UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL

GUSTAVO DE LUNA PINTO

META-HEURÍSTICA APLICADA AO PLANEJAMENTO DE ATIVIDADES DE LOTES DE VAGÕES EM TERMINAIS FERROVIÁRIOS

VITÓRIA
2019

GUSTAVO DE LUNA PINTO

META-HEURÍSTICA APLICADA AO PLANEJAMENTO DE ATIVIDADES DE LOTES DE VAGÕES EM TERMINAIS FERROVIÁRIOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Civil do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Mestre em Engenharia Civil, na área de concentração Transportes.

Orientador: Prof. Dr. Rodrigo de Alvarenga Rosa.

VITÓRIA
2019

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Pinto, Gustavo de Luna, 1992-
P659m Meta-heurística aplicada ao planejamento de atividades de lotes de vagões em terminais ferroviários / Gustavo de Luna Pinto. - 2019.
71 f. : il.

Orientador: Rodrigo de Alvarenga Rosa.
Dissertação (Mestrado em Engenharia Civil) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Otimização combinatória. 2. Alocação de recursos. 3. Ferrovias. 4. Ferrovias - Estações. I. Rosa, Rodrigo de Alvarenga. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 624

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

META-HEURÍSTICA APLICADA AO PLANEJAMENTO DE ATIVIDADES DE LOTES DE VAGÕES EM TERMINAIS FERROVIÁRIOS

Gustavo de Luna Pinto

Dissertação apresentada ao Curso de Mestrado em Engenharia Civil do Programa de Pós-Graduação em Engenharia Civil da Universidade Federal do Espírito, como requisito parcial para obtenção do título de Mestre em Engenharia Civil, área de Transportes.

Aprovada no dia **28 de março de 2019** por:

Prof. Dr. Rodrigo de Alvarenga Rosa
Doutor em Engenharia Eletrica
Orientador – UFES

Prof. Dr. Patricio José Moreira Pires
Doutor em Engenharia Civil
Examinador Interno – UFES

Prof. Dr. Rafael Buback Teixeira
Doutor em Engenharia de Produção
Examinador Externo - IFES Cariacica

Vitória – ES, março de 2019

RESUMO

A ferrovia é muito importante para a economia do Brasil, principalmente no transporte de grãos agrícolas e minérios. Uma das áreas de estudo da operação ferroviária é voltada para pátios e terminais ferroviários, sendo um terminal ferroviário um pátio especializado em carga e descarga de produtos. Nesses terminais, os vagões ferroviários passam metade de suas vidas úteis e realizam diversas atividades, geralmente feitas em conjuntos de vagões chamados de lotes. Uma atividade no terminal pode ter uma predecessora, ou seja, requer o término de outra atividade para ser iniciada. Além disso, cada uma pode ser executada de mais de um modo (Multi-modo), demandando diferentes recursos, que são caros e escassos, portanto, concorridos. Como os atrasos na operação podem gerar multas, busca-se minimizar o tempo total em que os lotes de vagões permanecem nos terminais (tempo de estadia), considerando a limitação de recursos do terminal. A operação nos terminais possui outras características que aumentam sua complexidade, como: tempo de setup entre atividades específicas; variação de disponibilidade de recursos ao longo do tempo (manutenção programada); e a característica de retenção, que mantém um recurso ocupado pelo lote mesmo após o término da atividade, devido à restrição de movimentação do lote imposta por trilhos e/ou outras instalações. Só foi encontrado um trabalho que considerou as características do problema, propondo um modelo matemático baseado no *Resource Constrained Project Scheduling Problem* (RCPSP). No entanto, o modelo leva muito tempo para resolver as instâncias, tornando-o inviável para a dinâmica da operação de um terminal ferroviário. Assim, foi desenvolvida nessa pesquisa uma meta-heurística baseada na meta-heurística *Ant Colony Optimization* (ACO) para o planejamento de atividades de lotes de vagões em terminais ferroviários, visando a minimização do tempo de estadia dos lotes no terminal e um tempo de processamento condizente com a necessidade de resposta sob a ótica operacional da ferrovia. Apesar de existirem muitas meta-heurísticas na literatura abordando o RCPSP, inclusive ACO, não foi encontrada uma que considerasse todas as características do problema, nem que abordasse o conceito de retenção de um recurso em cenário com múltiplos recursos por atividade. A meta-heurística gerou resultados iguais ou próximos ao ótimo, e nas instâncias maiores, obteve resultados até melhores que o modelo matemático encontrado na literatura. Além disso, levou menos de 2 minutos, um tempo de processamento aceitável do ponto de vista operacional, e muito abaixo do tempo de execução do modelo, que para as maiores instâncias durou mais de 20 horas.

Palavras-chave: Terminal ferroviário. Múltiplos modos. Tempo de setup. Retenção. Planejamento de atividades. Colônia de Formigas. Meta-heurística.

ABSTRACT

Railroad is very important for the Brazilian economy, especially in the transport of grains and ores. One area of study for railway operation is concerning rail yards and terminals, and a railway terminal is a yard specialized in products loading and unloading. In these terminals, railway wagons spend half of their service lives and carry out several activities, usually made by sets of wagons called lots. An activity in the terminal may have a predecessor, i.e., it requires the conclusion of another activity to start. In addition, it can be executed in more than one mode (Multi-mode), demanding different resources, which are expensive and scarce, therefore, shared. As delays in operation can generate fines, the goal is to minimize the total time that batches of wagons remain in the terminals (staying time), considering the limitation of terminal resources. The operation at terminals has other characteristics that increase its complexity, such as: setup time between specific activities; variation of resource availability over time (scheduled maintenance); and the blocking characteristic, which keeps a resource occupied by the lot even after the activity's end, due to the movement constraint of the batch imposed by rails and / or other facilities. Only one work was found in the literature considering the characteristics of the problem. It proposed a mathematical model based on the Resource Constrained Project Scheduling Problem (RCPSP). However, the model takes a long time to resolve instances, making it impractical for the dynamic of a rail terminal operation. Thus, a meta-heuristic based on the Ant Colony Optimization meta-heuristic (ACO) was developed in this research for the scheduling of activities of wagons' batches in rail terminals, aiming to minimize the staying time of the lots in the terminal and to do it with a processing time consistent with response time of the railway operation. Although there are many meta-heuristics in the literature addressing the RCPSP, including ACO, none were found, considering all the characteristics of the problem, neither addressing the concept of resource blocking with multiple resources per activity. The meta-heuristic generated results that were equal to or close to optimal, and for larger instances, it got results better than the mathematical model found in the literature. In addition, it took less than 2 minutes to do it, an operationally acceptable processing time, and far below the model execution time, which for the largest instances ran for more than 20 hours.

Key Words: *Railway Terminal. Multiple modes. Setup time. Blocking. Activity Schedule. Ant Colony Optimization. Meta-heuristic.*

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

De acordo com a Associação Nacional dos Transportadores Ferroviários (2019), o modo ferroviário tem aproximadamente 15% de participação na matriz de transportes brasileira e muito a crescer se comparado a outros países continentais como Estados Unidos e Austrália, onde a ferrovia transporta 43% da carga total. A ferrovia é fundamental no Brasil para levar minérios e sólidos agrícolas aos portos, com cerca de 95% e 40% de participação nos transportes dessas cargas, respectivamente.

Uma das áreas de estudo da operação ferroviária é voltada para pátios e terminais ferroviários, sendo um terminal ferroviário um pátio especializado em carga e descarga de produtos. Nesses terminais, os vagões ferroviários passam metade de suas vidas úteis e realizam diversas operações importantes para um bom desempenho da operação ferroviária, como classificação de vagões, carregamento e descarregamento, inspeção, manutenção, limpeza e formação de novos trens. Essas atividades são realizadas em conjuntos de vagões, chamados de lotes, e demandam recursos variados, escassos e dispendiosos (linha férrea, material rodante, mão de obra qualificada, equipamentos). Considerando que atrasos no processamento dos lotes podem gerar multas e impactar o desempenho da ferrovia, um bom planejamento é fundamental para reduzir o tempo total de permanência do lote no terminal, tratado como tempo de estadia (ROSA, 2016).

A complexidade do planejamento das operações nos terminais ferroviários é alta. Os lotes de vagões devem realizar diferentes atividades no terminal, respeitando relações lógicas de dependência do término de outras atividades, as predecessoras. Além disso, cada atividade pode ser executada de diversos modos, e cada modo de execução pode exigir uma configuração diferente de recursos e um tempo diferente de processamento.

Há ainda outras características que impõem restrições ao planejamento. As atividades relacionadas a um lote só podem ser realizadas após sua data de chegada ao terminal. Os recursos podem estar indisponíveis por determinado período de tempo, devido a manutenção ou treinamento, importantes para manter o nível de serviço da operação. E alguns recursos possuem tempo de setup em determinadas condições.

Uma característica do planejamento das atividades de lotes nos terminais que torna esse problema ainda mais complexo é a retenção de recursos. O recurso que possuir essa restrição só é liberado de uma atividade quando a atividade sucessora estiver liberada. Até então, determinados recursos devem permanecer ocupados mesmo com o fim da execução da atividade. Por exemplo, ao descarregar em uma moega ferroviária, um lote de vagões vai para a balança de pesagem, mas se ela não estiver liberada, o lote vai continuar ocupando a moega até que a balança seja liberada, devido à restrição da sua movimentação imposta pelos trilhos e pelo lote que ocupa a balança (PIMENTA, 2018).

Todas essas restrições tornam os recursos escassos e disputados pelas atividades dos lotes, portanto, a utilização desses recursos deve ser otimizada para a obtenção de um

melhor planejamento, que minimize o tempo de estadia dos trens no terminal ferroviário.

Só foi encontrado um trabalho na literatura que tratou todas essas características do planejamento de atividades de lotes de vagões em terminais ferroviários. Pimenta (2018) abordou o problema com uma variante do *Resource Contrained Scheduling Problem* (RCPSP), utilizando um modelo matemático.

Sabe-se que o *Resource Constrained Project Scheduling Problem* (RCPSP) é um problema *NP-hard* (BLAZEWICZ, LENSTRA e KAN, 1983), ou seja, um problema complexo. No trabalho de Pimenta (2018) o *solver* CPLEX executou o modelo matemático, encontrando soluções ótimas em instâncias com até 9 lotes de vagões em no máximo 8.915 segundos (aprox. 2,5 horas). Nas instâncias maiores, de 12 e 15 lotes de vagões, o CPLEX rodou por 79.200 segundos (22 horas) e não conseguiu achar a solução ótima. A maior instância (15 lotes) obteve GAP de 17,94%, indicando que a solução encontrada pode ser até 17,94% maior (pior) que a solução ótima.

Então, o modelo leva muito tempo para resolver as instâncias, tornando-o inviável para a dinâmica da operação de um terminal ferroviário. Assim, foi desenvolvida nessa pesquisa uma meta-heurística baseada na *Ant Colony Optimization* (ACO) para o planejamento de atividades de lotes de vagões em terminais ferroviários, visando a minimização do tempo de estadia dos lotes no terminal e considerando múltiplos modos de execução, manutenção programada, tempos de setup e retenção de recursos. Muitos trabalhos desenvolveram meta-heurísticas abordando o RCPSP e suas variantes, porém não foi encontrada nenhuma que abordou o conceito de retenção de um recurso, considerando múltiplos recursos por atividade.

Essa pesquisa foi suportada pela Fundação de Amparo à Pesquisa do Espírito Santo (FAPES) e Vale S/A, processo FAPES 75528452/2016, do edital FAPES Nº 01/2015 - COOPERAÇÃO FAPES/VALE.

1.1 Objetivo

1.1.1 Objetivo Geral

Desenvolver uma meta-heurística baseada na *Ant Colony Optimization* (ACO) para o planejamento de atividades de lotes de vagões em terminais ferroviários, visando a minimização do tempo de estadia do lote no terminal, demandando um tempo de processamento aceitável operacionalmente e considerando atividades com múltiplas predecessoras, múltiplos modos de execução, manutenção programada de recursos, tempos de *setup* e retenção de recursos.

1.1.2 Objetivos específicos

- Fazer uma análise de sensibilidade dos parâmetros da meta-heurística.

- Analisar as características da meta-heurística proposta e seu mecanismo de busca por soluções.
- Comparar os resultados da meta-heurística com os resultados do modelo matemático proposto por Pimenta (2018).

1.2 Justificativa

O atraso na liberação de lotes de terminais ferroviários acarreta em multas e perda de eficiência da ferrovia, portanto, é preciso um bom planejamento das operações de carga e descarga realizadas por esses lotes nos terminais. Essa tarefa se torna complexa ao considerar todas as restrições envolvidas, fazendo necessária a utilização de ferramentas para a tomada de decisão.

Há muitos trabalhos tratando planejamento de atividades, ou seja, em RCPSP e suas variantes, porém poucos aplicados ao problema de planejamento de atividades de lotes de vagões em terminais ferroviários. Pimenta (2018) desenvolveu o modelo matemático que mais se aproximou à realidade desse planejamento. Entretanto, o modelo matemático levou até 2,5 horas para resolver uma instância de 9 lotes e rodou por 22 horas para instâncias de 12 e 15 lotes, encontrando GAP de até 17,94%. Esses tempos não são aceitáveis para a dinâmica da operação de um terminal ferroviário. Nesses casos, meta-heurísticas são indicadas para conseguir bons resultados em tempos menores.

Visto que Pimenta (2018) considera múltiplos modos, tempos de *setup*, manutenção programada e retenção de recursos, não foram encontradas meta-heurísticas que trataram desse mesmo conjunto de características. Assim, essa pesquisa se justifica por propor uma meta-heurística baseada na *Ant Colony Optimization* (ACO) que considere todas as características tratadas por Pimenta (2018), visando gerar bons planejamentos em tempos viáveis para a realidade operacional dos terminais ferroviários.

1.3 Estrutura do Trabalho

O trabalho está organizado em seis capítulos, e nesse primeiro, Introdução, são apresentados os objetivos, a motivação, a aplicação e a estrutura do documento de pesquisa.

No segundo capítulo, Referencial Teórico, são apresentados todos os conceitos importantes para o entendimento do trabalho e também a revisão bibliográfica.

No terceiro capítulo, Método de Pesquisa, é apresentado o método de pesquisa, as etapas da metodologia, a descrição do problema e as instâncias utilizadas.

O quarto capítulo, Meta-Heurística Proposta, mostra a meta-heurística proposta e detalha suas funções e funcionamento.

No quinto capítulo, Resultados e Análises, são apresentadas a calibração dos parâmetros da meta-heurística, os resultados obtidos comparados ao modelo matemático da literatura e a análise desses resultados.

Por último, o sexto capítulo, Conclusões, apresenta as considerações finais e algumas sugestões de trabalhos futuros.

2. REFERENCIAL TEÓRICO

Nesse capítulo, os conceitos importantes para o entendimento do trabalho são revisados. Primeiro, são apresentados o RCPSP e suas variantes, e depois, o modelo matemático de Pimenta (2018). Então, uma breve descrição dos terminais ferroviários é dada. Também é realizada uma revisão bibliográfica do planejamento de atividades em terminais ferroviários e modelos e meta-heurísticas aplicadas ao RCPSP e suas variantes. Por fim, será explicado o funcionamento da meta-heurística *Ant Colony Optimization* (ACO) e sua variação.

2.1 Resource Constrained Project Scheduling Problem (RCPSP) e Variantes

O RCPSP é um problema de planejamento de projeto com restrições de recursos e uma função objetivo que é, normalmente, relacionada a tempo ou custo. Esse problema possui muitas variações e é considerado *NP-hard* (BLAZEWICZ, LENSTRA e KAN, 1983), ou seja, de grande complexidade. As características mais relevantes para o planejamento de atividades de lotes de vagões em pátios e terminais ferroviários são apresentadas a seguir.

A definição do *Resource Constrained Scheduling Problem* a seguir é baseada em Abdolshah (2014). O projeto consiste em uma determinada quantidade de atividades, cada uma demandando certo número de recursos e com um tempo de duração. Esses recursos podem ser renováveis ou não renováveis. O primeiro tipo fica disponível ao longo de todo o período, ou seja, não é consumido, apenas ocupado, como mão de obra. O segundo tipo é consumido no momento de sua utilização e tem quantidade finita, como capital, por exemplo.

Durante o processamento de uma atividade, os recursos renováveis utilizados são ocupados, ficando indisponíveis para serem utilizados em outras. Além disso, uma vez iniciada a atividade, não poderá ser interrompida, portanto, não é possível desocupar recursos, aloca-los a outra atividade e depois ocupa-los novamente com a que foi interrompida. As durações das atividades são consideradas determinísticas e são conhecidas antes da realização do planejamento.

Há também relações de precedência entre as atividades, as que possuem predecessoras só podem ser realizadas após a execução de todas as suas atividades predecessoras. Com essa restrição e considerando que os recursos são limitados pela sua disponibilidade de tempo, o problema de planejar um projeto se torna uma tarefa difícil.

A função objetivo mais utilizada na literatura é a minimização do *makespan*, ou seja, sequenciar as atividades de maneira que a duração total do projeto seja a menor possível e que a solução seja viável, respeitando as relações de precedência e a disponibilidade dos recursos. A solução final indicará as datas de início de cada atividade e quais recursos utilizarão. A seguir serão explicadas algumas variantes do RCPSP, com base em Hartmann e Briskorn (2010).

A data de liberação de uma atividade é a menor data em que é possível iniciar essa atividade e o prazo é a última data possível de término da atividade. Então uma atividade não pode ser programada fora desse intervalo. No entanto, dependendo do problema o prazo pode ser ultrapassado, acarretando em uma penalidade na função objetivo.

Uma das classificações do RCPSP é de acordo com o modo de execução. Pode ter modo único, indicando que uma atividade só pode ser executada de uma forma, ou pode ter múltiplos modos, indicando que há diferentes maneiras de realizar uma atividade (Multi-modo). Os modos de execução de uma atividade possuem as mesmas predecessoras, porém demandam diferentes configurações de recursos, ou seja, tipos ou quantidades diferentes de recursos. Então, um modo de uma atividade pode ser viável em determinado tempo e outro modo da mesma atividade não, caso só os recursos demandados pelo primeiro estejam disponíveis. Além disso, modos diferentes podem acarretar em diferentes tempos de processamento para uma mesma atividade.

Ao contrário do RCPSP padrão, a variante com *preemption* permite que uma atividade possa ser interrompida em dado momento, liberando os recursos alocados, e depois ser retomada de onde parou, ocupando novamente os recursos. Outra característica que o problema pode ter, ou não, é o tempo de *setup*, que seria o tempo necessário para a preparação de um recurso até que possa ser utilizado para a realização da atividade. Esses tempos podem variar de acordo com a atividade realizada anteriormente pelo recurso, portanto, uma atividade pode precisar de tempo de *setup* quando realizada após determinada atividade e não precisar após outras.

O RCPSP padrão possui apenas recursos renováveis, ou seja, eles não são consumidos, apenas ficam indisponíveis durante períodos de tempo em que são acionados por uma atividade. Já no RCPSP com recursos não renováveis, os recursos são consumidos, então se toda a quantidade disponível de um recurso não renovável for usada e a solução ainda demandar mais, essa solução será inviável. É possível também que o problema considere tanto os recursos renováveis, como os não renováveis.

A classificação em relação a projetos define que o RCPSP pode ter um único projeto sendo processado, ou múltiplos projetos sendo processados ao mesmo tempo. A abordagem de múltiplos projetos ao mesmo tempo é importante principalmente nos casos em que os projetos dividem recursos, criando uma concorrência pela utilização deles.

A meta-heurística proposta nessa pesquisa trata o RCPSP com data de liberação de atividades, porém sem prazos, com múltiplos modos, sem *preemption*, com tempos de *setup*, somente com recursos renováveis e com múltiplos projetos.

2.2 Terminal Ferroviário

A caracterização de terminal ferroviário apresentada foi adaptada de Rosa (2016). O terminal ferroviário é um pátio especializado em carga e descarga de produtos, onde um pátio ferroviário é uma área plana com um conjunto de vias para fins ligados à operação ferroviária. As áreas e equipamentos especializados no carregamento e descarregamento de produtos variam dependendo do tipo de carga, que pode ser carga a granel, carga geral e contêiner.

A carga a granel é uma carga não unitizada, não embalada, como minério de ferro ou grãos agrícolas. As instalações para o carregamento desse tipo de carga compreendem: praia do terminal, muros de carregamento e silos de carregamento.

A praia do terminal é a forma mais econômica, uma área plana paralela à linha de trem, onde os equipamentos vão operar, carregando os vagões, como as pás mecânicas, por exemplo. O muro de carregamento é similar, porém, as pás mecânicas operam em cima de uma área aterrada, suportada por um muro, que fica na altura dos vagões, facilitando o carregamento da carga. Os silos de carregamento são estruturas que armazenam uma grande quantidade de carga, levada a eles por meio de correias transportadoras. O trilho passa por baixo do silo, portanto, o vagão é carregado através de aberturas no silo, que liberam quantidades de carga padronizadas e distribuídas.

O descarregamento da carga a granel pode ser feito também pela praia do terminal. Outra forma seria por viradores de vagões, que travam o vagão (vagão gôndola) e giram-no 180° para que a carga caia por gravidade em correias. As instalações de moegas ferroviárias e de elevados funcionam de maneira similar, vagões com comportas laterais ou inferiores passam por cima de estruturas vazadas. As comportas dos vagões são abertas e a carga vai para galpões ou correias.

Os terminais para carga geral variam bastante, dependendo do tipo de carga operada. As operações podem ser feitas por empilhadeiras de garfo, empilhadeiras de *clamp*, pórticos, que são estruturas sobre trilhos ou pneus, e pontes rolantes, para operação dentro de armazéns.

Para terminais de contêineres, podem ser usadas empilhadeiras *reach stacker*, que levanta o contêiner por cima, ou *top lift*, que levanta por baixo, além do transtêiner, uma estrutura semelhante ao pórtico, porém especializada no transporte de contêineres.

