 16 das 20 instâncias. Num total de 30 instâncias, é possível observar que o algoritmo tabuMMAS obteve soluções ótimas em 21 instâncias, enquanto os algoritmos stdMMAS e algACO obtiveram em 19 instâncias. Portanto, apesar de mais lento, o tabuMMAS mostrou-se mais adequado para encontrar melhores soluções para instâncias de classe A. Nota-se, também, que para $n = 30$, o tabuMMAS obteve solução ótima em 4 das 10 instâncias, enquanto o modelo obteve solução ótima em 3 instâncias e os algoritmos stdMMAS e algACO obtiveram solução ótima em, respectivamente, 3 e 2 instâncias; considerando ainda que o tabuMMAS é mais rápido que o modelo em instâncias desta dimensão, temos que o tabuMMAS foi o algoritmo com melhor desempenho em relação à qualidade de soluções. No entanto, considerando o tempo gasto pelos algoritmos em relação à qualidade das soluções obtidas, tem-se que o algACO conseguiu um melhor balanço, pois possui o tempo de execução cerca de 10 vezes menor do que o tempo de execução do tabuMMAS.

Já na Tabela 4, vemos que o desempenho do algoritmo tabuMMAS é quase igual ao desempenho do stdMMAS (exceto na instância 4 para $n = 30$) e ligeiramente inferior ao desempenho do algACO, no quesito de qualidade de soluções obtidas. De forma similar ao que é visto na Tabela 3, os algoritmos e o modelo conseguem encontrar soluções ótimas na maioria das instâncias com $n \leq 20$. Contudo, neste caso, o modelo e o algoritmo algACO encontra soluções ótimas em 3 instâncias a mais que os algoritmos, tornando o algACO mais apropriado pra instâncias de classe B com $n \leq 20$ devido ao seu tempo de execução baixo. Para $n = 30$, o tabuMMAS e o algACO conseguiram encontrar soluções tão boas quanto as melhores encontradas pelo modelo exceto na instância 1, no caso do tabuMMAS, em que o modelo encontrou uma solução com 1 dia a menos.

Os resultados para as instâncias de classe C, expostos na Tabela 5, mostram que

essa classe de instância é significativamente mais difícil que as classes A e B, de forma que o modelo falhou em encontrar soluções ótimas na maioria das instâncias. Apenas em 3, das 20 instâncias, o modelo conseguiu encontrar soluções ótimas, sendo que em 2 dessas 3 instâncias, o algACO também conseguiu encontrar soluções ótimas. Nesta Tabela, nota-se, também, que não houve diferença na qualidade de soluções encontradas pelo tabuMMAS e stdMMAS, sendo que este último mostra-se melhor, neste caso, por ser mais rápido.

No geral, para os algoritmos desenvolvidos neste trabalho, o tabuMMAS mostrou-se com um desempenho melhor que o modelo e o stdMMAS. É, em média, mais rápido que o modelo e consegue encontrar soluções ótimas em mais instâncias que o stdMMAS, apesar de ser mais lento que este último. Contudo, o algACO mostra-se o algoritmo com melhor balanço entre qualidade de soluções e tempo de execução.

5. Conclusão

Neste artigo, descrevemos o MWSRPDT (*Multiperiod Workforce Scheduling and Routing Problem*) e propomos duas heurísticas construtivas para serem usadas com duas variações do ACO. Essas duas heurísticas construtivas, Heurística Padrão e Heurística Tabu, foram experimentadas com duas variações do ACO, o ACS e o MMAS. O algoritmo tabuMMAS, que é o ACO MMAS utilizando como heurística construtiva o Heurística Tabu, mostrou-se mais apropriado para resolver o MWSRPDT dentre os algoritmos desenvolvidos, por ter obtido soluções de qualidade em um tempo mais curto que o modelo. No entando, o algoritmo desenvolvido por Pereira, Alves e Moreira (2019) baseado no ACO foi o que obteve um melhor balanço entre qualidade das soluções obtidas e tempo de execução.

A heurística construtiva Heurística Tabu foi experimentada apenas com um tipo de lista tabu: uma lista de todos os componentes de uma solução encontrada pelo stdMMAS. É possível que essa heurística possua melhores rendimentos com outros tipos de lista tabu, como, por exemplo, uma lista com todos os componentes de apenas um dia específico de uma solução encontrada pelo stdMMAS. Futuramente, pretende-se experimentar a heurística Heurística Tabu com diferentes tipos de lista tabu, assim como experimentar diferentes heurísticas construtivas para o ACO.

Agradecimentos

Agradecemos à FAPEMIG, que financiou este trabalho através da bolsa APQ-02762-17.

Referências

BARNHART, C. et al. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, INFORMS, v. 46, n. 3, p. 316–329, 1998.

BECKERS, R.; DENEUBOURG, J.-L.; GOSS, S. Trails and u-turns in the selection of a path by the ant lasius niger. *Journal of theoretical biology*, Elsevier, v. 159, n. 4, p. 397–415, 1992.

CASTILLO-SALAZAR, A.; LANDA-SILVA, D.; QU, R. A survey of workforce scheduling and routing. 2012.

CHENG, E.; RICH, J. L. *A home health care routing and scheduling problem:* relatório técnico. [S.l.], 1998.

CORDEAU, J.-F. et al. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, Springer, v. 13, n. 4, p. 393–409, 2010.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management science*, Informs, v. 6, n. 1, p. 80–91, 1959.

DORIGO, M.; CARO, G. D. Ant colony optimization: a new meta-heuristic. In: IEEE. *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. [S.l.], 1999. v. 2, p. 1470–1477.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, IEEE, v. 1, n. 1, p. 53–66, 1997.

DORIGO, M. et al. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics*, v. 26, n. 1, p. 29–41, 1996.

GOEL, A.; MEISEL, F. Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research*, Elsevier, v. 231, n. 1, p. 210–228, 2013.

GOSS, S. et al. Self-organized shortcuts in the argentine ant. *Journal of Theoretical Biology*, p. 579–581, 1989.

KALLEHAUGE, B. et al. Vehicle routing problem with time windows. In: *Column generation*. [S.l.]: Springer, 2005. p. 67–98.

PEREIRA, D. L.; ALVES, J. C.; MOREIRA, M. C. de O. *A Multiperiod Workforce Scheduling and Routing Problem with Dependent Tasks:* relatório técnico. [S.l.], 2019.

RIZZOLI, A. E. et al. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, Springer, v. 1, n. 2, p. 135–151, 2007.

STÜTZLE, T.; HOOS, H. H. Max–min ant system. *Future Generation Computer Systems*, v. 16, n. 8, p. 889 – 914, 2000. ISSN 0167-739X. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0167739X00000431⟩.