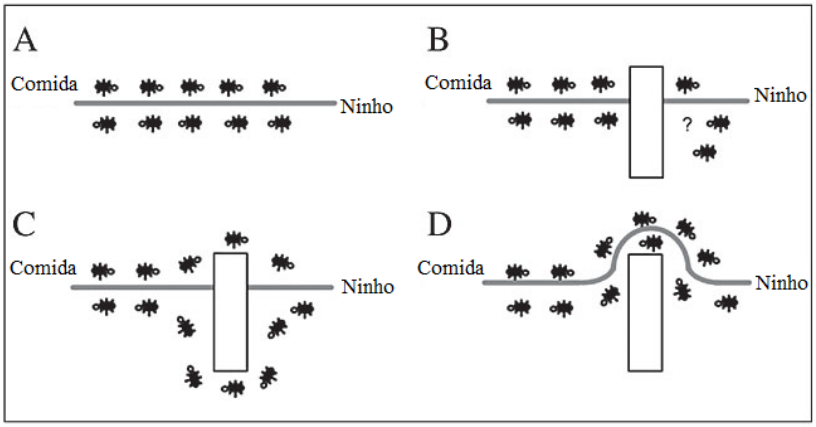
2.3 Ant Colony Optimization (ACO)

Nesse subcapítulo será apresentada uma visão geral da meta-heurística *Ant Colony Optimization* (ACO). No Capítulo 4, será detalhado o funcionamento da meta-heurística proposta baseada na ACO.

A ACO é uma meta-heurística construtiva inspirada na inteligência coletiva das formigas, que depositam feromônios por onde passam para indicar caminhos melhores para outras formigas da colônia (DORIGO e STÜTZLE, 2004). Na Figura 1, esse

processo é ilustrado, primeiro (Figura 1-A), o caminho mais curto para a fonte de comida é uma reta, então um caminho de feromônio é formado e as formigas o seguem. Posteriormente, um obstáculo é colocado no meio do caminho entre ninho e comida (Figura 1-B) e as formigas passam a seguir por dois caminhos diferentes encontrados (Figura 1-C). Cada formiga deixa um rastro de feromônio, e quanto melhor (menor) for o caminho percorrido por ela, maior será a quantidade de feromônio. Portanto, o caminho mais curto naturalmente receberá cada vez mais feromônios e as formigas que escolhiam aleatoriamente entre os caminhos no início, agora seguem pelo que possui mais feromônios e é o mais curto, Figura 1-D (PERRETTO e LOPES, 2005).

Figura 1 - Busca das formigas pelo melhor caminho.



Fonte: Adaptado de Perretto e Lopes (2005).

A conceituação e as equações do algoritmo *Ant Colony Optimization* foram retiradas e adaptadas de Perretto e Lopes (2005) e Li e Zhang (2013). Na meta-heurística ACO, a escolha do caminho pela formiga é probabilística. A equação (1) calcula a probabilidade p_{ij} de a formiga percorrer o arco (i,j), onde o nó i é sua posição atual e o nó j pertence ao conjunto Ω dos nós que ainda não foram escolhidos pela formiga. No cálculo, leva-se em consideração a quantidade de feromônios τ_{ij} no arco (i,j) e uma informação heurística η_{ij} que varia de acordo com o problema, representando normalmente a distância, o custo, ou o tempo de percorrer o arco (i,j).

As variáveis α e β representam as influências relativas do feromônio e da informação heurística, respectivamente. Então, quando α for zero, o feromônio não influenciará na escolha do nó, que será baseada apenas na proximidade deles (ou na informação heurística específica do problema), ou seja, será um algoritmo guloso, que busca mínimos locais. Já quando β for zero, a escolha será baseada na quantidade de feromônio em cada arco, portanto, a tendência é de continuar a fazer rotas que já foram percorridas, sem testar novas possibilidades. Então, deve haver um balanço entre as duas variáveis de forma a tornar a meta-heurística mais eficiente, evitando mínimos locais.

$$\begin{cases} p_{ij} = \dfrac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{h \in \Omega}\ [\tau_{ih}]^{\alpha}.[\eta_{ih}]^{\beta}} & \text{Se } j \in \Omega \\ 0 & \text{Caso contrário.} \end{cases} \quad (1)$$

Esse processo de escolha acontece para todos os nós de uma formiga até que sua solução seja construída, e o mesmo ocorre para as outras formigas, construindo suas próprias soluções. Quando um determinado número de formigas acaba de gerar soluções, um ciclo é finalizado e as quantidades de feromônios nos arcos são atualizadas com o que foi depositado pelas formigas do ciclo. A trilha de feromônios resultante irá influenciar o ciclo seguinte de formigas, criando uma memória coletiva, e esse processo vai acontecer até chegar a um número de ciclos estipulado como limite.

Existem diferentes estratégias para a atualização da quantidade de feromônios τ_{ij}, como o *Elitist Ant System*, o *Max-Min Ant System* e o *Rank-based Ant System* (DORIGO e STÜTZLE, 2004). Nessa pesquisa será utilizada a estratégia *Elitist-Rank Based Strategy* (AS_{rank}), proposta por Bullnheimer *et al.* (1997).

Após cada ciclo no AS_{rank}, as soluções geradas pelas formigas desse ciclo são ordenadas por função objetivo (FO) em um *ranking*, onde o peso da contribuição de cada formiga depende de sua posição nesse *ranking*. Além disso, a quantidade de formigas do ciclo a contribuir pode ser menor que o total de formigas por ciclo, evitando que soluções ruins afetem os feromônios do algoritmo. Nessa estratégia, além das melhores formigas do ciclo, a melhor solução encontrada até o momento (solução global) também é considerada na atualização dos feromônios.

O cálculo da atualização da quantidade de feromônio τ_{ij} no arco (i, j) é calculado de acordo com a equação (2). Primeiro, ocorre a evaporação da quantidade anterior de feromônio no arco por meio da taxa de evaporação ρ. Isso é importante para que o feromônio depositado por soluções ruins deixe de influenciar, aos poucos, a meta-heurística, permanecendo a influência somente nos trechos que continuam recebendo feromônio ao longo do algoritmo. Depois da evaporação, são adicionadas as quantidades de feromônios das melhores formigas do ciclo ($\Delta\tau_{ij}$) e da solução global ($\Delta\tau_{ij}^{*}$).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \quad (2)$$

Considerando o peso σ para a solução global como o maior peso dado a uma solução, temos que o número de formigas do ciclo que irão contribuir (λ) é igual a $\sigma - 1$ e que o peso da formiga de posição μ no *ranking* é $\sigma - \mu$. Então, a melhor formiga do ciclo terá sua posição μ igual a 1 e sua influência será $\sigma - 1$, já a pior formiga do ciclo terá sua posição μ no *ranking* igual a λ e seu peso igual a 1 ($\lambda = \sigma - 1$).

A equação (3) mostra o cálculo da quantidade de feromônio adicionada pelas λ melhores formigas do ciclo. O peso de contribuição da formiga é multiplicado pela constante Q e dividido pela função objetivo da formiga f (FO^f), onde Q é a quantidade de feromônio deixada por uma formiga durante todo o seu caminho. Então, quanto menor o caminho (FO^f), maior a concentração de feromônio ao longo dele, portanto, maior a influência. Uma formiga não vai adicionar feromônios em um arco se não o percorreu.

$$\begin{cases} \Delta\tau_{ij} = \sum_{f=1}^{\lambda} (\sigma - \mu).\, Q/FO^f & \text{Se a formiga } f \text{ percorreu o arco } (i,j) \\ 0 & \text{Caso contrário.} \end{cases} \quad (3)$$

A equação (4) mostra o cálculo da quantidade de feromônio adicionada pela formiga f^* que gerou a solução global de função objetivo FO^*. O peso σ é multiplicado pela constante Q e dividido pela função objetivo FO^*. Caso a formiga global não tenha percorrido um arco, ela não vai adicionar feromônios.

$$\begin{cases} \Delta\tau_{ij}^* = \sigma.\, Q/FO^* & \text{Se a formiga } f^* \text{ percorreu o arco } (i,j) \\ 0 & \text{Caso contrário.} \end{cases} \quad (4)$$

2.4 Revisão Bibliográfica

Essa pesquisa tem por objetivo desenvolver uma meta-heurística para o problema de Pimenta (2018), que tratou do *Multi-mode Resource Constrained Project Scheduling Problem* (MRCPSP) com manutenção programada, tempos de *setup* e retenção de recursos, e propôs um modelo matemático que encontrava soluções ótimas, por meio do CPLEX, em instâncias com até 9 lotes de vagões. Então, essa revisão terá como foco, os artigos que propuseram meta-heurísticas para o problema RCPSP e suas variantes, sobretudo, com as características semelhantes às de Pimenta (2018).

A Tabela 1 mostra os artigos encontrados na literatura que são mais relevantes para essa pesquisa e as meta-heurísticas que foram utilizadas por eles para realizar o planejamento de projetos. Percebe-se que os mais utilizados foram a ACO e o GA. Dentre os artigos, apenas Sabino *et al*. (2010) não abordou o RCPSP e suas variantes, mas sim o problema *Switch Engine Scheduling*. O artigo foi incluído pela proximidade de seu tema com o planejamento de operações em terminais, mesmo que com foco somente em locomotivas de manobra, e por propor uma meta-heurística baseada na *Ant Colony Optimization*, igual será feito nessa pesquisa.

Tabela 1 - Resumo de Artigos e das Meta-heurísticas utilizadas para o planejamento de projetos

Autores	Simulated Annealing (SA)	Ant Colony Optimization (ACO)	Algoritmo Genético (GA)	Tabu Search (TS)	Regras de Prioridade (RP)	Meta-heurística híbrida GA-VNS (GA-VNS)	Algoritmo Evolucionário (EA)	Imperialist Competitive Algorithm (ICA)	Branch and Bound (BB)
Merkle *et al.* (2002)		X							
Chiang *et al.* (2008)		X							
Van Peteghem e Vanhoucke (2010)			X						
Sabino *et al.* (2010)		X							
Chen *et al.* (2010)		X							
Barrios *et al.* (2011)			X						
Rahimi *et al.* (2013)	X						X	X	
Li e Zhang (2013)		X							
Chen e Zhang (2013)		X							
Afshar-Nadjafi e Majlesi (2014)			X						
Tavana *et al.* (2014)							X		
Cheng *et al.* (2015)					X				X
Beşikci *et al.* (2015)			X						
Wang *et al.* (2017)					X				
Tritschler *et al.* (2017)						X			
Ning *et al.* (2017)	X			X					
Al-Shihabi e Aldurgam (2017)		X							
Meta-heurística proposta nessa dissertação		X							

Merkle *et al.* (2002) propôs a meta-heurística Ant Colony Optimization (ACO) para o RCPSP. A ACO proposta atualiza a trilha de feromônios com a melhor solução do ciclo e a melhor solução encontrada até o momento, sendo que essa melhor solução é esquecida em períodos regulares para evitar convergências rápidas de soluções. O algoritmo também considera uma variação dos parâmetros de influência da informação heurística e dos feromônios ao longo de sua execução. Foram realizadas comparações com um *Simulated Annealing* e um Algoritmo Genético levantados na literatura, onde a meta-heurística proposta foi superior.

Chiang *et al.* (2008) propôs uma ACO para o MRCPSP, ou seja, considerou múltiplos modos de execução. A meta-heurística estabelece probabilidades a partir da informação heurística e da quantidade de feromônio de cada atividade ser executada em cada modo.

A informação heurística divide-se em duas partes, com um parâmetro calibrando cada parte, a primeira refere-se ao consumo relativo de recursos e a segunda à duração relativa das atividades. Como também considera recursos não renováveis, algumas soluções não são viáveis devido à falta desses recursos. Então, é proposta uma adaptação dinâmica dos parâmetros para aumentar a chance de construir soluções viáveis, ora os parâmetros visam melhorar a utilização dos recursos, ora visam reduzir a duração total do projeto.

Van Peteghem e Vanhoucke (2010) utilizaram um algoritmo genético com *preemption* para o MRCPSP. Duas populações foram utilizadas nesse algoritmo, uma para a geração da programação *foward* e uma para *backward*, ou seja, da data de início em diante, ou da data final para trás. Além disso, depois de gerada a solução, há um procedimento para verificar a duração de todos os modos das atividades sequenciadas e escolher a de menor duração quando essa escolha não tornar a solução inviável, devido a indisponibilidade de recursos. Esse procedimento não é feito para atividades que foram interrompidas e retomadas. Os resultados foram comparados com algoritmos existentes na literatura, provando sua eficiência.

Sabino *et al.* (2010) utilizaram a meta-heurística *Ant Colony Optimization* (ACO) para programar as locomotivas de pátios ferroviários, utilizadas para manobrar lotes de vagões. O algoritmo visou minimizar o custo total das manobras, buscando encontrar resultados satisfatórios em tempos pré-definidos e obedecendo as políticas operacionais do pátio. Não foram considerados múltiplos recursos para o trabalho.

Chen *et al.* (2010) tratou o *Multi-mode Resource Constrained Project Scheduling Problem with Discounted Cash Flows* (MRCPSPDCF). Uma ACO foi desenvolvida para o problema e oito informações heurísticas diferentes foram testadas, considerando diferentes questões, como tempo, custo ou recursos. A que apresentou melhores resultados foi uma heurística híbrida considerando relações de precedência, utilização de recursos, tempo e custo, fazendo um *trade-off* para combinar todas as abordagens. A meta-heurística foi comparada com dois Algoritmos Genéticos, um *Simulated Annealing* e um *Tabu Search*, obtendo resultados melhores.

Barrios *et al.* (2011) propuseram um algoritmo genético de duas fases (cada uma com diferentes procedimentos) para o MRCPSP (múltiplos modos) com atrasos máximos das atividades pré-definidos. O algoritmo inclui um procedimento para melhorar a solução, varrendo as atividades e verificando se é possível alterar modos de execução de forma a reduzir a duração da atividade, sem influenciar outras.

Rahimi *et al.* (2013) abordaram a versão de *Mode Identity* do RCPSP, uma variação do multi-modo onde algumas atividades do projeto são interdependentes, formando subgrupos, e as atividades de um subgrupo precisam ser executadas com o mesmo modo. Foram criadas três meta-heurísticas para o problema, um *Simulated Annealing*, um algoritmo evolucionário e um *Imperialist Competitive Algorithm*, e as soluções geradas são processadas por um modulo de aprendizado, gerando melhores resultados.

Li e Zhang (2013) utilizaram a meta-heurística ACO para o MRCPSP. O artigo também trabalha com recursos renováveis e não renováveis, mas sem múltiplos projetos. O algoritmo utiliza a *Elitist Rank Estrategy* para a atualização dos feromônios, além disso, trabalha com dois níveis de feromônios e informações heurísticas, um para as atividades e outro para os modos de execução. Primeiro, a atividade seguinte é selecionada, com base no modo de execução da ultima atividade definida, e depois seus modos de execução são avaliados para escolha, portanto, só os dados de um dos modos são considerados na seleção da atividade. Ou seja, as atividades e seus modos de execuções são escolhidos em etapas diferentes, ao contrário da meta-heurística proposta por essa pesquisa, que define os dois simultaneamente. Devido aos recursos não renováveis, a viabilidade da solução só pode ser verificada após a geração completa dessa solução, a medida adotada para penalizar inviabilidades é de permitir apenas a evaporação do feromônio nessas soluções, e não o incremento. O artigo também realizou uma calibração dos parâmetros, utilizando, por fim, valores variáveis ao longo do algoritmo para as influências relativas do feromônio e do valor heurístico.

Chen e Zhang (2013) desenvolveram uma ACO para o problema *Software Project Scheduling and Staffing with an Event-Based Scheduler*, portanto lida apenas com recursos humanos (renováveis) e considera *preemption*, ou seja, permite a interrupção de atividades para posterior retomada. Um algoritmo *Event Based Scheduler* (EBS) é utilizado para gerar o cronograma. O EBS aumenta a eficiência do algoritmo, ajustando a alocação de recursos apenas para instantes de tempo definidos como eventos, onde um evento é definido quando o projeto é iniciado, quando uma atividade se encerra, liberando recursos e quando empregados entram no projeto ou saem.

Afshar-Nadjafi e Majlesi (2014) propuseram um algoritmo genético para o RCPSP com tempos de *setup* e *preemption*. Os tempos de *setup* dependem da atividade, sendo os mesmos no início e na retomada da atividade após interrupção. Os parâmetros são calibrados por meio de um método estatístico e a meta-heurística é testada com sucesso em 100 instâncias de teste.

Tavana *et al.* (2014) propuseram um algoritmo evolucionário para o *Discrete time-cost-quality trade-off problem* com *preemption*. O algoritmo trata simultaneamente dos conflitos de objetivos (tempo, custo e qualidade) e, na parte de preemption, foi considerado: tempo mínimo antes da interrupção, máximo número de interrupções por atividade e tempo máximo entre a interrupção e a retomada da atividade. Os resultados foram comparados com um eficiente modelo matemático.

Cheng *et al.* (2015) exploraram a diferença entre *preemption* e *activity splitting*, ou divisão de atividades. Com preemption, uma atividade que já foi iniciada poderia continuar a ser processada, mas foi interrompida. Já na divisão de atividades, a atividade foi interrompida por motivos mais críticos e não pode ser processada, por exemplo, devido à indisponibilidade de recursos no período. Um algoritmo foi desenvolvido e aplicado para três casos, o MRCPSP sem divisão de atividades, com divisão de

atividades e sem preemption e com preemption. A influência da interrupção de atividades na solução foi analisada.

Beşikci *et al.* (2015) abordaram o problema de múltiplos projetos transformando em diferentes problemas de projeto único e resolvendo cada um separadamente. Primeiro é feita uma etapa de alocação de recursos aos projetos, criando recursos dedicados e depois os projetos são otimizados por meio de um algoritmo genético.

Wang *et al.* (2017) investigou o desempenho de regras de prioridade para a versão estocástica do RCMPSP (múltiplos projetos). A duração das atividades do problema é uma variável estocástica e duas novas medidas de robustez foram criadas para analisar o *trade-off* entre qualidade e robustez das soluções.

Tritschler *et al.* (2017) desenvolveram uma meta-heurística híbrida com algoritmo genético e *Variable Neighbourhood Search* para o RCPSP *with flexible resource profiles*, onde a quantidade de recursos alocados a uma atividade pode variar ao longo do tempo. Não foram considerados múltiplos modos.

Ning *et al.* (2017) investigaram o *Cash Flow Balanced Project Scheduling Problem* com durações de atividades estocásticas. Para minimizar a diferença entre entradas e saídas de caixa, uma programação robusta é gerada por um *Simulated Annealing* (SA) e por um *Tabu Search* (TS), onde o SA se provou mais eficiente para cenários de maior escala. O SA usa três movimentos para geração de soluções vizinhas, um movimento que troca o modo de execução da atividade, um que altera o tempo extra da atividade, ou seja, uma folga para atender a variações na duração da atividade e um que troca a data de início da atividade.

O *Finance Based Scheduling Problem* (FBSP) foi tratado por Al-Shihabi e Aldurgam (2017) e consiste em planejar as atividades sem exceder uma linha de crédito imposta, não considerando múltiplos modos de execução. Três meta-heurísticas *Max-Min Ant System* (MMAS) foram desenvolvidas e comparadas, cada uma possuindo informações heurísticas diferentes, onde a melhor foi a que considerou o número de sucessoras da atividade, portanto, o número de possíveis sequencias que a atividade teria.

A Tabela 2 mostra algumas classificações do problema para cada artigo descrito acima. A coluna (1) mostra os autores dos artigos e coluna (2) os métodos de otimização utilizados. A coluna (3) indica se o problema tratado foi de múltiplos modos (Multi-modo) e a coluna (4) indica se considerava múltiplos projetos. A coluna (5) vai mostrar se o artigo considerou recursos renováveis (R), recursos não-renováveis (NR), ou os dois juntos. A coluna (6) indica se a abordagem feita foi estocástica. A (7) indicará se o problema possui datas de liberação para iniciar as atividades, prazos para termina-las ou os dois juntos. A coluna (8) mostra se o problema admite *preemption*, ou seja, interrupção de atividades. A (9) indica se a disponibilidade, ou capacidade total, de recursos sofre variações ao longo do tempo. A coluna (10) indica se os tempos de *setup* são considerados e, por fim, a (11) indica se a característica de retenção é tratada no

problema. A última linha da tabela corresponde às características da meta-heurística proposta nessa pesquisa.

Tabela 2 - Características dos problemas tratados

Autores (1)	Método de otimização (2)	Múltiplos modos (3)	Múltiplos Projetos (4)	Tipos de Recursos (5)	Estocás-tico (6)	Data de Liberação/ prazo (7)	*Preemption (8)*	Disp. variável de recursos (9)	Tempos de *setup (10)*	Retenção (11)
Merkle *et al.* (2002)	ACO			R						
Chiang *et al.* (2008)	ACO	X		R e NR						
Van Peteghem e Vanhoucke (2010)	GA	X		R			X			
Sabino *et al.* (2010)	ACO			R						
Chen *et al.* (2010)	ACO	X		R e NR						
Barrios *et al.* (2011)	GA	X				Prazo				
Rahimi *et al.* (2013)	SA; EA; ICA	X		R						
Li e Zhang (2013)	ACO	X		R e NR						
Chen e Zhang (2013)	ACO			R			X	X		
Afshar-Nadjafi e Majlesi (2014)	GA			R			X		X	
Tavana *et al.* (2014)	EA			R e NR			X			
Cheng *et al.* (2015)	RP; BB	X		R e NR		Liberação e prazo	X	X		
Beşikci *et al.* (2015)	GA		X	R e NR						
Wang *et al.* (2017)	RP; BB		X	R	X					
Tritschler *et al.* (2017)	GA-VNS			R						
Ning *et al.* (2017)	SA; TS	X		R e NR	X					
Al-Shihabi e Aldurgam (2017)	ACO			R e NR						
Meta-heurística proposta nessa dissertação	ACO	X	X	R		Liberação		X	X	X

Analisando a Tabela 2, é possível perceber que poucos artigos consideraram múltiplos projetos, datas de liberação ou prazo de atividades, disponibilidade variável de recursos e tempos de *setup*, e nenhum tratou a retenção de recursos. A meta-heurística proposta na presente pesquisa considera múltiplos projetos (cada lote de vagões é um projeto), datas de liberação, que são as datas de chegadas dos lotes, disponibilidade variável de recursos, que é a manutenção programada de recursos, tempos de *setup* entre atividades e a característica de retenção de recursos. Não foi encontrada nenhuma meta-heurística na literatura que tratasse esse conjunto de características.

3. MÉTODO DE PESQUISA

Nesse capítulo é apresentada a classificação da metodologia de pesquisa utilizada quanto à natureza, abordagem, objetivos e procedimentos técnicos. Então, as etapas cumpridas da metodologia de pesquisa proposta são apresentadas, além do estudo do problema, utilizado para avaliar a meta-heurística proposta. Por fim são detalhadas as instâncias utilizadas e seu levantamento.

3.1 Etapas do Método de Pesquisa

Para atingir o propósito desse trabalho, foi utilizada a metodologia de pesquisa descrita a seguir. A primeira etapa foi o estudo do problema, o planejamento de atividades dos lotes de vagões em terminais ferroviários. Pimenta (2018) tratou do mesmo problema com um modelo matemático, portanto, o estudo desse trabalho permitiu o entendimento das características específicas desse tema para posterior tratamento na meta-heurística proposta.

Ainda nessa etapa, foi realizado um extenso levantamento de trabalhos que utilizaram meta-heurísticas para tratar o *Resource Constrained Project Scheduling Problem* (RCPSP) e suas variações, visto que Pimenta (2018) modelou o problema baseado nessa abordagem. Uma importante linha de pesquisa visou trabalhos que consideraram a característica de múltiplos modos de execução (multi-modo). Esse levantamento foi essencial para entender como associar o problema a uma meta-heurística, quais meta-heurísticas tinham sido mais utilizadas e como trataram a característica de multi-modo.

A segunda etapa foi o desenvolvimento da meta-heurística proposta para o RCPSP com multi-modo, tempo de *setup*, manutenção programada (indisponibilidade de recursos) e retenção de recursos. Seu desenvolvimento foi baseado no algoritmo *Ant Colony Optimization* e a linguagem de programação utilizada foi C. Seu funcionamento é detalhado no Capítulo 4.

Na terceira etapa, as instâncias utilizadas no trabalho de Pimenta (2018) foram compiladas e a estrutura de dados foi ajustada para a utilizada na meta-heurística. Com essas instâncias, os parâmetros necessários para a meta-heurística foram calibrados, visando obter melhores resultados com menor esforço computacional possível.

Na última etapa, a meta-heurística já com parâmetros calibrados foi utilizada para resolver as instâncias. A partir dos resultados foi possível analisar a qualidade das soluções obtidas, por meio da comparação com os resultados de Pimenta (2018), além disso, foi possível verificar a estabilidade do algoritmo e o tempo de processamento que exige.

3.2 Problema Estudado

As informações do problema estudado são baseadas em Pimenta (2018), já que essa pesquisa propõe uma meta-heurística para o mesmo problema tratado por ele, que

aplicou seu modelo matemático ao terminal ferroviário da Estrada de Ferro Vitória a Minas (EFVM) que fica dentro do Complexo de Tubarão (Vitória-ES).

Os lotes de vagões têm horários programados para chegar ao terminal ferroviário. O tipo, a quantidade e a sequência das atividades realizadas no terminal podem variar para cada lote, como acontecem com lotes de vagões de minério de ferro, carvão mineral, produtos siderúrgicos, fertilizantes e grãos agrícolas, que são as principais cargas operadas pela EFVM.

O terminal possui recursos limitados para realizar as operações nos lotes de vagões e a disponibilidade varia com o tempo, dependendo da ocupação desses recursos para atender a demandas de atividades. Os tipos de recursos à disposição do terminal e suas quantidades são apresentados na Tabela 3.

Tabela 3 - Recursos à disposição do Terminal

Tipos de recurso	Quantidades
Manobreiro	4
Locomotiva	4
Local de Manobra A (linhas)	3
Moega 1	1
Moega 2	1
Estrutura Limpeza	2
Estrutura Fertilizantes	1
Pá carregadeira	3
Local de Manobra B (linhas)	3

Fonte: Pimenta (2018), p. 38.

O manobreiro é o profissional responsável pelas manobras dos veículos ferroviários, além do preenchimento de registros burocráticos com informações das atividades realizadas. A locomotiva do terminal é uma locomotiva de manobra, ou seja, não sai do terminal, apenas movimenta os vagões lá dentro. O recurso Locomotiva irá se referir à locomotiva de manobra e ao maquinista que a conduz.

O Local de Manobra A e o Local de Manobra B são os locais onde acontecem as manobras de triagem. Os tempos de operação podem variar dependendo do local onde a atividade for realizada, devido à distância entre as atividades. Cada um dos locais possui quatro linhas de manobra. A Moega 1 e a Moega 2 são instalações utilizadas para o descarregamento de lotes de vagões. São representadas por tipos de recursos diferentes, por possuírem diferentes capacidades. O recurso Estrutura Limpeza possui capacidade para processar dois lotes de vagões ao mesmo tempo e faz a limpeza do lote de vagões em determinados casos em que a carga a ser transportada é diferente da anterior, como no caso do lote que vai começar a carregar fertilizante.

O carregamento de fertilizante é realizado no recurso Estrutura Fertilizantes e o recurso Pá carregadeira pode ser utilizado para levar os vagões da Estrutura Limpeza para a

Estrutura Fertilizantes, devido a uma adaptação feita no recurso. Além disso, a Pá carregadeira também é utilizada no carregamento de fertilizante nos lotes de vagões.

Esses recursos são utilizados em seis operações principais que são realizadas no terminal: recepção; descarga; triagem; manobra de limpeza; limpeza e carregamento de fertilizante; e entrega do lote.

A atividade de recepção consiste na recepção dos lotes que chegam da ferrovia ao terminal e no estacionamento deles em linhas disponíveis. A atividade de descarga é a movimentação do lote de vagões da linha em que estava para uma das moegas, o descarregamento dos vagões e também a inspeção dos lotes pela equipe de manutenção para identificar vagões que precisam de reparos.

Na atividade de triagem, o lote vai da moega em que foi descarregado para um dos locais de manobra para realizar, por meio de manobras, a triagem de vagões do lote que precisarão de reparos. Além disso, os vagões que estavam sendo reparados na oficina são incluídos no lote para completar a quantidade de vagões no lote estabelecida. Após a descarga dos vagões, A atividade manobra de limpeza leva os lotes para o local onde ocorre a limpeza dos resíduos das cargas, para posterior carregamento de fertilizante.

Na atividade limpeza e carregamento de fertilizante é realizada a limpeza, a movimentação do lote do local de limpeza para o local de carregamento de fertilizante. Ao fim do carregamento, a atividade entrega do lote consiste na retirada do lote da área do pátio de carregamento e sua destinação para entrega em linha disponível, deixando o lote preparado para seguir viagem.

É possível que haja múltiplos modos de execução por atividade, implicando que uma mesma atividade possa ser realizada com diferentes configurações de recursos (tipo e quantidade de recursos) e tempos de processamento (durações). Apesar da diferença entre os modos de execução de uma atividade, suas predecessoras são as mesmas, portanto, ou todos os modos podem ser realizados, ou nenhum pode, desconsiderando a disponibilidade de recursos.

Entre os modos de execução de uma atividade e os modos de outra pode haver tempos de *setup*, ou seja, preparações de equipamentos para outro tipo de processamento. Então, é possível que duas atividades tenham tempo de *setup*, ou não, dependendo dos modos de execução utilizados. No terminal tratado, não há exclusividade na descarga de produtos, ou seja, milho, farelo de soja ou soja podem ser descarregados na Moega 1 ou na Moega 2. Portanto, quando o tipo de produto descarregado em uma mesma moega muda, é necessário um tempo para limpeza e preparação, evitando contaminação de produtos. Para esse controle, a Moega 1 é utilizada no modo 1 da atividade de descarga e a Moega 2 é utilizada no modo 2. Os tempos de descarga entre tipos de produtos são apresentados na Tabela 4.

Tabela 4 - *Setup* da Moega entre lotes de produtos

Produto	Milho	Soja	Farelo de Soja
Milho	0	1	4
Soja	1	0	4
Farelo de Soja	1	1	0

Um tipo de recurso possui, ou não, a característica de retenção, que é ficar ocupado mesmo que a atividade já tenha sido finalizada, devido à ocupação de um recurso da atividade sucessora. Essa retenção ocorre, principalmente, quando esses recursos são instalações ligadas por uma linha férrea sem desvios e ambos estão ocupados por lotes de vagões. Por fim, alguns recursos podem receber manutenções programadas por determinado período de tempo, tornando-os indisponível nesse período para utilização nas atividades.

O principal objetivo da programação dos lotes de vagões é minimizar seus tempos de estadia no terminal ferroviário, pois o atraso na liberação dos trens pode gerar multas. Então, o planejamento das operações dos lotes de vagões no terminal é feito em posse do horário programado de chegada de cada lote, de informações dos recursos disponíveis e das atividades, das manutenções pré-programadas e dos tempos de *setup* entre modos de execução das atividades.

3.3 Instâncias

Essa pesquisa utiliza instâncias que foram propostas e executadas por Pimenta (2018), portanto, o detalhamento das instâncias é feito abaixo com base em seu trabalho. Essa prática permite a avaliação dos resultados da meta-heurística proposta quando comparados com um método exato, pois o problema tratado é o mesmo.

Foram utilizadas 13 instâncias e suas características são apresentadas na Tabela 5. As Instâncias 1 a 5, Grupo A, foram criadas para teste, adicionando características do problema a cada instância, visando avaliar a contribuição de cada uma para o resultado final e para a complexidade de resolução do problema. A Instância 1 não possui nenhuma característica específica do problema, ou seja, é um *Resource Constrained Scheduling Problem* (RCPSP). A Instância 5 é a primeira a possuir todas as características, que são adicionadas na seguinte ordem: múltiplos modos de execução; tempos de setup; indisponibilidade de recursos; e retenção de recursos. Todas possuem 9 lotes.

Nas Instâncias 6 a 9, do Grupo B, são testados diferentes períodos de indisponibilidade de recursos. Para isso, as outras características das instâncias são iguais, ou seja, possuem a mesma quantidade e tipos de lotes, horários de chegada, múltiplos modos, retenção e tempo de *setup*.

As Instâncias 10 e 11, Grupo C, representam cenários maiores, com 12 e 15 lotes respectivamente, e têm o objetivo de testar a eficiência do modelo matemático e algoritmo. Por fim, as Instâncias 12 (7 lotes) e 13 (9 lotes), Grupo D, representam cenários reais, possuindo todas as características específicas do problema.

Tabela 5 - Características das Instâncias

Grupo	Instância	N° de Lotes	Múltiplos Modos	Tempo de Setup	Indisp. de Recursos	Retenção
A	1	9				
	2	9	X			
	3	9	X	X		
	4	9	X	X	X	
	5	9	X	X	X	X
B	6	9	X	X	X	X
	7	9	X	X		X
	8	9	X	X	X	X
	9	9	X	X	X	X
C	10	12	X	X		X
	11	15	X	X	X	X
D	12	7	X	X	X	X
	13	9	X	X	X	X

A Tabela 6 mostra os tipos de lotes e indisponibilidade de recursos por instância. No grupo A, por exemplo, todos as instâncias possuem a mesma composição de lotes, 3 de soja, 3 e milho e 3 de farelo de soja. A indisponibilidade de recursos é adicionada na Instância 4 e 5, com uma locomotiva a menos do tempo 1 ao tempo 10 e com a Moega1 indisponível do tempo 11 ao 30. Essa mesma leitura da tabela pode ser feita para as demais instâncias. Para mais informações a respeito das instâncias, ver Pimenta (2018).

Tabela 6 - Tipos de lotes e indisponibilidades por Instância

Grupo	Instância	Lotes	Indisponibilidade
A	1	3 Soja; 3 Milho; 3 Farelo de soja.	
	2		
	3		
	4		Indisp. Locomotiva 1t-10t Indisp. Moega1 11t-30t
	5		Indisp. Locomotiva 1t-10t Indisp. Moega1 11t-30t
B	6	4 Soja; 2 Farelo de soja; 3 Milho	Indisp. 2 Manob. 21t a 30t Indisp. Moega2 16t a 20t
	7		Sem indisponibilidade
	8		Indisp. 2 Manob. 31t a 40t Indisp. Moega2 16t a 20t
	9		Indisp. 2 Manob. 31t a 40t Indisp. Moega2 31t a 35t
C	10	6 Soja; 3 Farelo de soja; 3 Milho	Sem indisponibilidade
	11	6 Soja; 6 Farelo de soja; 3 Milho	Indisp. 2 Local de Manobra A 15t-28t Indisp. Estrutura de fertilizantes 31t-40t
D	12	3 Soja; 4 Milho	Indisp. Estrutura Fertilizantes 18t-23t Indisp. Estrutura Limpeza 18t-23t
	13	6 Soja; 2 Milho; 1 Farelo de soja	Indisp. Estrutura Fertilizantes 18t-23t Indisp. Estrutura Limpeza 18t-23t

4. META-HEURÍSTICA PROPOSTA

Essa pesquisa propõe uma meta-heurística baseada na *Ant Colony Optimization* (ACO) para o problema abordado por Pimenta (2018), descrito no Capítulo 2. A meta-heurística proposta é baseada na *Ant Colony Optimization* e aborda o problema *Multi-mode Resource Constrained Multi-project Scheduling Problem* (MRCMPSP) aplicado ao planejamento de operações de lotes de vagões em terminais ferroviários. Cada formiga percorre os nós para gerar uma solução, e nessa meta-heurística, cada nó corresponde a um modo de execução de uma determinada atividade, representado como uma seleção {atividade, modo de execução}. Portanto, modo e atividade são escolhidos ao mesmo tempo, diferentemente do MRCPSP de Li e Zhang (2013), em que a atividade é o nó e depois que é escolhida, define-se o seu modo de execução.

Então, a formiga escolhe seleções, calculando para cada uma, a data de início mais cedo em que os recursos necessários estarão disponíveis pelo período de tempo demandado. Com isso, as seleções não são escolhidas necessariamente na ordem cronológica em que serão realizadas, a ordem das seleções e a disponibilidade dos recursos são controlados por outros mecanismos de controle na meta-heurística, que serão apresentados nesse capítulo.

A formiga não pode escolher todas as seleções, somente as que são elegíveis, ou seja, são viáveis de serem realizadas. Há três casos em que uma seleção não é considerada elegível. O primeiro caso é quando uma atividade predecessora ainda não foi realizada. O segundo caso é quando a seleção já foi escolhida pela formiga. Por último, considerando que uma seleção escolhida é um modo de execução de determinada atividade, as seleções que correspondem aos outros modos de execução dessa mesma atividade, e que eram elegíveis até o momento, se tornam inelegíveis. Portanto, a formiga não percorrerá todas as seleções do problema, pois não importa o modo que a atividade foi feita, e sim se foi feita ou não.

Na Figura 2 é ilustrada a viagem de uma formiga. Para oito seleções e 4 atividades diferentes, a formiga primeiro realizou a seleção 4, que é a atividade dois no modo dois {A2,M2}, depois a seleção {A3,M1}, a {A4,M2} e por fim a seleção {A1,M2}, percorrendo quatro nós para quatro atividades. Mesmo que a Seleção {A1,M2} esteja por último, é possível que tenha sido realizada no início, antes de outras atividades. Isso seria possível se a seleção não tiver atividades predecessoras, se seu lote já estivesse no terminal e se houvesse disponibilidade de recursos no momento. Então, a ordem das atividades não pode ser visualizada somente com essa representação.

Figura 2 - Rota de uma formiga

Fonte: Próprio autor.

Para implementar a meta-heurística com as características citadas anteriormente, uma série de estruturas, variáveis e parâmetros foram criados, e os mais importantes para o entendimento dessa dissertação serão apresentados e explicados. Depois, a visão geral da meta-heurística proposta é apresentada e, por fim, as principais funções são detalhadas.

4.1 Estruturas, Variáveis e Parâmetros da Meta-heurística Proposta

Nesse subcapítulo, são definidos os principais parâmetros, estruturas e variáveis da meta-heurística proposta. Posteriormente, as estruturas e suas variáveis são detalhadas.

Os dados que definem o tamanho dos principais vetores e matrizes são descritos a seguir:

$maxAtiv$	Número máximo de atividades
$maxPred$	Número máximo de predecessoras de uma atividade
$maxModos$	Número máximo de modos de execução de uma atividade
$maxTpRec$	Número máximo de tipos de recursos
$maxLotes$	Número máximo de lotes de vagões
$maxSelecoes$	Número máximo de seleções
$maxHorizonte$	Horizonte máximo de tempo
$maxFormigas$	Número máximo de formigas
$maxCiclos$	Número máximo de ciclos

As variáveis importantes para o funcionamento da meta-heurística são descritas a seguir:

$\tau[i][j]$	Quantidade de feromônio no arco (i,j), onde i e j variam de 0 a $maxSelecoes$;
$\Delta\tau[i][j]$	Quantidade de feromônio que será adicionada no arco (i,j) por

	todas as formigas do último ciclo completado, onde i e j variam de 0 a $maxSelecoes$;
$\Delta\tau^*[i][j]$	Quantidade de feromônio que será adicionada no arco (i, j) pela formiga global, onde i e j variam de 0 a $maxSelecoes$;
$comp[k]$	Assume valor 1 se a atividade k foi realizada e 0 caso contrário, onde k varia de 1 a $maxAtiv$;
$ultimaSel$	Número que identifica a última seleção que a formiga atual escolheu, ou 0 caso não tenha iniciado a solução;
ρ	Taxa de evaporação de feromônios;
Q	Quantidade total de feromônio deixada por uma formiga durante todo o seu caminho;
$\mu[f]$	Para cada formiga f, onde f varia de 1 a $maxFormigas$, esse vetor indica a posição de f em ordem crescente de função objetivo (ex: valor para segunda melhor função objetivo é 2).
$tempoSetup[l][l']$	Tempo de *setup* entre um lote l e outro l', ambos variando entre 1 e $maxLotes$

A seguir serão apresentadas as estruturas e suas variáveis:

A estrutura $Atividade$ (Figura 3) representa uma atividade de um lote de vagões e é composta por sete variáveis. As variáveis $lote$, $numPred$ e $numModos$ representam, respectivamente, o identificador do lote que realiza a atividade e o número de predecessoras e de modos de execução da atividade. A $duracao[m]$ é um vetor contendo a duração das seleções, cada posição do seu vetor contém a duração de um modo de execução da mesma atividade.

Figura 3 - Estrutura *Atividade*.

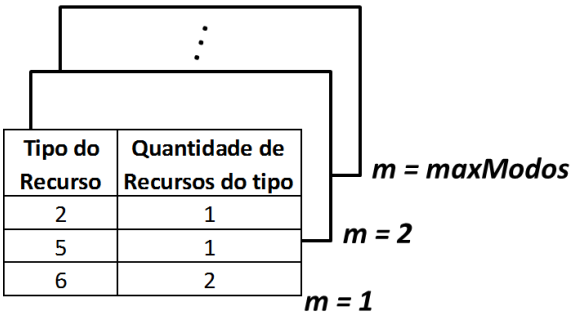
$Atividade\{$	
$int\ lote;$	
$int\ numPred;$	
$int\ numModos;$	
$int\ duracao[m];$	m varia de 1 a $maxModos$;
$int\ predecessoras[g];$	g varia de 1 a $maxPred$;
$int\ numTpRecPorMd[m];$	m varia de 1 a $maxModos$;
$int\ tpRecPorMd[m][r][C];$	m varia de 1 a $maxModos$, r varia de 1 a maxTpRec e C assume (coluna) 1 ou (coluna) 2.
$\};$	

O vetor $predecessoras[g]$ mostra o número que identifica cada atividade predecessora. Portanto, uma atividade sem predecessoras teria um vetor de zeros e uma atividade com duas predecessoras teria um vetor em que as duas primeiras posições seriam os números das atividades predecessoras e as outras posições seriam zero.

O vetor $numTpRecPorMd[m]$ indica o número de tipos de recursos utilizados por modo de execução, já que cada modo de uma atividade pode demandar diferentes configurações de recursos. A matriz $tpRecPorMd[m][r][C]$ informa a quantidade de

recursos de cada tipo utilizada pelo modo. A Figura 4 mostra, que o modo de execução $m = 1$ da atividade demanda 1 recurso do tipo 2, 1 recurso do tipo 5 e 2 recursos do tipo 6. Assim, o $tpRecPorMd[m][r][1]$ indica o tipo de recurso na posição r do modo m, e o $tpRecPorMd[m][r][2]$ indica a quantidade necessária desse recurso. Com base na estrutura $Atividade$, foi declarado o vetor de atividades $ativ[k]$ (k variando de 1 a $maxAtiv$), que representa a lista de atividades de todos os lotes de vagões.

Figura 4 - Exemplo da matriz $tpRecPorMd[m][r][C]$



Fonte: Próprio autor.

A estrutura $Lote$ (Figura 5) representa um lote de vagões que está programado para operar no terminal e possui três variáveis. A variável $Chegada$ representa o tempo em que o lote chega ao terminal. É importante notar que é do tipo inteira, pois o tempo é discretizado na meta-heurística proposta. A $numAtivRota$ é o número total de atividades programadas para o lote em questão e o vetor $ativRota[k]$ contém os números que identificam cada atividade a ser feita pelo lote. Por exemplo, um lote com seis atividades programadas em sua rota no terminal, terá as seis primeiras posições de seu vetor preenchidas com os números correspondentes às atividades do lote e o resto das posições serão zeros. Então, a última posição, diferente de zero, do vetor indicará a última atividade a ser feita pelo lote de vagões. Dessa estrutura, foi declarado o vetor $lote[l]$ (l variando de 1 a $maxLotes$), que corresponde à lista de lotes de vagões que estão programados para chegarem ao terminal.

Figura 5 - Estrutura *Lote*.

```
Lote{
    int Chegada;
    int numAtivRota;
    int ativRota[k];        k varia de 1 a maxAtiv
};
```

A estrutura $TipoRecurso$ (Figura 6) corresponde a um tipo de recurso do terminal e é composta por duas variáveis. A $recPorTp$ mostra a quantidade de recursos disponíveis do tipo de recurso em questão e $blocking$ é uma variável que assume o valor de 1, caso o tipo de recurso tenha a característica de retenção, e 0 caso contrário. Com base nessa

estrutura, foi declarado o vetor $tpRec[r]$ (r variando de 1 a $maxTpRec$) que representa todos os tipos de recursos do terminal.

Figura 6 - Estrutura *TipoRecurso*.

```
TipoRecurso{
    int recPorTp;
    int blocking;
};
```

A estrutura $Elegiveis$ (Figura 7), declarada como $eleg$, auxilia na escolha das seleções. Uma seleção é elegível quando todas as suas predecessoras foram realizadas ou quando não possuir predecessoras. Sabendo que as predecessoras dependem da atividade e não do modo de execução, se as predecessoras de uma atividade foram concluídas, então todas as seleções compostas por modos dessa mesma atividade são seleções elegíveis. Logo, o conjunto de elegíveis é um subconjunto do conjunto de seleções, e mesmo tendo tamanho máximo $maxSelecao$, e dificilmente todas as seleções serão elegíveis por algumas possuírem predecessoras, portanto, os conjuntos de elegíveis e de seleções terão tamanhos diferentes.

Figura 7 - Estrutura *Elegiveis*.

```
Elegiveis{
    int eAtiv[e];          e varia de 1 a maxSelecoes;
    int eModo[e];          e varia de 1 a maxSelecoes;
    int η[e];              e varia de 1 a maxSelecoes;
    float prob[e];         e varia de 1 a maxSelecoes;
    int seq[e];            e varia de 1 a maxSelecoes;
    int eIniTempo[e];      e varia de 1 a maxSelecoes;
    int eFimTempo[e];      e varia de 1 a maxSelecoes;
    int eTemSetup[e];      e varia de 1 a maxSelecoes;
};
```

O conjunto de elegíveis é redefinido sempre que for preciso escolher uma seleção {atividade, modo de execução}, ou seja, quando a formiga precisar escolher para qual nó seguir. Isso é importante para garantir que a formiga só tenha opções de nós (seleções) para os quais ela realmente possa ir (seleções elegíveis).

Sete vetores com tamanho máximo igual ao número máximo de seleções compõem $eleg$, ou seja, para pegar todas as informações de uma determinada seleção elegível é preciso acessar a mesma posição em cada vetor, já que essa posição está identificando essa elegível. Os vetores $eAtiv[e]$ e $eModo[e]$ armazenam, respectivamente, o número identificador da atividade e do modo de execução para cada seleção elegível. O vetor $\eta[e]$ recebe a informação heurística de cada elegível e o $prob[e]$ recebe a probabilidade que cada nó (seleção) tem de ser escolhido pela formiga.

Para calcular a probabilidade de ir para uma elegível, considera-se também a quantidade de feromônio $\tau[i][j]$, onde i é o nó em que a formiga se encontra e j é a seleção de destino. Essa matriz de feromônio não é recalculada para todos os nós, somente no fim dos ciclos, portanto, é uma matriz quadrada com cada dimensão igual ao número total de seleções do problema. Enquanto isso, os vetores da estrutura $eleg$ armazenam apenas as seleções que são elegíveis, ou seja, uma seleção pode ter índice j na matriz de feromônio e ter um índice e, diferente de j, nos vetores da estrutura $eleg$. Para relacionar o índice dos vetores de $eleg$ com o da matriz $\tau[i][j]$, permitindo o cálculo da probabilidade das seleções, foi criado o vetor $seq[e]$. Para ilustrar como funciona, um exemplo com nove seleções é mostrado na Tabela 7. A primeira coluna apresenta todas as seleções j, variando de 1 a 9, e a segunda coluna indica se cada seleção é elegível ou não. No exemplo, somente as seleções 1, 4, 5, 6 e 9 são elegíveis.

Tabela 7 - Seleções do exemplo

j	**Elegível?**
1	Sim
2	Não
3	Não
4	Sim
5	Sim
6	Sim
7	Não
8	Não
9	Sim

Na Tabela 8, a primeira coluna mostra o índice e do conjunto de elegíveis, variando de 1 a 5, por somente cinco seleções serem elegíveis. A segunda coluna relaciona o índice das elegíveis e com o índice do conjunto de seleções j, fazendo o valor de $seq[e]$ igual a j. Para o exemplo, $seq[1] = 1$, $seq[2] = 4$, $seq[3] = 5$, $seq[4] = 6$ e $seq[5] = 9$.

Tabela 8 - Relação entre índice do conjunto de elegíveis e do conjunto de seleções

e	*j (seq[e])*
1	1
2	4
3	5
4	6
5	9

O vetor $eIniTempo[j]$ representa a hora em que é possível iniciar uma seleção, que deve ser após a chegada do lote de vagões relacionado à atividade e na hora em que todos os recursos necessários para executar essa seleção estejam disponíveis. O $eFimTempo[j]$ indica a hora em que a seleção termina, sendo a hora inicial mais a duração da seleção. Por fim, a variável $eTemSetup[e]$ indica se a elegível e tem *setup*,

para isso, analisa se possui recurso com essa característica e o último lote que utilizou esse recurso.

A estrutura $Solucao$ (Figura 8) corresponde a uma solução gerada por uma formiga e é declarada de duas formas no algoritmo, como um vetor de soluções $S[f]$ (f variando de 1 a $maxFormigas$) e como uma única solução S^*, que é a solução global, a melhor encontrada até o momento, independente do ciclo. A variável FO armazena a função objetivo da solução e consiste em minimizar o tempo de estadia do lote de vagões no terminal.

Figura 8 - Estrutura *Solucao*.

$Solucao\{$	
$int\ FO;$	
$int\ schedule[k][t];$	k varia de 1 a $maxAtiv$, t de 1 a $maxHorizonte$;
$int\ iniAtiv[k];$	k varia de 1 a $maxAtiv$;
$int\ fimAtiv[k];$	k varia de 1 a $maxAtiv$;
$int\ modoAtiv[k];$	k varia de 1 a $maxAtiv$;
$int\ caminho[i][j];$	i e j variam de 0 a $maxSelecoes$;
$int\ recOcupPorAtiv[k][t][r];$	k varia de 1 a $maxAtiv$, t de 1 a $maxHorizonte$, r de 1 a maxTpRec;
$int\ recOcup[r][t];$	r varia de 1 a $maxTpRec$, t de 1 a $maxHorizonte$;
$int\ nAtivLote[l];$	l varia de 1 a $maxLotes$.
$\};$	

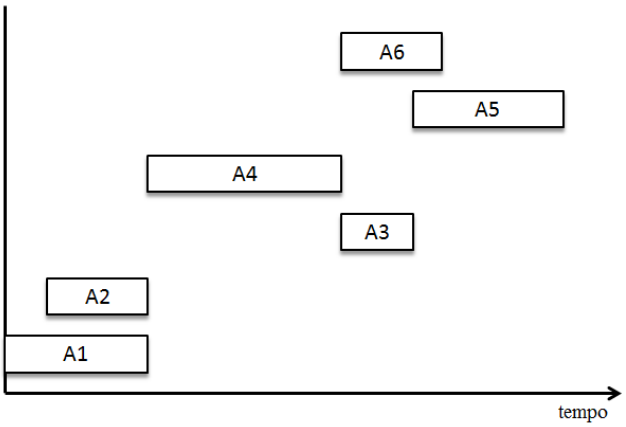
A matriz $schedule[k][t]$ é importante para a análise e apresentação dos resultados, suas linhas e colunas representam atividades e unidades de tempo, respectivamente. Quando uma posição da matriz assume o valor 1, significa que a atividade representada pela linha da posição está sendo processada no tempo representado pela coluna dessa posição. Caso contrário, a posição assume o valor de 0. A matriz funciona como um gráfico de Gantt, indicando início e duração de cada atividade do terminal, como na Figura 9, em que a atividade A4 começa após o término de A1 e A2, as atividades A3 e A6 são iniciadas depois de A4 e a última atividade, A5, começa depois que A3 acaba. Esses horários são determinados por dependência de predecessoras, disponibilidade de recursos, data de chegada do lote, manutenção programada, tempos de *setup* e retenção de recursos. Todos esses fatores determinarão se a atividade pode ser feita em paralelo, buscando reduzir o tempo total, ou se deve esperar o término de outras.

Os vetores $iniAtiv[k]$, $fimAtiv[k]$ e $modoAtiv[k]$ armazenam informações sobre as atividades. O primeiro armazena a hora de início de cada atividade, o segundo a hora de fim e o último o modo de execução em que cada atividade foi realizada. A matriz $caminho[i][j]$ assume 1 se a formiga percorreu o arco (i, j), e 0 caso contrário.

A matriz tridimensional $recOcupPorAtiv[k][t][r]$ indica quais tipos e quantos recursos de cada tipo estão realizando cada atividade em cada unidade de tempo, sendo importante também para a análise e apresentação dos resultados. Para melhor

entendimento, pode-se considerar essa matriz tridimensional como r matrizes bidimensionais de atividades por horizonte de tempo, sendo r a quantidade de tipos de recursos. Então, para uma matriz bidimensional de determinado tipo de recurso, suas posições indicarão a quantidade de recursos do tipo de recurso em questão que estavam ocupados com cada atividade ao longo do tempo.

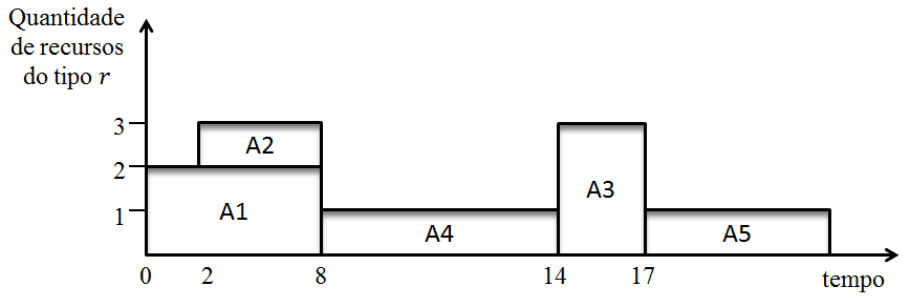
Figura 9 - Gráfico de Gantt ($schedule[k][t]$)



Fonte: Próprio autor.

Na Figura 10 há uma ilustração da ocupação de três recursos do tipo r, permitindo apontar que a atividade A1 demanda dois recursos r, A2 demanda um, a atividade A4, apesar de longa, demanda apenas um recurso, enquanto A3 é curta, porém, precisa dos três recursos para ser processada. A atividade A6 foi realizada em paralelo com A3, Figura 9, pois não demanda nenhum recurso do tipo r, já que todos estavam indisponíveis no momento. Por fim, a atividade A5 foi realizada após o término de A3, ocupando apenas um recurso.

Figura 10 - Matriz $recOcupPorAtiv[k][t][r]$ para o tipo de recurso r.



Fonte: Próprio autor.

A matriz $recOcup[k][t\}$ é um resumo da $recOcupPorAtiv[k][t][r]$, indicando quantos recursos de cada tipo estão ocupados em cada unidade de tempo, sem considerar qual atividade foi realizada. Essa matriz é importante para a verificação mais eficiente da

disponibilidade de recursos ao longo do tempo. E a variável $nAtivLote[l]$ contabiliza quantas atividades por lote foram feitas até o momento, sendo um valor importante para o cálculo da informação heurística da meta-heurística proposta.

4.2 Visão Geral da Meta-heurística Proposta

Esse subcapítulo apresenta a visão geral da meta-heurística proposta, com o pseudocódigo exibido pela Figura 11.

No início, os parâmetros do algoritmo são inicializados com valores pré-determinados pela função *InputParametros* e todas as variáveis necessárias para o funcionamento da ACO são inicializadas pela função *InicializarVariaveis*. Por fim, acontece a leitura da instância pela função *LerDados* e a atribuição de valores às variáveis $ciclo$, f e $numCompletas$, que serão incrementadas ao longo do pseudocódigo.

O algoritmo entra no *loop* de ciclos, rodando até chegar ao número máximo de ciclos estipulado ($numCiclos$). Depois, entra no *loop* de formigas, incrementando f até o número de formigas por ciclo ($numFormigas$). Nessa etapa, a variável que registra a última seleção escolhida pela formiga ($ultimaSel$) é inicializada como 0, indicando que a formiga começará a construir uma solução, escolhendo sua primeira seleção entre as elegíveis, que são as seleções que não possuem predecessoras. O *loop* seguinte (linha 10) só é interrompido quando o número de atividades completadas for igual ao número total de atividades da instância, ou seja, toda a solução será construída nessa etapa.

Primeiro, a estrutura de seleções elegíveis $eleg$ é definida pela função *DefinirElegiveis*, que corresponde aos modos das atividades não realizadas, cujas predecessoras já foram completadas ou que não possuem predecessoras. Após a definição das elegíveis, a função *CalcularInfoHeuristica* calcula a informação heurística de cada uma, assim como o tempo mais cedo disponível para serem iniciadas, considerando chegada do lote, duração da atividade, tempos de *setup* e disponibilidade de recursos, que podem estar em uso por outras atividades, em manutenção programada, ou retidos.

Depois de inserir a opção de retenção de recursos, em que um recurso permanece retido após fim da atividade e até o início da sucessora, essa função foi revisada para também tratar essa característica. A dificuldade de considerar retenção é que, até a atividade sucessora ser escolhida pela formiga e inserida no planejamento, não se sabe quando será iniciada ou mesmo se haverá disponibilidade. Portanto, o término da retenção de um recurso utilizado em determinada atividade só é conhecido ao alocar a sucessora dessa atividade. Para garantir que o recurso retido não seja utilizado por outras atividades, mesmo sem saber o término dessa retenção, a função *CalcularInfoHeuristica* utiliza um tempo de reserva após a duração da atividade na verificação de disponibilidade de recursos.

Figura 11 - Pseudocódigo da Meta-heurística Proposta

```
1  InputParametros ()
2  InicializarVariaveis ()
3  LerDados ()
4  ciclo = 1
5  f = 1
6  numCompletas = 0
7  ENQUANTO (ciclo <= numCiclos)
8      ENQUANTO (f <= numFormigas)
9          ultimaSel = 0
10         ENQUANTO (numCompletas <= numAtividades)
11             DefinirElegíveis (lote, ativ, eleg, tpRec, comp)
12             viabilidade =CalcularInfoHeuristica (eleg, tpRec, S[f], ativ, lote)
13             SE (viabilidade == 0)
14                 CalcularProb (τ, eleg, ultimaSel)
15                 DefinirSelecao (S[f], ultimaSel, eleg, comp)
16                 viabilidade = viabilidade + OcuparRecursos (S[f], ativ)
17                 SE (viabilidade == 0)
18                     numCompletas + +
19                 FIM-SE
20             FIM-SE
21             SE(viabilidade > 0)
22                 numCompletas = numAtividades + 1
23             FIM-SE
24         FIM-ENQUANTO
25         SE(viabilidade = 0)
26             CalcularFO (S[f], lote)
27             SE (S[f].FO < S*.FO)
28                 CopiarSolucao (S[f], S*)
29                 CalcularDeltaTauGlobal (S, Δτ*)
30             FIM-SE
31         FIM-SE
32         viabilidade = 0
33         f + +
34     FIM-ENQUANTO
35     RankearSolucoes (S, μ)
36     CalcularDeltaTau (S, Δτ)
37     CalcularTau (S, τ, Δτ, Δτ*, ρ)
38     ciclo + +
39 FIM-ENQUANTO
```

Devido a essa quantidade de restrições, o algoritmo pode não encontrar disponibilidade para uma elegível dentro do horizonte de tempo, o que torna a solução inviável. Nesse caso, a função retorna 1 para a variável $viabilidade$, e caso não haja problemas, retorna

0. Se a solução for viável, a probabilidade de uma elegível ser escolhida é calculada pela função *CalcularProb*, levando em conta o valor heurístico e a matriz de feromônios $\tau[i][j]$. Com a probabilidade da formiga ir de seu atual nó para cada nó elegível, um número aleatório é gerado na função *DefinirSelecao* para auxiliar na escolha da seleção {atividade, modo de execução}. Além disso, nessa função também são registradas informações sobre a seleção escolhida e a construção da solução, auxiliando na apresentação de resultados e no funcionamento da meta-heurística.

Com atividade e modo já definidos, a função *OcuparRecursos* ocupa os recursos demandados pela seleção ao longo de sua duração. Também é seu papel atualizar a retenção de recursos (que possuam essa característica) com o tempo de reserva após duração, além de ajustar término da retenção e liberar recursos da predecessora que estavam retidos. Quando não é possível reter um recurso pelo tempo necessário devido à indisponibilidade de recursos, a função retorna 1, indicando que a solução é inviável. Caso não haja problema na retenção, retorna 0 para indicar a viabilidade até o momento. A variável $viabilidade$ recebe seu próprio valor somado ao valor retornado pela função *OcuparRecursos*.

Se a solução for viável, $numCompletas$ é incrementada, indicando que mais uma atividade foi programada. Se houver inviabilidade, indicada por *CalcularInfoHeuristica* ou por *OcuparRecursos*, a variável $viabilidade$ será maior que 0, e $numCompletas$ recebe $numAtividades + 1$, forçando sua saída do *loop* de criação da solução. Essas funções serão detalhadas no subcapítulo seguinte, no tópico de construção da solução.

Ao programar todas as atividades, a solução está criada e o algoritmo sai do *loop* de atividades. A função *CalcularFO* calcula a função objetivo, que é a soma do tempo de estadia de todos os lotes de vagões no terminal e é detalhada no subcapítulo seguinte, no tópico compilação de soluções. Sendo $S[f]$ a solução que acabou de ser construída, se $S[f].FO$ for menor que a função objetivo da solução global $S^*.FO$, a função *CopiarSolucao* copia $S[f]$ para S^*, definindo uma nova solução global. E sempre que isso acontece, é preciso recalcular a matriz de quantidade de feromônio que será adicionada aos arcos pela formiga global ($\Delta\tau^*[i][j]$) com a função *CalcularDeltaTauGlobal*. Depois disso, o algoritmo sai da verificação da solução global, reinicializa $viabilidade$ e incrementa o número de formigas f.

Ao rodar todas as formigas do ciclo, construindo suas soluções e calculando e comparando suas funções objetivo, o *loop* de formigas chega ao fim. O próximo passo é atualizar a matriz de feromônio $\tau[i][j]$, que é feito nessa meta-heurística com a estratégia *Elitist-Rank Based Strategy* (AS_{rank}), proposta por *Bullnheimer et al.* (1997). Todas as funções ligadas ao AS_{rank}, ou seja, à atualização de feromônios, são detalhadas no subcapítulo seguinte, no tópico estratégia de atualização de feromônios. Essa estratégia consiste em atualizar a matriz de feromônio levando em conta a melhor solução encontrada até o momento (global) e as melhores soluções do ciclo que acabou de ser finalizado. Os feromônios da solução global são multiplicados pelo maior peso, enquanto os feromônios das soluções do ciclo são ordenados em um *ranking* de acordo

com a qualidade de suas funções objetivo e são multiplicadas por um peso proporcional às suas posições nesse *ranking*.

A função *RankearSolucoes* cria um *ranking*, ordenando as soluções de acordo com suas funções objetivos. A função *CalcularDeltaTau* calcula a quantidade de feromônios que será adicionada nos arcos por todas as formigas do ciclo completado e a função *CalcularTau* atualiza a matriz de feromônio $\tau[i][j]$, evaporando parte da quantidade de feromônio que havia e somando as quantidades de feromônios adicionadas pela solução global ($\Delta\tau^*[i][j]$) e pelas formigas do ciclo ($\Delta\tau[i][j]$).

Posteriormente, um novo ciclo é iniciado e todo o processo de geração de soluções e atualização de feromônios é repetido até atingir um limite máximo de ciclos especificado como parâmetro da meta-heurística. Então, as soluções das formigas de um ciclo são influenciadas pela solução global e pelas soluções geradas por formigas de ciclos anteriores (feromônios), além da informação heurística, que tem origem na própria solução que ajuda a construir. Portanto, a probabilidade das escolhas das formigas vem, em parte, da memória do algoritmo e, em parte, das características da solução, além de um fator aleatório na escolha dos nós.

4.3 Detalhamento das Funções

Nesse subcapítulo, serão detalhadas as principais funções da meta-heurística proposta, divididas em três tópicos: 1) construção da solução; 2) compilação de soluções; e 3) atualização de feromônios.

4.3.1 Construção da solução

A função DefinirElegiveis roda o número de atividades para verificar quais são elegíveis. Se uma atividade for elegível, todos os seus modos também são, pois dependem das mesmas predecessoras.

O primeiro passo é verificar se a atividade já foi concluída. Cada atividade só pode ser feita uma vez, portanto, as seleções que correspondem aos modos de execuções de atividades que já foram realizadas não entram na lista de elegíveis, não podem ser escolhidas. A seguir, é verificado se a atividade tem predecessoras, caso não tenha, seus modos de execução já são definidos como seleções elegíveis. Caso tenha, o vetor auxiliar $comp[k]$ é utilizado para indicar se as predecessoras (vetor $predecessoras[g]$) foram realizadas. Se verificado que alguma não foi feita, os modos da atividade são definidos como não elegíveis, e se todas foram feitas, são definidos como seleções elegíveis, registrando as variáveis $eAtiv[e]$, $eModo$ e seq da estrutura $eleg$.

Definidas as seleções elegíveis, a função *CalcularInfoHeuristica* calcula, não só a informação heurística de cada elegível (cada nó que a formiga pode ir), como também o início ($eIniTempo[e]$) e término ($eFimTempo[e]$) mais cedo possíveis, respeitando

disponibilidade de recursos, e indica se a elegível irá demandar tempo de *setup* ($eTemSetup[e]$) e em qual recurso.

Os recursos demandados por cada elegível são varridos ao longo do horizonte de tempo, buscando um período em que todos possam ser ocupados, respeitando as restrições do problema. Quando esse período é encontrado para o primeiro recurso, é testado também para os outros, e caso todos consigam ser ocupados, o início da elegível ($eIniTempo[e]$) é definido. Caso contrário, o algoritmo volta para o primeiro recurso e testa novamente sua disponibilidade, porém começando a partir do fim do período encontrado anteriormente, somado a 1 unidade de tempo. Se a verificação chegar ao final do horizonte e não encontrar o período de disponibilidade, a solução é considerada inviável. Para aumentar a eficiência do algoritmo, a verificação de disponibilidade de início da elegível ($tIni$) é feita a partir do instante de chegada do lote no terminal (lote relacionado à seleção em questão), dado pela variável $Chegada$, evitando percorrer todo o horizonte de tempo.

A Figura 12 mostra um exemplo simplificado de verificação de disponibilidade de recursos para uma seleção elegível e em um cenário com dois recursos de cada tipo. A elegível e tem duração de três unidades de tempo e demanda dois recursos do tipo $r1$, um recurso do tipo $r2$ e dois recursos do tipo $r3$ para ser executada. Considerando que o lote relacionado à e chega no tempo $t3$, a verificação começa a partir desse instante, com a matriz $recOcup[r][t]$ indicando a quantidade de recursos de cada tipo ocupada em cada unidade de tempo. Como há um recurso $r3$ ocupado ($recOcup[r3][t3] = 1$), não há disponibilidade em $t3$, visto que são necessários dois recursos $r3$ e só há um disponível, e em $t4$ também não há quantidade suficiente de recursos $r1$ e $r3$. Somente em $t5$ haverá disponibilidade para a elegível ser realizada, visto que há quantidade suficiente dos recursos demandados em $t5$, $t6$ e $t7$, que é a duração da elegível e. Caso não houvesse disponibilidade em $t7$, por exemplo, a verificação para e ia recomeçar a partir de $t8$.

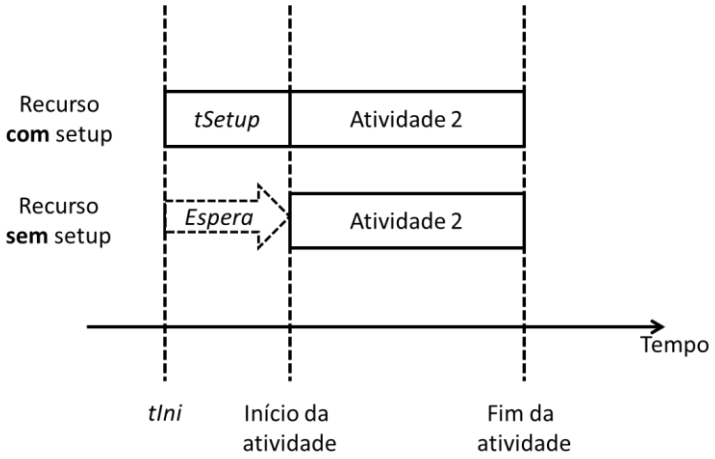
Figura 12 - Verificação de disponibilidade na matriz $recOcup[r][t]$

Tipo de Recurso	t1	t2	t3	t4	t5	t6	t7	t8
r1	0	0	0	1	0	0	0	0
r2	0	1	1	1	1	0	0	0
r3	2	2	2	1	0	0	0	0

Além de considerar os recursos ocupados com atividades ($recOcup[r][t]$) para a disponibilidade, também são considerados os tempos de *setup*, a manutenção programada e os recursos retidos. Como o tempo inicial é definido por elegível e não por recurso, as variáveis $espera$ e $tSetup$ são utilizadas para auxiliar no tratamento dos tempos de *setup*, já que os recursos com essa característica devem ser ocupados antes dos demais recursos utilizados na atividade. A Figura 13 ilustra o funcionamento dessas variáveis. O recurso com *setup* é ocupado de $tIni$ até $tIni + tSetup +$ duração da

atividade, enquanto o recurso sem *setup* é de $tIni + espera$ até $tIni + espera +$ duração da atividade.

Figura 13 - Variáveis $espera$ e $tSetup$



Para generalizar o cálculo, foram definidos quatro cenários possíveis relacionados a *setup* na verificação de disponibilidade de recursos, Tabela 9. No Cenário 1, $espera$ e $tSetup$ estão zerados, pois só há *setup* se o recurso já foi utilizado por outro lote antes, então todos os recursos são ocupados ao mesmo tempo, logo em $tIni$. O mesmo acontece no Cenário 2, em que a atividade não possui recursos com característica de *setup*. Já nos Cenários 3 e 4, a atividade tem *setup*, a diferença é que no 3 o tipo de recurso testado pelo algoritmo no momento não é o que possui *setup*, portanto será ocupado mais tarde, enquanto no último cenário, o recurso testado possui *setup* e deve ser ocupado com antecedência. O tempo de *setup* será determinado de acordo com a matriz $tempoSetup[maxLotes][maxLotes]$. Com a utilização desses cenários e variáveis auxiliares, considerando *setup* e duração da atividade, os recursos com e sem *setup* são verificados no período de $tIni + espera$ até $tIni + tSetup +$ duração da atividade.

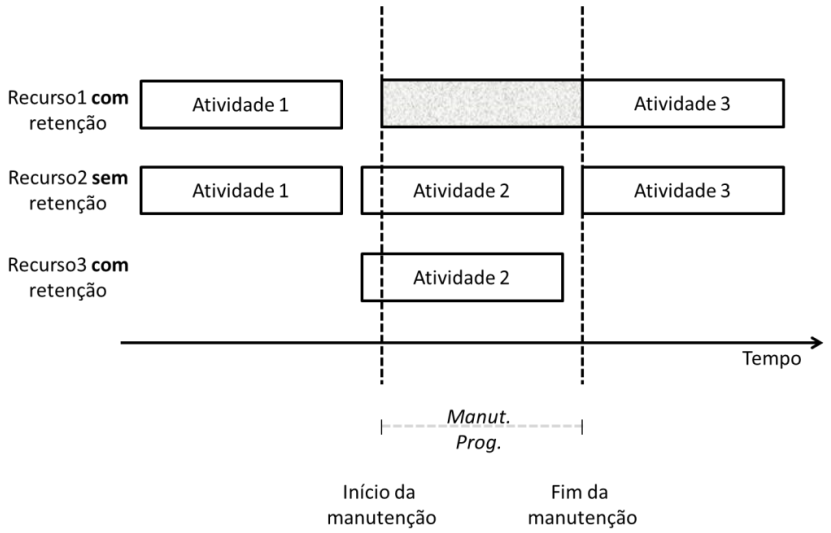
Tabela 9 - Cenários de *Setup*

Número	Cenário	*espera*	*tSetup*
1	Atividade tem recurso com característica de *setup*, mas recurso ainda não foi utilizado, então não terá *setup*.	0	0
2	Atividade não tem recurso com característica de *setup*	0	0
3	Atividade tem recurso com característica de *setup*, mas não é o recurso testado.	Tempo de *setup*	Tempo de *setup*
4	Atividade tem recurso com característica de *setup*, é o recurso testado e já foi utilizado em outra atividade.	0	Tempo de *setup*

A manutenção programada, ou indisponibilidade de recursos, acontece em um período de tempo pré-definido pela instância, ou seja, não é determinado pela meta-heurística. A

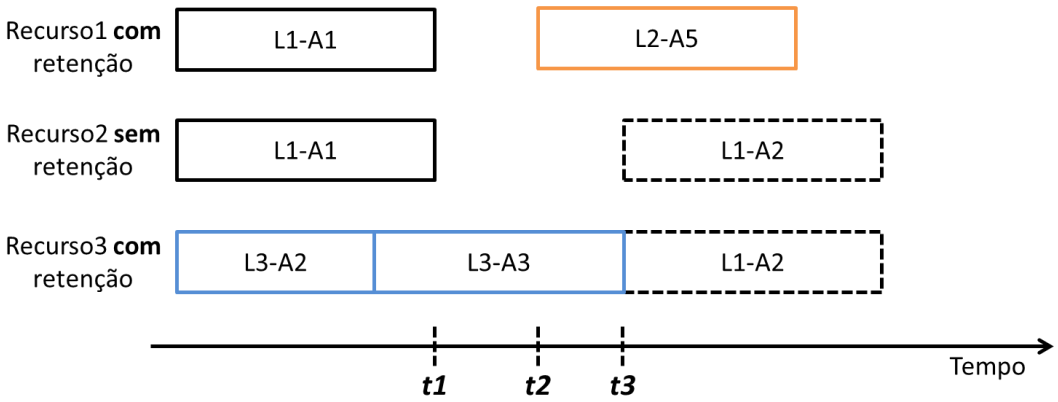
Figura 14 representa um pouco do funcionamento dessa característica do problema e do seu tratamento pela meta-heurística. No esquema, há uma manutenção programada do recurso 1, que possui retenção. Além do recurso não poder ser utilizado durante essa indisponibilidade, também não pode estar retido, portanto sua sucessora já deve ter sido iniciada para que ele seja liberado. A atividade 1 utiliza os recursos 1 e 2, e a sua sucessora, atividade 2 (demandando os recursos 2 e 3), pode começar até o início da indisponibilidade para que a solução seja viável. Caso contrário, a solução será descartada. No exemplo, a atividade 3 aguarda o fim do período cinza para iniciar, mesmo com o recurso 2 disponível antes.

Figura 14 - Esquema de Manutenção Programada



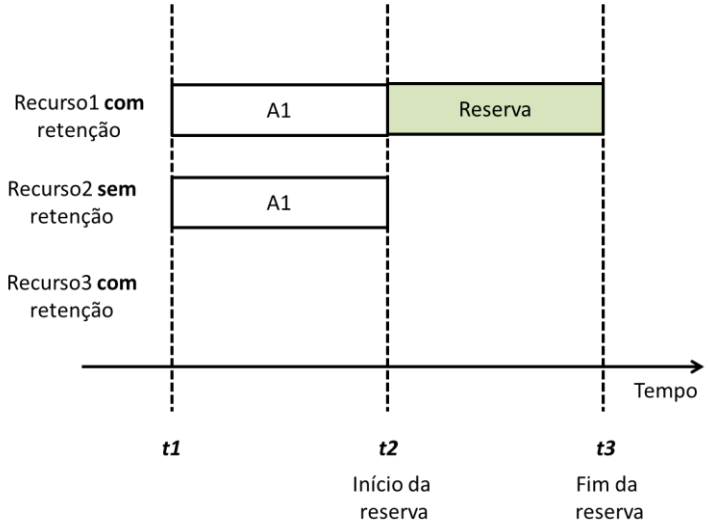
Na meta-heurística, ao agendar uma atividade, não se sabe ainda quando, nem para quando, será agendada a sua sucessora, portanto, a informação do fim da retenção de um recurso é desconhecida no momento do agendamento da atividade que o utiliza. Isso torna essa característica muito complicada de ser tratada na meta-heurística, pois resulta em muita inviabilidade. A Figura 15 ilustra o processo. A atividade 1 do lote 1, L1-A1, acaba em $t1$, ocupando os recursos 1 e 2, após esse período, o recurso 1 ficará retido até que sua sucessora L1-A2 seja realizada. As atividades do lote 3, L3-A2 e L3-A3, impedem que L1-A2 seja realizada até o tempo $t3$, porém, o recurso 1 já está ocupado a partir de $t2$ com L2-A5. Logo, a solução é inviável, pois o recurso 1 deveria estar retido até $t3$, mas está realizando uma atividade em $t2$. Essa solução seria viável apenas se a sucessora L1-A2 pudesse ser alocada entre $t1$ e $t2$. Essa dificuldade na geração de soluções e garantia de viabilidade motivou a criação do tempo de reserva, um pulmão de tempo inserido no agendamento de recursos com retenção que reduz bruscamente as inviabilidades por retenção.

Figura 15 - Retenção



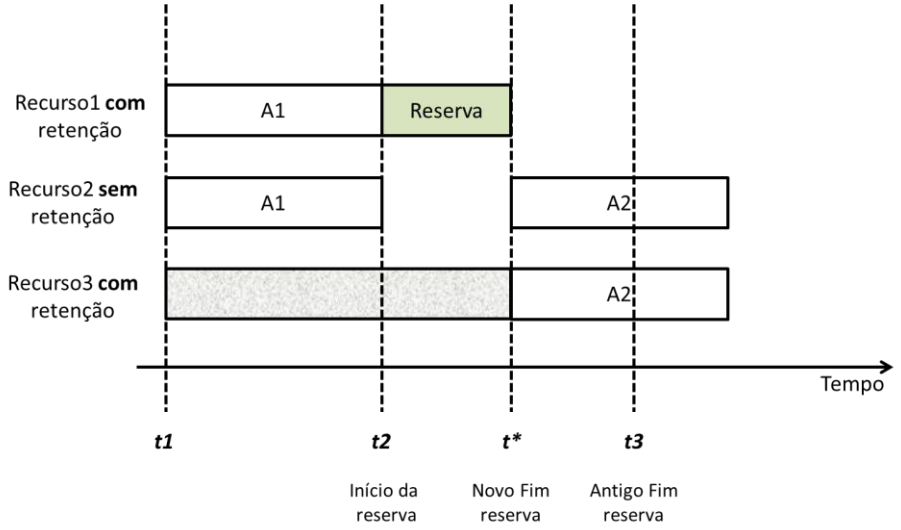
Para explicar o tempo de reserva, a Figura 16 mostra o momento logo após o agendamento de uma atividade A1. Foi verificado se havia disponibilidade para o recurso 1 entre $t1$ e $t3$, ou seja, considerando o tempo de reserva, e para o recurso 2 entre $t1$ e $t2$. Com isso, garante-se que a sucessora terá ao menos um pulmão de tempo para ser realizada. Ressaltando, caso houvesse uma atividade sendo realizada pelo recurso 1 entre $t2$ e $t3$, a atividade A1 só poderia ser agendada após $t3$.

Figura 16 - Tempo de Reserva



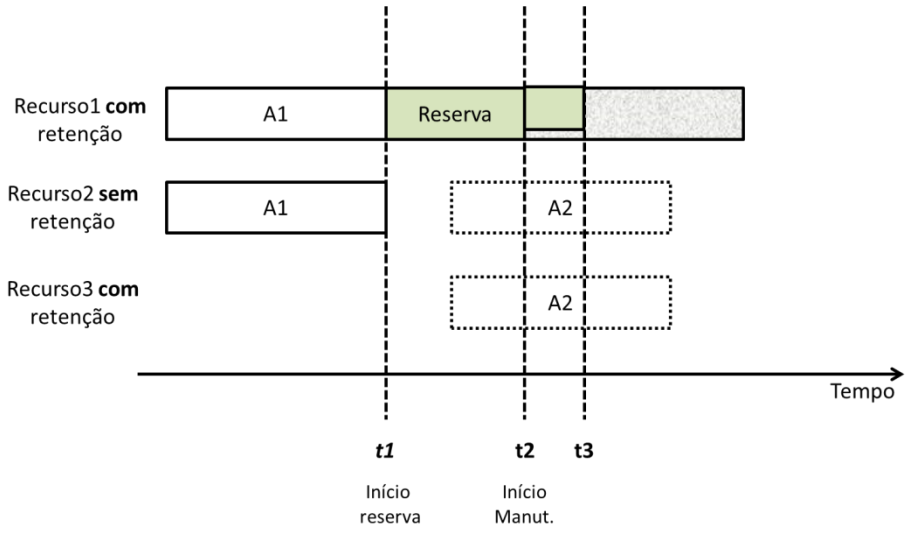
Na Figura 17, tem-se o instante após o agendamento de A2, sucessora de A1 (A1 agendada na Figura 16). Considerando que A2 foi agendada em t^*, devido à ocupação prévia do recurso 3, o tempo de reserva do recurso 1 é atualizado, passando a terminar em t^*, ao invés de $t3$.

Figura 17 - Atualização de Tempo de Reserva



Com a criação do tempo de reserva, começou um conflito entre esse mecanismo e a manutenção programada. Na Figura 18, um recurso com retenção $r1$ recebe uma manutenção, ficando indisponível a partir de $t2$. Porém, há tempo para realizar a atividade A1, e sua sucessora A2 antes de $t2$, liberando $r1$ a tempo para a manutenção programada. Com o mecanismo de tempo de reserva de $t1$ até $t3$, não seria possível fazer essas atividades antes de $t2$, pois a reserva iria conflitar com o período de manutenção e A1 seria agendado somente após o fim da indisponibilidade, gerando um agendamento ruim. A solução foi tornar o tempo de reserva dinâmico nessas situações em que o recurso tem retenção e período de indisponibilidade, e quando a atividade começa antes desse período e tempo de reserva acaba depois.

Figura 18 - Tempo de reserva x Manutenção Programada



É importante ressaltar que o conceito de atualização da retenção dos recursos (tempo de reserva) auxilia no entendimento da verificação de disponibilidade, por isso foi

abordado nessa função. Porém, é realizado na função *OcuparRecursos*, então outros detalhes dessa atualização são explicados posteriormente.

Inicialmente, o cálculo da informação heurística privilegiava as atividades com disponibilidade mais próxima de realização. Então, quanto mais próxima estava a disponibilidade de uma seleção, maior era seu valor heurístico. Após inserir a característica de retenção, esse cálculo precisou ser reformulado, pois começava atividades de diferentes lotes ao mesmo tempo e não conseguia liberar os recursos com retenção devido a falta de recursos para atividades sucessoras. Ou seja, um gargalo de recursos no processo fazia com que muitos recursos ficassem retidos e tornava a solução ruim. Para o problema tratado, é importante dar maior prioridade para a liberação de recursos, já que esse tempo de retenção seria uma perda de produtividade do recurso. Com isso, o cálculo da nova informação heurística para cada elegível ($eleg.\eta[e]$) é exibido na Figura 19, sendo $c1$ e $c2$ coeficientes de peso, multiplicados respectivamente pelo número de atividades já realizadas pelo lote da elegível ($sol.nAtivLote[eL]$) e pelo número de predecessoras que essa elegível possui. Portanto, com foco para liberar recursos retidos.

Figura 19 - Informação Heurística

$e = 1$
ENQUANTO ($e < numElegiveis$)
    $eA = eleg.eAtiv[e]$
    $eL = ativ[eA].lote$
    $eleg.\eta[e] = 1 + c1 * sol.nAtivLote[eL] + c2 * ativ[eA].numPred$
    $e++$
FIM-ENQUANTO

A função *CalcularProb* calcula a probabilidade que cada elegível tem de ser escolhida pela formiga e é apresentada na Figura 20. A variável $divisor$ é incrementada no loop de elegíveis com o produto entre $\tau[i][eleg.seq[e]]^{\propto}$ e $eleg.\eta[e]^{\beta}$, onde i indica onde a formiga estava (ultima seleção escolhida), e $eleg.seq[e]$ é o índice correspondente da elegível e no conjunto de todas as seleções (índice na matriz $\tau[i][j]$). O vetor $eleg.\eta[e]$ utiliza o índice e pois só é calculado para as elegíveis, portanto não é necessário acessar o conjunto de todas as seleções.

Após achar o $divisor$, a probabilidade de cada elegível e ($eleg.prob[e]$) é calculada elevando a $\propto$ a quantidade de feromônio no arco entre a posição atual da formiga e a elegível e, multiplicando pelo valor heurístico da elegível elevado a β e dividindo esse produto pela variável $divisor$.

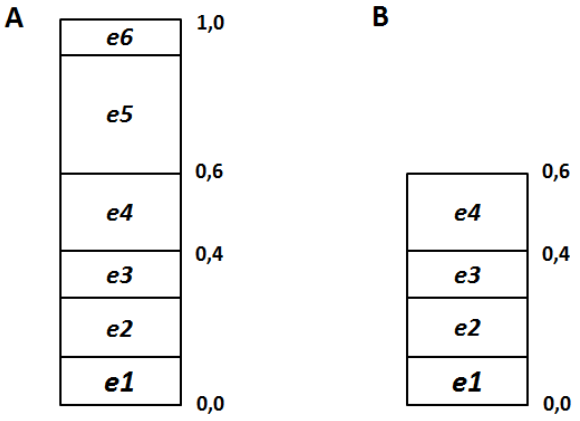
Figura 20 - Cálculo de probabilidade das seleções elegíveis

$divisor = 0$
$e = 1$
$i = ultimaSel$
ENQUANTO ($e \leq numElegiveis$)
    $divisor = divisor + \tau[i][eleg.seq[e]]^{\propto} * eleg.\eta[e]^{\beta}$
    $e = e + 1$
FIM-ENQUANTO
$e = 1$
ENQUANTO ($e \leq numElegiveis$)
    $eleg.prob[e] = \tau[i][eleg.seq[e]]^{\propto} * eleg.\eta[e]^{\beta} / divisor$
    $e = e + 1$
FIM-ENQUANTO

Com as probabilidades de cada elegível, a função *DefinirSelecao* define a próxima seleção {atividade, modo de execução} a ser programada. Um valor aleatório entre 0 e 1 é gerado (x), utilizando como semente aleatória a hora do sistema em segundos. Depois, o algoritmo roda o número de elegíveis calculando uma probabilidade acumulada (pAc) a partir das probabilidades individuais ($eleg.prob[e]$). A cada probabilidade de elegível acrescentada à pAc, é averiguado se x é menor ou igual a essa variável, ou seja, se a probabilidade acumulada ultrapassou o valor aleatório gerado. Caso tenha ultrapassado, significa que x está na faixa de valores coberta pela probabilidade da última elegível (e) somada. Então, essa elegível é escolhida pela formiga e as variáveis de controle da solução são atualizadas.

Em um exemplo com seis elegíveis, a Figura 21-A mostra como ficaria a probabilidade acumulada das seleções elegíveis. Considerando que o valor aleatório gerado (x) fosse de 0,45, a elegível escolhida seria a $e4$, pois sua probabilidade vai de 0,4 a 0,6, portanto, compreende o valor de x. A partir do momento em que o intervalo de probabilidade que contém x é encontrado, o algoritmo para de calcular a probabilidade acumulada e define a elegível como a nova seleção a ser programada, Figura 21-B.

Figura 21 - Escolha da seleção na função *DefinirSelecao*



Definida a seleção {Atividade, modo de execução}, a função *OcuparRecursos* atualiza retenções e ocupa os recursos nos tempos demandados pela seleção que acabou de ser escolhida. Se a nova seleção (atividade A2) possuir uma predecessora A1, é verificado se houve retenção em algum dos recursos utilizados por A1. Caso haja, inicia-se um processo para atualização do tempo de reserva, que se tornará de fato o tempo que o recurso ficou retido, Tabela 10. No primeiro cenário, o tempo de reserva alocado foi mais que suficiente, e o recurso será liberado de parte desse tempo, com fim da retenção igual ao tempo anterior ao início de A2. No segundo cenário, o fim da retenção de A1 é calculado da mesma forma, porém agora, A2 começa após o fim da reserva de A1. Isso implica em aumentar o tempo de reserva do recurso retido até antes de início de A2. Nesse caso, esse aumento na retenção pode ser inviável, descartando a solução.

Tabela 10 - Ações de Atualização de Recursos

Cenário	Liberar/reter Recursos	Novo Fim de Retenção de A1
Sucessora começa antes de fim de reserva de tempo	Libera o recurso entre início de A2 até fim de tempo de reserva de A1	início de A2 - 1
Sucessora começa após fim de reserva de tempo	Ocupa recurso entre fim atual de reserva de A1 até um tempo antes de início de A2	início de A2 - 1

Se a solução for viável, não possuindo predecessora com retenção de recurso, ou atualizando com sucesso o tempo de retenção, o algoritmo ocupa os recursos utilizados pela nova seleção. A função passa por cada tipo de recurso e atualiza, desde o tempo inicial da atividade até seu fim, a quantidade de recursos ocupados do tipo com a respectiva quantidade demandada pela seleção e já considerando tempo de *setup*. Logo após, é verificado se essa seleção possui algum recurso com característica de retenção, e se não é a última do lote. Se aplicável, o recurso será retido pelo tempo de reserva.

4.3.2 Compilação de Soluções

De acordo com a Figura 22, a função *CalcularFO* calcula a função objetivo da formiga f, considerando $numLotes$ como o número total de lotes da instância. Como a variável $numAtivRota$ indica a quantidade de atividades na rota do lote, a posição de número $numAtivRota$ do vetor $ativRota[maxAtiv]$ contém o número identificador da ultima atividade que deve ser feita pelo lote. Com esse número, é possível acessar a hora de término dessa atividade, através de $S[f].fimAtiv$, e subtrair a hora de chegada do lote ($lote[l].chegada$), encontrando o tempo de estadia no terminal para cada lote de vagões. Então, os tempos de estadia de todos os lotes são somados para determinar o valor da função objetivo ($S[f].FO$).

Figura 22 - Cálculo da função objetivo

$l = 1$
$S[f].FO = 0$
ENQUANTO ($l \leq numLotes$)
    $S[f].FO = S[f].FO + S[f].fimAtiv[lote[l].ativRota[numAtivRota]] - lote[l].chegada$
    $l = l + 1$
FIM-ENQUANTO

4.3.3 Estratégia de Atualização de Feromônios

Como já apresentado anteriormente, a estratégia de atualização de feromônios é a *Elitist-Rank Based Strategy* (AS_{rank}), proposta por Bullnheimer *et al.* (1997) e apresentada na Figura 23. Essa atualização acontece após todas as formigas de cada ciclo construírem suas soluções. O algoritmo percorre toda a matriz, calculando a nova quantidade de feromônios ($\tau[i][j]$) para cada arco saindo de i para j ($i, j \in [0, numSelecoes]$), onde $numSelecoes$ é o número total de seleções da instância.

O novo valor de $\tau[i][j]$ é a própria quantidade de feromônios de cada arco ($\tau[i][j]$) depois da evaporação (ρ), somada às quantidades de feromônio adicionadas por todas as formigas do ciclo ($\Delta\tau[i][j]$) e pela formiga global ($\Delta\tau^*[i][j]$). A taxa de evaporação ρ é importante para a renovação da memória da meta-heurística proposta, permitindo que soluções ruins sejam aos poucos descartadas.

Figura 23 - Atualização da Matriz de Feromônios

$i = 1$
$j = 1$
ENQUANTO ($i \leq numSelecoes$)
    ENQUANTO ($j \leq numSelecoes$)
        $\tau[i][j] = (1 - \rho) * \tau[i][j] + \Delta\tau[i][j] + \Delta\tau^*[i][j]$
        $j = j + 1$
    FIM-ENQUANTO
    $i = i + 1$
FIM-ENQUANTO

Após construir uma solução $S[f]$, é averiguado se essa solução é melhor que a global (S^*), e caso seja, se torna a nova S^* pela função *CopiarSolucao*. Quando uma nova solução global é encontrada, a função *CalcularDeltaTauGlobal* recalcula a matriz $\Delta\tau^*[i][j]$, de acordo com a Figura 24.

As formigas do *ranking*, que contribuem para $\Delta\tau[i][j]$, são todas do ciclo que acabou de ser completado, enquanto a formiga global, relacionada a $\Delta\tau^*[i][j]$, pode ser de qualquer um dos ciclos realizados pela meta-heurística. Para um número λ de formigas no *ranking*, a variável σ é igual a $\lambda + 1$, e indica o número total das formigas elitistas,

ou seja, as formigas do ranking mais a formiga global. No algoritmo, a função $inicializarDeltaTauGlobal$ zera todos os valores de $\Delta\tau^*[i][j]$ para que novos valores sejam atribuídos.

Figura 24 - Cálculo da matriz $\Delta\tau^*[i][j]$

```
inicializarDeltaTauGlobal (Δτ*)
ENQUANTO (i ≤ numSelecoes)
    ENQUANTO (j ≤ numSelecoes)
        SE (S*.caminho[i][j] == 1)
            Δτ*[i][j] = (σ * Q )/( S*.FO)
            j = j + 1
        FIM-SE
    FIM-ENQUANTO
    i = i + 1
FIM-ENQUANTO
```

Então, o valor de $\Delta\tau^*[i][j]$ $(i, j \in [0, numSelecoes])$ é alterado somente quando o valor de $S^*.caminho[i][j]$ é 1, mostrando que a formiga foi do nó i para o j. O valor de $\Delta\tau^*[i][j]$ é calculado como a variável σ multiplicada pela quantidade de feromônio deixada pela formiga ao longo de todo o ciclo (Q) e dividida pela função objetivo da própria solução global $(S^*.FO)$. Portanto, quanto menor a função objetivo, maior será a influência da solução global.

Para criar o *ranking*, é preciso ordenar as soluções de acordo com suas funções objetivo, o que é feito pela função *RankearSolucoes*. Essa função preenche o vetor μ[f] $(f \in [1, numFormigas])$, onde a formiga com a menor função objetivo recebe o valor 1, a segunda menor recebe 2 e assim em diante, até a formiga de maior função objetivo, que recebe a posição λ.

Para o cálculo da matriz $\Delta\tau[i][j]$, a quantidade de feromônio que será adicionada por todas as formigas do *ranking* será como mostrado na Figura 25. A função calcula, para todos os arcos (i, j) de todas as formigas do ciclo, a quantidade adicionada por cada formiga f. Um peso de contribuição é calculado como o resultado do número de formigas elitistas σ menos a posição da formiga f no *ranking* $(\mu[f])$. Então, o produto entre esse peso e Q é dividido pela solução objetivo de f (S$[f].FO$), chegando à quantidade de feromônio adicionada por f. Então, a formiga f^* com a melhor solução do ciclo terá $\mu[f^*]$ igual a 1 e peso $\sigma - 1$, enquanto a formiga f^- com a pior solução terá seu $\mu[f^-]$ igual a λ e peso 1 ($\sigma = \lambda + 1$). Assim, esse peso controla a influência das formigas de acordo com suas posições no ranking.

Figura 25 - Cálculo da matriz $\Delta\tau[i][j]$

```
f = 1
i = 1
j = 1
inicializarDeltaTau (Δτ)
ENQUANTO (f ≤ numFormigas)
    ENQUANTO (i ≤ numSelecoes)
        ENQUANTO (j ≤ numSelecoes)
            SE (S*.caminho[i][j] == 1)
                Δτ[i][j] = Δτ[i][j] + ((σ − μ[f]) * Q)/( S[f].FO)
                j = j + 1
            FIM-SE
        FIM-ENQUANTO
        i = i + 1
    FIM-ENQUANTO
    f = f + 1
FIM-ENQUANTO
```

Com os valores das matrizes de $\Delta\tau[i][j]$ e $\Delta\tau^*[i][j]$ atualizados, é possível então fazer o cálculo da matriz de feromônios que influenciará o ciclo seguinte, o que irá acontecer até o momento em que o número de ciclos estipulado em $numCiclos$ seja alcançado.

5. RESULTADOS E ANÁLISES

Esse capítulo apresenta a calibração dos parâmetros da meta-heurística proposta e os resultados.

5.1 Calibração de Parâmetros

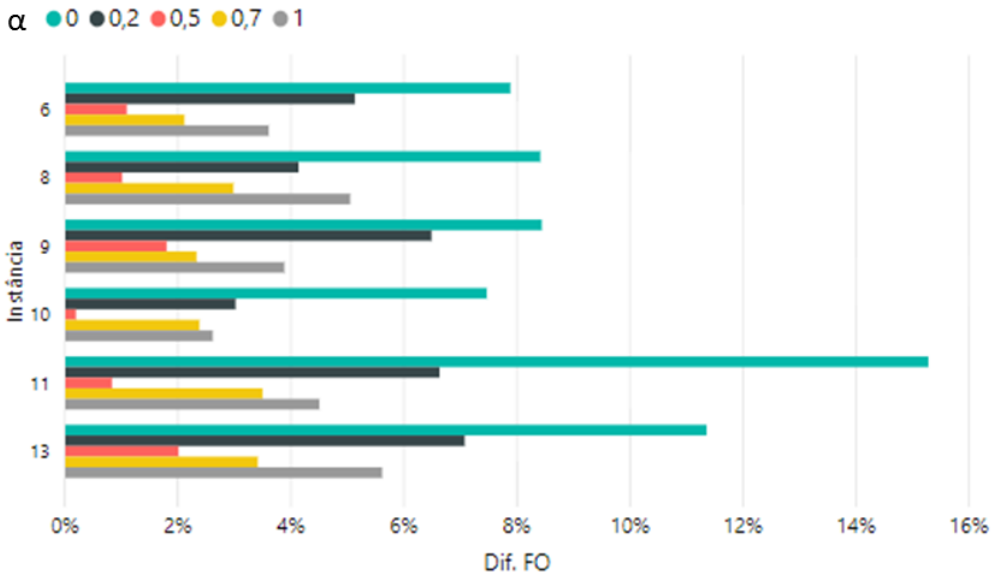
Os parâmetros calibrados para a meta-heurística proposta foram: a influência do feromônio $\propto$, a influência do valor heurístico β, a taxa de evaporação dos feromônios entre seleções ρ, o número de formigas por ciclo $numFormigas$ e o número de ciclos realizados $numCiclos$.

Primeiro, os valores de $\propto$ e β foram avaliados. Como esses parâmetros indicam a influência relativa do feromônio e da informação heurística, respectivamente, o valor de β foi fixado em 2,0 e a influência do feromônio $\propto$ variou entre 0 e 1. Ressaltando que um $\propto$ igual a 0 tira o aprendizado da meta-heurística, não considerando o feromônio para o cálculo da probabilidade de escolha da formiga. As Instâncias 6, 8, 9, 11 e 13 foram utilizadas na calibração de $\propto$, para garantir o valor que melhor atenda diferentes tamanhos de instâncias. Para cada valor testado e cada instância, a meta-heurística foi executada 10 vezes. O parâmetro ρ foi fixado em 0,5, o número de formigas por ciclo em 150 e o número de ciclos em 100. Esses valores foram definidos empiricamente, com base nos testes feitos ao longo do desenvolvimento da meta-heurística.

A Figura 26 mostra os resultados dos testes com a variável $\propto$ assumindo os valores: 0; 0,2; 0,5; 0,7; e 1,0 para $\beta = 2{,}0$. O eixo X indica a *Dif. FO*, que é a diferença percentual entre a média das 10 funções objetivo obtidas pela meta-heurística proposta e do CPLEX. Esse valor é calculado subtraindo a FO do CPLEX da FO Média da meta-heurística e dividindo o resultado pela FO do CPLEX. Portanto, quando o valor é positivo, o CPLEX teve melhor desempenho, caso contrário, a meta-heurística foi melhor, e quando for zero, ambas tiveram o mesmo resultado. O conceito de *Dif. FO* será usado em outras análises na dissertação e deve ser interpretado da mesma forma. Essa medida é importante para comparar a qualidade dos resultados entre instâncias com diferentes funções objetivo. O eixo Y mostra as instâncias analisadas, e para cada instância, há 5 barras, cada uma representando um valor de $\propto$ avaliado.

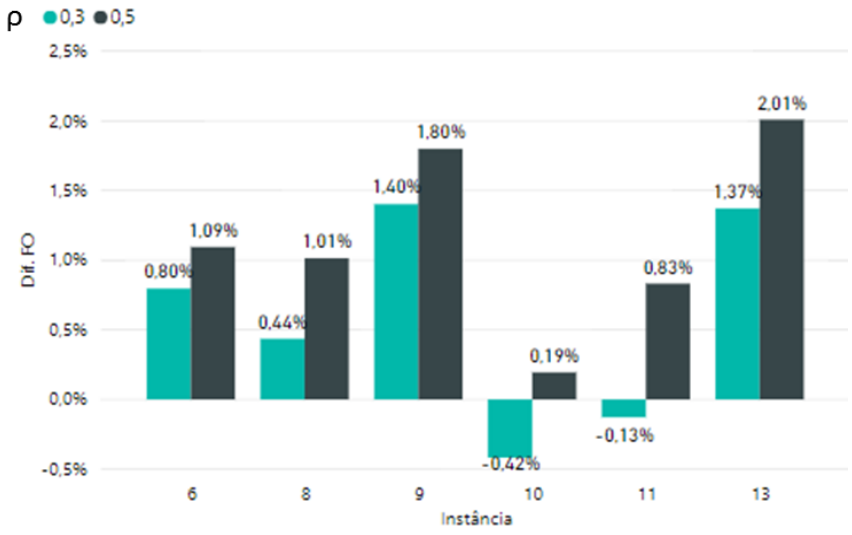
Pode-se observar que o $\propto$ igual a 0 (barra verde) retorna os piores resultados, em especial na Instância 11, que é a mais complexa. A falta de direcionamento do feromônio dificulta a obtenção de soluções melhores. Analisando o gráfico, a menor diferença percentual considerando $\beta = 2{,}0$ para todas as instâncias foi quando $\propto = 0{,}5$.

Figura 26 - Calibração de $\propto$ para $\beta = 2{,}0$



Com os valores de $\propto$ e β definidos em 0,5 e 2,0, respectivamente, foi testado o parâmetro ρ igual a 0,3 e comparado com os valores obtidos anteriormente ($\rho = 0{,}5$). O número de formigas por ciclo permaneceu em 150 e de ciclos em 100. Para cada instância (no eixo X), a Figura 27 mostra (no eixo Y) em verde a diferença percentual de FO entre meta-heurística e CPLEX *Dif. FO* para $\rho = 0{,}3$ e em preto para $\rho = 0{,}5$. Esse parâmetro representa a taxa de evaporação do feromônio entre seleções, ou seja, quanto maior o ρ, mais rápido o feromônio irá evaporar e menor será o tempo de influência de uma formiga nos resultados seguintes. Logo, quanto menor o ρ, menor a taxa de evaporação e maior o tempo de influência da formiga. Analisando o gráfico, a melhor taxa foi de 0,3, com resultados melhores em todas as instâncias. Na Instância 10, em que o valor de *Dif. FO* está negativo, o entendimento é de que o resultado médio obtido pela meta-heurística nessa etapa é 0,42% melhor que o do CPLEX. A mesma interpretação deve ser feita para a Instância 11.

Figura 27 - Comparação de valor de ρ



Estabelecidos os valores de $\propto$, β e ρ, foram realizados testes na Instância 11 para verificar o número mais adequado de formigas por ciclo e de ciclos. A instância foi definida por ser a que possui mais lotes e ter maior complexidade. A Tabela 11 mostra as combinações testadas, todas com os mesmos valores de $\propto$, β e ρ. A meta-heurística foi executada 10 vezes para cada combinação e a coluna *Dif. FO* mostra a diferença percentual entre a média das funções objetivo da meta-heurística e do CPLEX. A Combinação 1, $numFormigas = 150$ e $numCiclos = 100$, é a que estava sendo testada até o momento, e na Figura 27 obteve diferença percentual de -0,1% na Instância 11. Na Combinação 2 o número de formigas por ciclo foi reduzido, para avaliar se traria prejuízo à solução, e o resultado confirmou essa hipótese. Se a quantidade de formigas por ciclo for pequena, a quantidade de soluções geradas será menor, portanto, provavelmente pior. Quando o ciclo encerrar, os feromônios serão incrementados com base nessas soluções ruins encontradas, e isso irá direcionar o algoritmo a convergir para uma solução de baixa qualidade.

Tabela 11 - Combinações de N° de Formigas por Ciclo e N° de Ciclos - Instância 11

Combinação	N° de Formigas por Ciclo	N° de Ciclos	β	α	ρ	Dif. FO
1	150	100	2,0	0,5	0,3	-0,1%
2	100	100	2,0	0,5	0,3	0,8%
3	250	100	2,0	0,5	0,3	0,0%
4	150	150	2,0	0,5	0,3	-0,3%

Na terceira combinação, o número de formigas aumentou para 250, visando verificar se haveria algum ganho na solução. O resultado foi semelhante à Combinação 1, em que o número de formigas era 150, então foi dado preferência ao menor número de formigas, que demanda menor tempo de processamento (gera menos soluções). Observação semelhante foi feita por Li e Zhang (2013), em que a partir de determinada quantidade de formigas, o aumento desse parâmetro não resulta em ganho na qualidade da solução. A Combinação 4 utilizou 150 formigas por ciclo e aumentou o número de ciclos também para 150. Com isso, obteve o melhor resultado, -0,3%, ou seja, a meta-heurística foi 0,3% melhor que o CPLEX na Instância 11.

Combinações com mais de 150 ciclos não foram testadas, pois foi verificado que a meta-heurística já converge para uma única solução antes dos 150 ciclos, devido à influência dos feromônios. Isso significa que a partir de determinado ponto, todas as soluções geradas são iguais. Essa convergência é ilustrada na Figura 28. O Eixo X corresponde ao ciclo, a linha preta ao menor valor de FO gerado em cada ciclo, e a verde ao maior. Por volta do ciclo 130, o feromônio passa a ser tão forte em uma solução que faz o algoritmo convergir. Isso aconteceu para todas as outras instâncias utilizando os mesmos parâmetros, na Figura 29 é apresentada essa convergência para a Instância 9.

Figura 28 - Convergência da FO - Instância 11

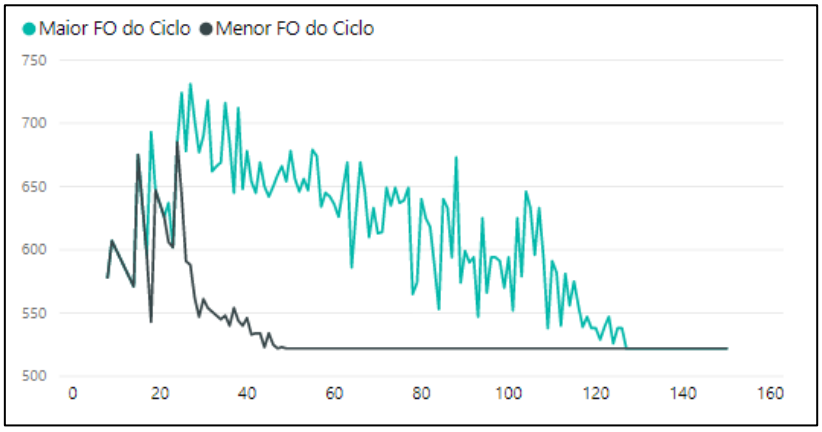
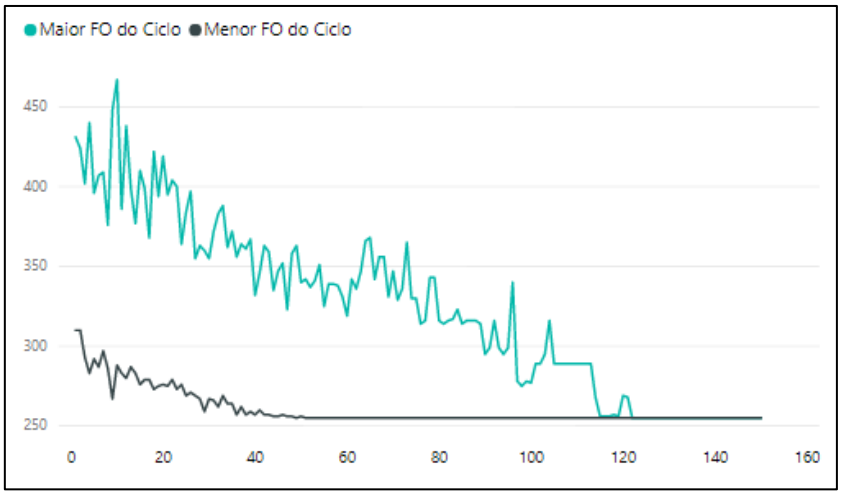


Figura 29 - Convergência da FO - Instância 9



Portanto, a combinação final de parâmetros utilizada para a obtenção de resultados dessa dissertação é: $numFormigas = 150$, $numCiclos = 150$, $\propto = 0{,}5$, $\beta = 2$ e $\rho = 0{,}3$. A seguir os resultados e as análises são apresentados.

5.2 Resultados e Análises

De acordo com Pimenta (2018), o modelo matemático foi resolvido com o solver IBM® ILOG® CPLEX® Versão 12.6 em computador com processador Intel® i7 com 16 GB de memória RAM. A meta-heurística proposta foi desenvolvida em linguagem C e executada em computador com as mesmas características que o utilizado pelo modelo matemático.

Foi definido um tempo limite para execução do CPLEX de 79.200 segundos, equivalente a 22 horas (PIMENTA, 2018). Para obter os resultados da meta-heurística, cada instância foi executada 15 vezes, possibilitando a verificação de sua estabilidade a partir da diferença entre os 15 resultados gerados.

A Tabela 12 exibe os resultados gerados pela meta-heurística e pelo CPLEX para cada instância indicada na coluna *Instância*. As colunas de 2 a 6 contêm informações sobre o desempenho da meta-heurística. *FO Média* é a média da função objetivo das 15 execuções da instância; *Desv. Pad. Rel. FO* indica o desvio padrão relativo da Função objetivo, calculado como a divisão do desvio padrão da FO pela média. A coluna *Menor FO* apresenta o menor valor encontrado nas 15 execuções; *Tempo Exec. (s)* mostra (em segundos) a quantidade de tempo média que o algoritmo demandou em toda a execução, e *Tempo p/ Melhor (s)*, a quantidade de tempo média (em segundos) que levou para achar a melhor solução.

Tabela 12 - Resultados Meta-heurística proposta x CPLEX

Instância	ACO					CPLEX					Diferença	
	FO Média	Desv. Pad. Rel. FO	Melhor FO	Tempo Exec. (s)	Tempo p/ Melhor (s)	FO	LB	UB	GAP	Tempo Exec. (s)	Dif. FO	Dif. Tempo Exec.
1	350,0	0,0%	350,0	47,5	4,1	350,0	-	-	0,00%	76	0,00%	-37,5%
2	255,9	0,1%	255,0	19,3	4,7	255,0	-	-	0,00%	105	0,37%	-81,7%
3	331,0	0,0%	331,0	23,9	8,0	331,0	-	-	0,00%	74	0,00%	-67,7%
4	390,0	0,0%	390,0	37,2	13,1	390,0	-	-	0,00%	81	0,00%	-54,1%
5	390,5	0,1%	390,0	31,5	11,4	390,0	-	-	0,00%	409	0,14%	-92,3%
6	269,9	0,4%	268,0	27,1	10,2	268,0	-	-	0,00%	1933	0,72%	-98,6%
7	253,9	0,4%	252,0	33,8	20,4	252,0	-	-	0,00%	8315	0,77%	-99,6%
8	260,7	0,2%	260,0	38,3	14,9	260,0	-	-	0,00%	1747	0,28%	-97,8%
9	254,9	0,2%	253,0	30,0	11,2	252,0	-	-	0,00%	5335	1,16%	-99,4%
10	364,5	0,4%	362,0	44,5	25,6	367,0	350,2	367,0	4,58%	79200	-0,67%	-99,9%
11	519,0	0,8%	512,0	91,6	39,3	521,0	427,5	521,0	17,94%	79200	-0,38%	-99,9%
12	168,0	0,0%	168,0	33,5	16,5	168,0	-	-	0,00%	74	0,00%	-54,7%
13	211,8	0,2%	211,0	28,3	10,6	209,0	-	-	0,00%	656	1,34%	-95,7%

As colunas de 7 a 11 contêm informações do desempenho do CPLEX. *FO* exibe a função objetivo encontrada pelo modelo; *LB* e *UB* são, respectivamente, *Lower Bound* e *Upper Bound* da solução e não são exibidos quando o CPLEX encontra a solução ótima, pois são iguais à própria FO. A coluna *GAP* é maior que 0 quando o modelo não encontra o ótimo, sendo calculado como $(UB - LB)/UB$; *Tempo Exec. (s)*, coluna 11, mostra o tempo que o CPLEX levou para encontrar a solução ótima, porém limitado a 79200 segundos (22 horas). As colunas 12 e 13 mostram comparações entre os resultados da meta-heurística proposta e do CPLEX. *Dif. FO* é a diferença percentual média entre as funções objetivo, calculado como: a diferença entre a FO Média da meta-heurística e a FO do CPLEX, dividida pela FO do CPLEX. Portanto, quando o valor é positivo, o CPLEX teve melhor desempenho, caso contrário, a meta-heurística foi melhor, e quando igual a zero, ambas chegaram ao mesmo resultado. O mesmo raciocínio se aplica à *Dif. Tempo Exec.*, que indica a diferença percentual entre o tempo de execução da meta-heurística e do CPLEX.

Analisando os resultados, observa-se que a meta-heurística é bastante estável, com desvio padrão relativo médio de 0,2% e máximo de 0,8% na maior instância, Instância

11. Em relação à função objetivo, em 4 instâncias (Instâncias 1, 3, 4 e 12) a meta-heurística obteve as 15 soluções ótimas, resultando em diferença percentual média de FO igual a 0. Em 5 instâncias (Instâncias 2, 5, 6, 7 e 8), a meta-heurística encontrou a solução ótima, porém isso não aconteceu em todas as execuções, gerando diferença percentual média abaixo de 1%.

Em duas instâncias, a meta-heurística obteve diferença percentual acima de 1%, a Instância 9 (1,16%) e a Instância 13 (1,34%), sendo a diferença em valor absoluto igual a 2,9 e 2,8 respectivamente. O valor possui casa decimal por ser a média, porém, o tempo é discretizado, e cada unidade de tempo representa 30 minutos. Multiplicando a maior diferença (2,9) pelo tempo que cada unidade representa (30 minutos) e dividindo pela quantidade de lotes da Instância (9 lotes em ambas), temos o valor de 9,67, ou seja, um atraso de cerca de 10 minutos por lote, que para uma operação de terminal ferroviário não causa um grande impacto. Considerando as melhores soluções geradas para essas instâncias, a diferença cai, a melhor FO da Instância 9 foi 253, uma unidade de tempo a mais que a FO do CPLEX, e a melhor da Instância 11 foi 211, duas unidades a mais que o CPLEX.

Nas Instâncias 10 (12 lotes) e 11 (15 lotes), que representam cenários maiores, o CPLEX não encontrou a solução ótima, apresentando GAP’s de 4,58% e 17,94%, respectivamente. A meta-heurística encontrou FO Média abaixo do CPLEX nas duas, com diferença média percentual de -0,67% (Instância 10) e -0,38% (Instância 11). Considerando as melhores soluções encontradas dentre as 15 execuções, a diferença vai para -1,36% e -1,73%.

Em relação ao tempo de execução das Instâncias, a execução da meta-heurística proposta é extremamente mais rápida que o CPLEX, com uma média de 37,4 segundos. Na instância mais simples, Instância 1, a meta-heurística foi 37,5% mais rápida que o CPLEX, e em 8 das instâncias, a diferença percentual é acima de 90%. Para as instâncias mais complexas, em que o CPLEX não encontrou a solução ótima, o tempo total de execução foi de 22 horas, enquanto para a meta-heurística foi de 44,5 segundos (Instância 10) e 91,6 segundos (Instância 11), uma ordem de grandeza completamente diferente.

A Tabela 13 mostra o valor da diferença percentual entre a função objetivo da meta-heurística e do CPLEX, ressaltando que valores positivos indicam vantagem do CPLEX e negativos da meta-heurística. Os resultados das execuções de cada instância estão ordenados de forma crescente, portanto as melhores soluções encontradas pela meta-heurística para cada instância estão na execução número 1.

Tabela 13 - Diferença (%) de FO CPLEX x Meta-heurística proposta por Execução de cada Instância

Instância	Execução 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Total**
1	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,00**
2	0,00	0,39	0,39	0,39	0,39	0,39	0,39	0,39	0,39	0,39	0,39	0,39	0,39	0,39	0,39	**0,37**
3	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,00**
4	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,00**
5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,26	0,26	0,26	0,26	0,26	0,26	0,26	0,26	**0,14**
6	0,00	0,00	0,37	0,37	0,75	0,75	0,75	0,75	0,75	0,75	1,12	1,12	1,12	1,12	1,12	**0,72**
7	0,00	0,00	0,40	0,79	0,79	0,79	0,79	0,79	0,79	0,79	0,79	1,19	1,19	1,19	1,19	**0,77**
8	0,00	0,00	0,00	0,00	0,38	0,38	0,38	0,38	0,38	0,38	0,38	0,38	0,38	0,38	0,38	**0,28**
9	0,40	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,59	**1,16**
10	-1,36	-1,09	-1,09	-0,82	-0,82	-0,82	-0,82	-0,82	-0,54	-0,54	-0,54	-0,27	-0,27	-0,27	0,00	**-0,67**
11	-1,73	-1,54	-1,15	-0,96	-0,77	-0,77	-0,58	-0,38	-0,38	0,19	0,19	0,38	0,38	0,58	0,77	**-0,38**
12	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,00**
13	0,96	0,96	0,96	1,44	1,44	1,44	1,44	1,44	1,44	1,44	1,44	1,44	1,44	1,44	1,44	**1,34**

Nessa visão, é possível verificar o desempenho da meta-heurística em cada execução individual. Na Instância 2, por exemplo, encontrou a solução ótima, porém em uma das 15 execuções, todas as outras foram 1 unidade de tempo acima da ótima, gerando a diferença de 0,39%. Já na Instância 5, 40% das execuções encontraram o ótimo e 60% encontraram uma FO com diferença de 0,36% para a do CPLEX, o que também representa 1 unidade de tempo a mais. Na Instância 9, a melhor solução encontrada dentre as 15 execuções foi 0,4% pior que a gerada pelo CPLEX, enquanto a maioria foi 1,19% e a última chegou a 1,59%.

Na Instância 10 a meta-heurística obteve 14 resultados melhores que a solução do CPLEX, e 1 resultado igual. Esses resultados melhores variaram entre -0,27% e -1,36%. Nas execuções da Instância 11, em 60% a meta-heurística foi melhor e em 40% foi pior que o CPLEX. Quando foi melhor, a diferença percentual variou entre -0,38% e -1,73%, e quando pior, variou entre 0,19% e 0,77%. A execução com pior resultado da meta-heurística, ou seja, com a maior diferença percentual positiva, foi a execução 15 da Instância 9, com 1,59%.

A seguir, serão exibidos alguns gráficos para demonstrar o comportamento da meta-heurística ao longo de uma execução para algumas instâncias. A Figura 30 mostra a evolução da FO global por ciclo para uma única execução da Instância 4, ou seja, indica em qual ciclo a meta-heurística encontrou uma nova melhor solução. A Figura 31 mostra a quantidade de soluções inviáveis geradas e descartadas por ciclo. O mesmo entendimento deve ser considerado para os gráficos das próximas instâncias.

Figura 30 - FO Global por ciclo - Instância 4

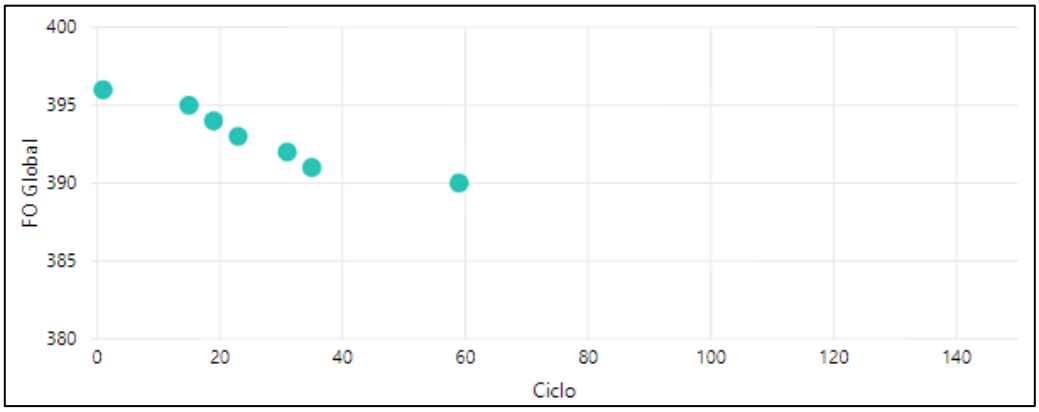
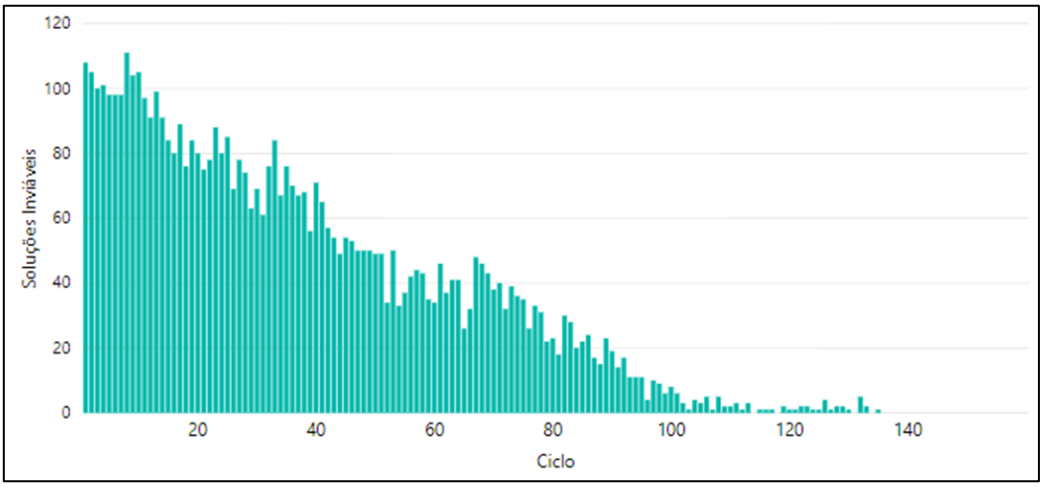
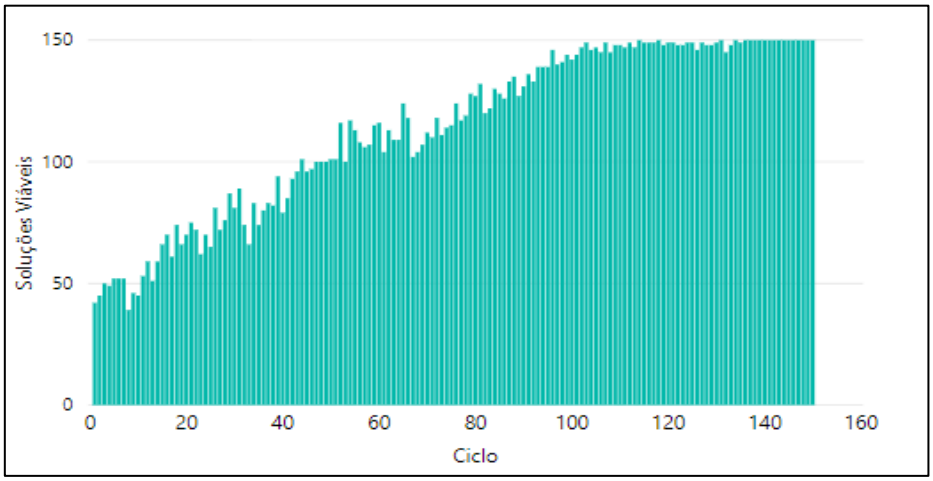


Figura 31 - Soluções inviáveis por Ciclo - Instância 4



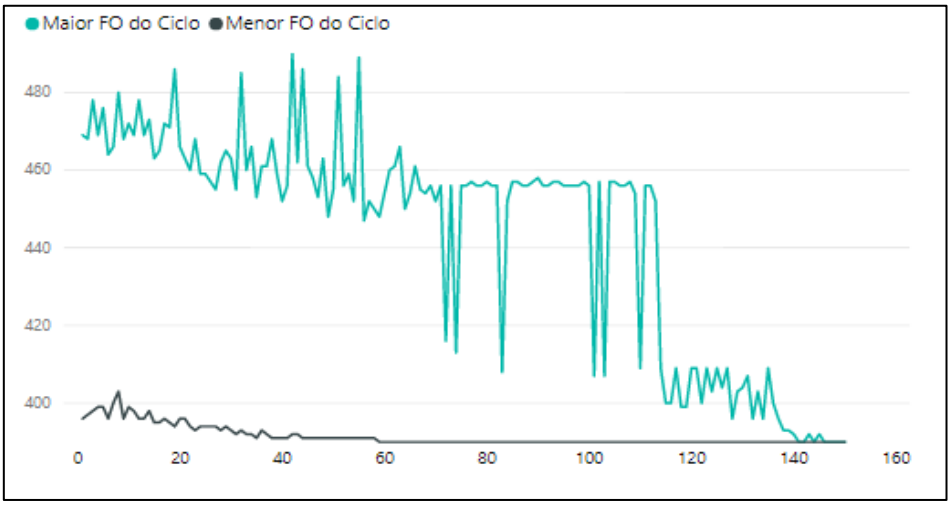
Na Figura 30, a primeira solução gerada já foi muito próxima ao ótimo, que é 390, e a evolução aconteceu com pequenas melhorias até a solução ótima, no ciclo 59, com um total de soluções globais encontradas de 7. Na Figura 31, percebe-se que o número de soluções inviáveis geradas no início é grande e vai diminuindo na medida em que o algoritmo avança e converge, o total de inviabilidades foi de 5.388 nessa execução. A Figura 32 mostra o número de soluções viáveis por ciclo, portanto, é complementar ao gráfico de soluções inviáveis, a soma dos dois valores por ciclo resulta em 150, o número de formigas por ciclo.

Figura 32 - Soluções viáveis por ciclo - Instância 4



A convergência da Instância 4 pode ser vista na Figura 33, em que a linha verde indica a maior FO gerada no ciclo e a linha preta indica a menor. Mesmo com a solução ótima definida no ciclo 59, novas soluções são geradas até quase o final, convergindo no ciclo 146, por meio da influência dos feromônios.

Figura 33 - FO mínima e máxima por Ciclo - Instância 4



A Figura 34 exibe o comportamento da Instância 8. Em comparação com a Instância 4, a solução global final foi definida antes, no ciclo 43, apesar de que a solução inicial encontrada foi ruim. Pode-se perceber que a melhoria das soluções globais nos primeiros ciclos foi muito agressiva, possibilitando encontrar a solução ótima em menos ciclos. Um total de 16 soluções globais foram encontradas nessa instância. Na Figura 35, o número de soluções inviáveis por ciclo da Instância 8 é apresentado, sendo quase 140 no início, mas igual a 0 a partir do ciclo 120, com um total de 5.584 soluções descartadas.

Figura 34 - FO Global por ciclo - Instância 8

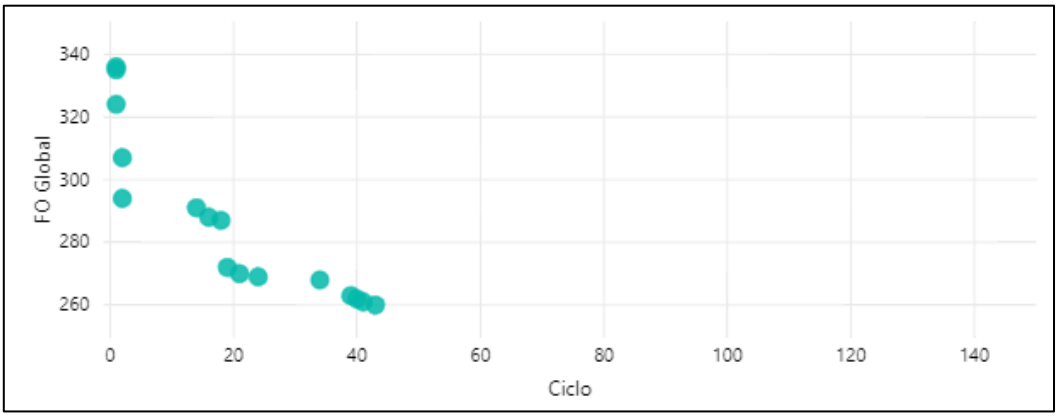
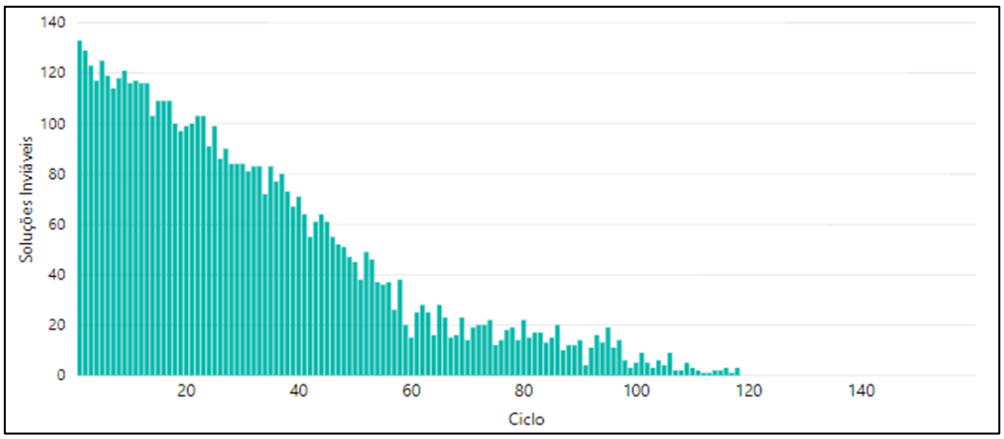
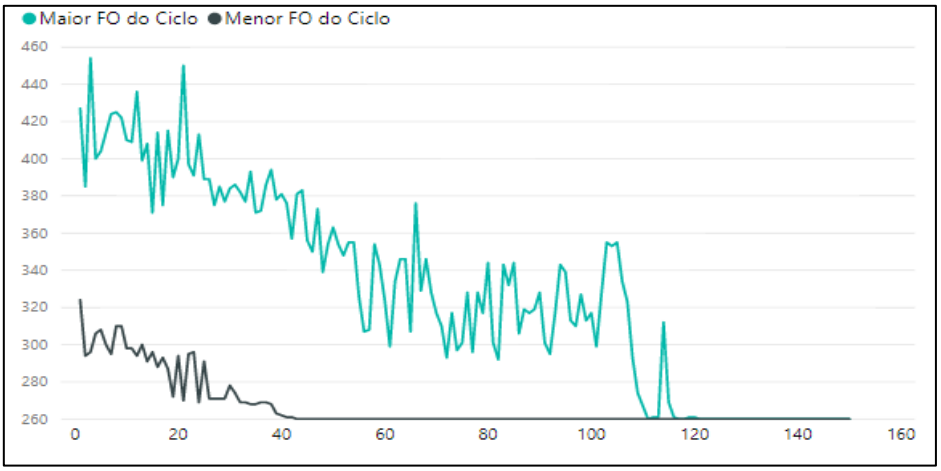


Figura 35 - Soluções Inviáveis por Ciclo - Instância 8



A Figura 36 mostra a FO mínima e máxima por ciclo para a Instância 8. Analisando o gráfico, percebe-se que a convergência de soluções geradas também foi mais rápida para a instância 8 que para a Instância 4, acontecendo no ciclo 121. Isso implica também em parar de gerar soluções inviáveis antes, perto do período de convergência.

Figura 36 - FO mínima e máxima por ciclo - Instância 8



A Figura 37 apresenta o comportamento da Instância 11, que possui 15 lotes. Nessa instância o CPLEX encontrou um GAP de 17,94%, rodando por 22 horas e a meta-heurística obteve FO até 1,73% melhor, em tempo abaixo de 2 minutos. O número de soluções inviáveis foi muito alto nessa execução, com um total de 7.919 soluções descartadas, Figura 38. A primeira solução viável encontrada pela meta-heurística nessa execução foi no ciclo 18, após a geração de 2.551 formigas, o que acontece devido à quantidade de restrições desse problema combinadas a essa instância. E só por volta do ciclo 30 que a base de feromônios passou a ter maior influência e o número de inviabilidades foi reduzir, isso pode ser concluído na Figura 38, em que até por volta do ciclo 30, quase 150 soluções foram descartadas por ciclo.

Figura 37 - FO Global por ciclo - Instância 11

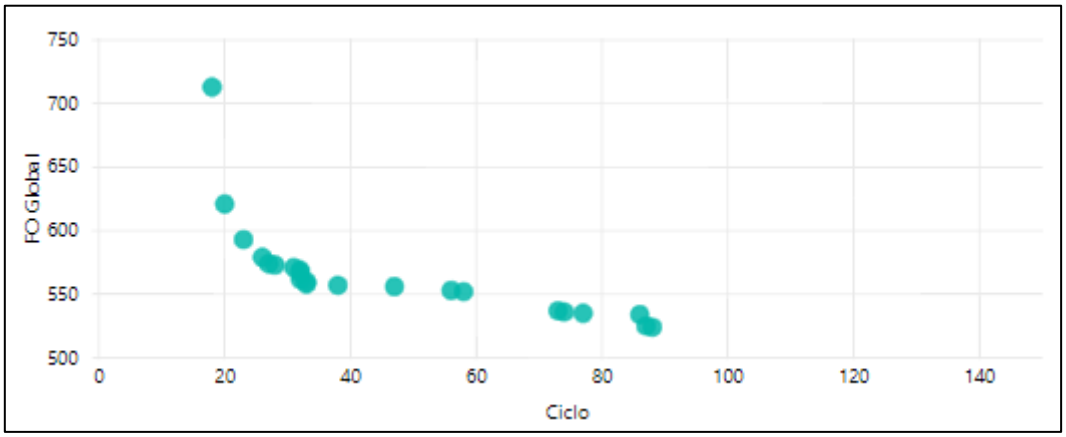
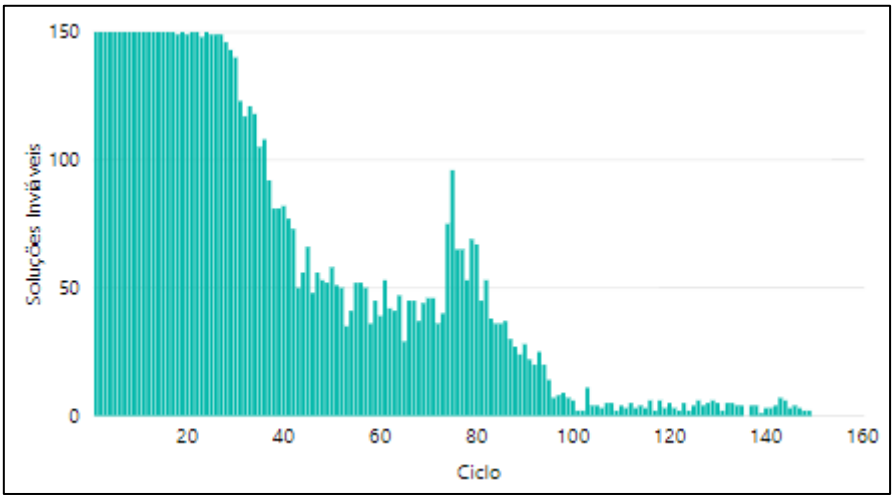


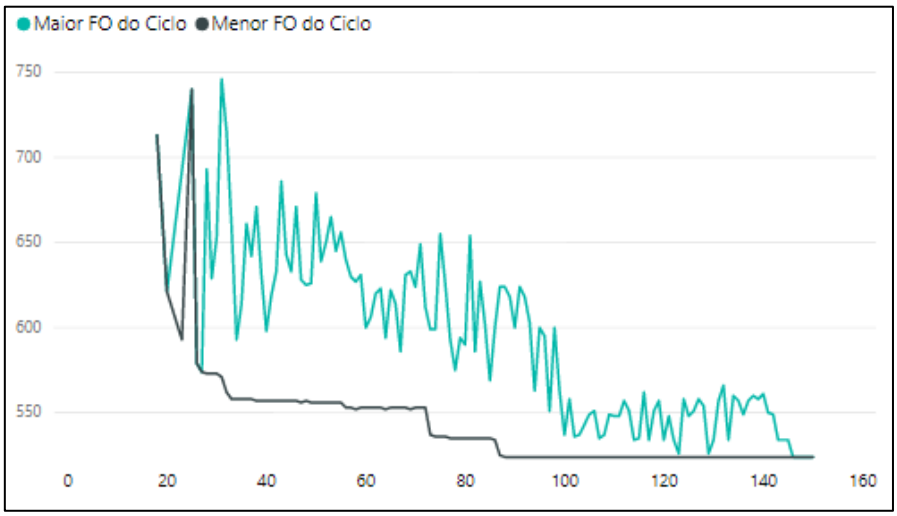
Figura 38 - Soluções Inviáveis por Ciclo - Instância 11



É possível observar na Figura 39 a dificuldade de encontrar soluções para a Instância 11. Logo no início, a linha preta, que representa a menor FO encontrada no ciclo, tem um pico com FO igual a 740 no ciclo 25, sendo que já havia encontrado soluções de até

593 antes, no ciclo 23. Ou seja, foram 150 soluções do ciclo 25 que não encontraram nenhuma FO abaixo de 740, mesmo com a influência dos feromônios das formigas anteriores. Nas demais instâncias não houve um pico tão grande da menor FO encontrada no ciclo.

Figura 39 - FO máxima e mínima - Instância 11



A Figura 40 mostra o comportamento da Instância 13, em que a meta-heurística não chegou à solução ótima encontrada pelo CPLEX. O número de soluções inviáveis começou bastante alto, perto de 150 no primeiro ciclo, e totalizando 5.914 na execução (Figura 41), porém caiu rapidamente.

Figura 40 - FO Global por ciclo - Instância 13

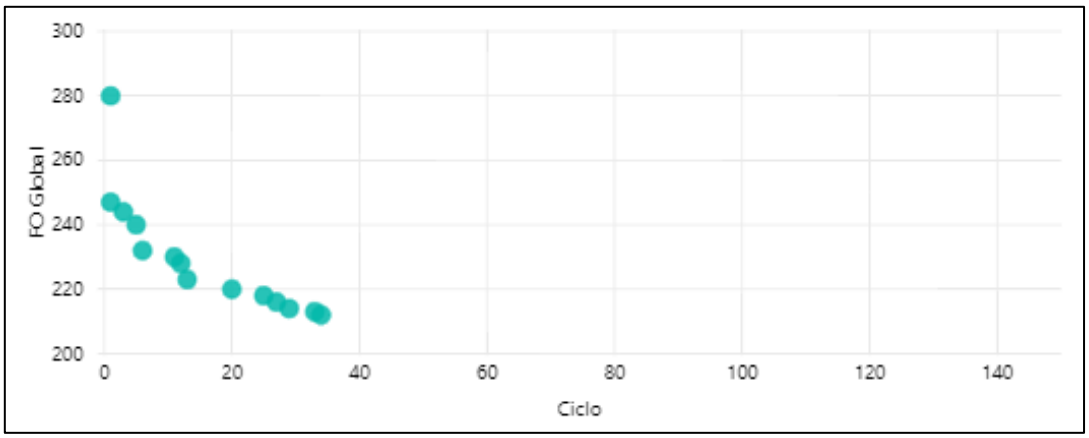
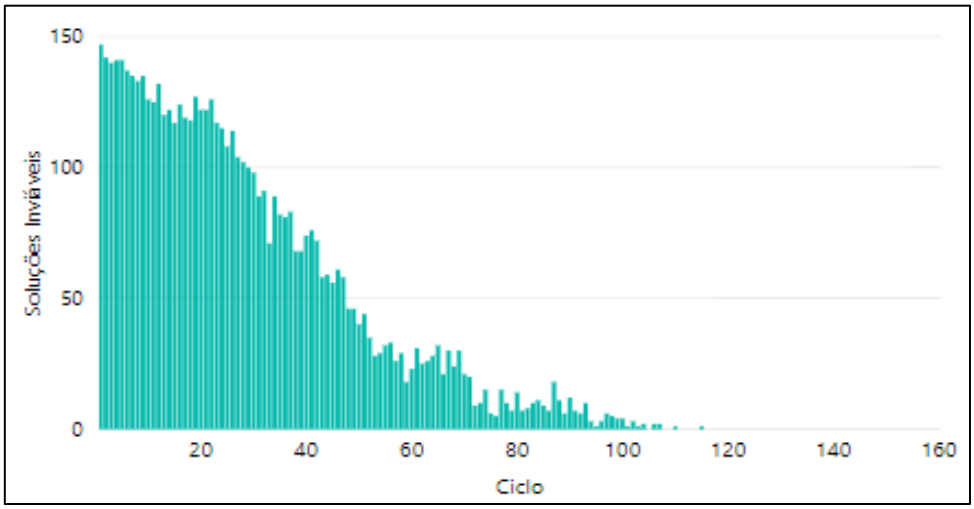
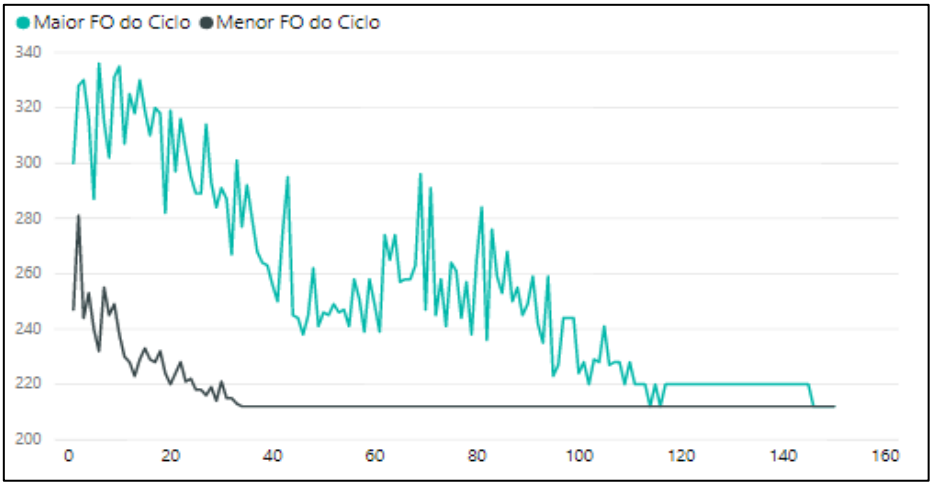


Figura 41 - Soluções Inviáveis por Ciclo - Instância 13



A Figura 42 mostra a FO mínima e a máxima por ciclo na Instância 13. Apesar da convergência das soluções pelos feromônios, foram geradas soluções diferentes até o ciclo 140. Um ponto importante a se observar é que todas as soluções globais foram encontradas com certa antecedência, o que pode significar que seria possível diminuir o número de ciclos sem perda aos resultados. Como a meta-heurística já possui um tempo baixo de processamento, e principalmente, como algumas soluções só convergem próximas ao ciclo 150, foi decidido por manter esse parâmetro. Para as execuções e instâncias apresentadas, as soluções geradas próximas do ciclo 150 não se tornaram soluções globais, mas isso não significa que em diferentes execuções ou diferentes instâncias, isso não possa acontecer.

Figura 42 - FO mínima e máxima por ciclo - Instância 13



Analisando o comportamento das execuções em diferentes instâncias, percebe-se a interferência da complexidade da instância na convergência da FO e na viabilidade das soluções geradas. Mesmo assim, a meta-heurística é estável em suas execuções, consegue alcançar resultados ótimos, ou próximos ao ótimo em menos de 1 minuto, e para instâncias maiores, levou menos de 2 minutos para chegar a soluções melhores que as encontradas pelo CPLEX, que executou por 22 horas. Isso torna a meta-heurística proposta bastante indicada para o uso prático, na dinâmica das operações dos terminais ferroviários, além de poder ser utilizada no replanejamento de atividades, motivado por mudanças inesperadas nas restrições, como quebra de equipamentos.

6. CONCLUSÕES

Esse trabalho propôs uma meta-heurística baseada na *Ant Colony Optimization* (ACO) aplicada ao planejamento de atividades de lotes de vagões em terminais ferroviários, visando minimizar o tempo de estadia dos lotes no terminal, e considerando múltiplos modos de execução, manutenção programada, tempos de setup e retenção de recursos.

A meta-heurística proposta foi avaliada em relação aos resultados obtidos pelo modelo matemático da literatura. Conseguiu resolver ou se aproximar da solução ótima nas instâncias até 9 lotes, e chegou a resultados melhores que o modelo matemático nas instâncias de 12 e 15 lotes. O tempo de processamento demandado pela meta-heurística foi extremamente baixo, levando sempre menos de 2 minutos por execução, até mesmo na instância de 15 lotes. Além disso, mostrou um desempenho estável, com um desvio padrão relativo de no máximo 0,8%.

Considerando o *trade-off* entre tempo de execução e qualidade da solução, a meta-heurística é bastante atraente, trazendo resultados satisfatórios em menos de 2 minutos de execução, em comparação com o modelo matemático que traz resultados próximos, mas rodando por horas, chegando até 22 horas, dependendo da instância. Portanto, a meta-heurística seria uma boa ferramenta de suporte e otimização do planejamento de atividades de lotes nos terminais.

A meta-heurística proposta pode ser aplicada ao planejamento de atividades, não só em terminais, mas também em outros tipos de pátios ferroviários. Devido ao seu curto tempo de execução, pode ser utilizada também no replanejamento de atividades, motivado por mudanças inesperadas nas restrições, como quebra de equipamentos.

Como recomendação de trabalhos futuros, é proposto o desenvolvimento de meta-heurísticas com novas abordagens para o tratamento da característica de retenção de recursos e comparação de resultados com a abordagem utilizada nessa dissertação. Além disso, a criação de uma heurística pós-otimização para buscar pequenas melhorias a serem realizadas nas soluções encontradas pela meta-heurística, visando melhores resultados.

REFERÊNCIAS

ABDOLSHAH, Mohammad. A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions. **International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies**, v. 5, n. 4, p. 253-286, 2014.

AFSHAR-NADJAFI, Behrouz; MAJLESI, Mahyar. Resource constrained project scheduling problem with setup times after preemptive processes. **Computers & Chemical Engineering**, v. 69, p. 16-25, 2014.

AL-SHIHABI, Sameh T.; ALDURGAM, Mohammad M. A max-min ant system for the finance-based scheduling problem. **Computers & Industrial Engineering**, v. 110, p. 264-276, 2017.

ASSOCIAÇÃO NACIONAL DOS TRANSPORTADORES FERROVIÁRIOS. **Informações Gerais**. Disponível em: <http://www.antf.org.br/informacoes-gerais>, Acesso em: 8 de Fevereiro de 2019.

BARRIOS, Agustín; BALLESTÍN, Francisco; VALLS, Vicente. A double genetic algorithm for the MRCPSP/max. **Computers & Operations Research**, v. 38, n. 1, p. 33-43, 2011.

BEŞIKCI, Umut; BILGE, Ümit; ULUSOY, Gündüz. Multi-mode resource constrained multi-project scheduling and resource portfolio problem. **European Journal of Operational Research**, v. 240, n. 1, p. 22-31, 2015.

BLAZEWICZ, Jacek; LENSTRA, Jan Karel; KAN, AHG Rinnooy. Scheduling subject to resource constraints: classification and complexity. **Discrete Applied Mathematics**, v. 5, n. 1, p. 11-24, 1983.

BULLNHEIMER, Bernd; HARTL, Richard F.; STRAUSS, Christine. A new rank based version of the Ant System. **A computational study**. 1997.

CHEN, Wei-Neng et al. Optimizing discounted cash flows in project scheduling—An ant colony optimization approach. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 40, n. 1, p. 64-77, 2010.

CHEN, Wei-Neng; ZHANG, Jun. Ant colony optimization for software project scheduling and staffing with an event-based scheduler. **IEEE Transactions on Software Engineering**, v. 39, n. 1, p. 1-17, 2013.

CHENG, Junzilan et al. Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. **Computers & Operations Research**, v. 53, p. 275-287, 2015.

CHIANG, Chuan-Wen; HUANG, Yu-Qing; WANG, Wen-Yen. Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. **Journal of Intelligent & Fuzzy Systems**, v. 19, n. 4, 5, p. 345-358, 2008.

DORIGO, M.; STÜTZLE, T. **Ant Colony Optimization**. USA: MIT Press, 2004.

HARTMANN, Sönke; BRISKORN, Dirk. A survey of variants and extensions of the resource-constrained project scheduling problem. **European Journal of operational research**, v. 207, n. 1, p. 1-14, 2010.

LI, Heng; ZHANG, Hong. Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. **Automation in construction**, v. 35, p. 431-438, 2013.

MERKLE, Daniel; MIDDENDORF, Martin; SCHMECK, Hartmut. Ant colony optimization for resource-constrained project scheduling. **IEEE transactions on evolutionary computation**, v. 6, n. 4, p. 333-346, 2002.

NING, Minjing et al. Metaheuristics for multi-mode cash flow balanced project scheduling with stochastic duration of activities. **Automation in Construction**, v. 81, p. 224-233, 2017.

PERRETTO, Mauricio; LOPES, Heitor Silvério. Reconstruction of phylogenetic trees using the ant colony optimization paradigm. **Genetics and Molecular Research**, v. 4, n. 3, p. 581-589, 2005.

PIMENTA, L. B., **Planejamento da sequência das atividades e da utilização dos recursos na operação de lotes de vagões em terminais ferroviários**. 2018. Dissertação (Mestrado em Engenharia Civil na Área de Transportes) – Universidade Federal do Espírito Santo, Vitória.

RAHIMI, Amir; KARIMI, Hamid; AFSHAR-NADJAFI, Behrouz. Using meta-heuristics for project scheduling under mode identity constraints. **Applied Soft Computing**, v. 13, n. 4, p. 2124-2135, 2013.

ROSA, R. A. **Operação Ferroviária**. Rio de Janeiro: Ed. LTC, v.01, 2016.

SABINO, Jodelson A. et al. A multi-objective ant colony optimization method applied to switch engine scheduling in railroad yards. **Pesquisa Operacional**, v. 30, n. 2, p. 486-514, 2010.

TAVANA, Madjid; ABTAHI, Amir-Reza; KHALILI-DAMGHANI, Kaveh. A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems. **Expert Systems with Applications**, v. 41, n. 4, p. 1830-1846, 2014.

TRITSCHLER, Martin; NABER, Anulark; KOLISCH, Rainer. A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles. **European Journal of Operational Research**, v. 262, n. 1, p. 262-273, 2017.

VAN PETEGHEM, Vincent; VANHOUCKE, Mario. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. **European Journal of Operational Research**, v. 201, n. 2, p. 409-418, 2010.

WANG, Yanting et al. On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem. **Computers & Industrial Engineering**, v. 114, p. 223-234, 2017.