UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÕES

RICARDO HEIMOSKI

# USO DO *ANT COLONY OPTIMIZATION* COMO SOLUÇÃO PARA O PROBLEMA DE DISCAGEM PARA CARONAS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2019

RICARDO HEIMOSKI

# USO DO *ANT COLONY OPTIMIZATION* COMO SOLUÇÃO PARA O PROBLEMA DE DISCAGEM PARA CARONAS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do curso de Bacharelado em Sistemas de Informação da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador:Prof. Dr.Marcelo Mikosz Gonçalves

CURITIBA

2019

## TERMO DE APROVAÇÃO

# “USO DO ANT COLONY OPTIMIZATION COMO SOLUÇÃO PARA O PROBLEMA DE DISCAGEM PARA CARONAS”

por

## “Ricardo Heimoski”

Este Trabalho de Conclusão de Curso foi apresentado no dia 09 de Agosto de 2019 como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Curitiba. O(a)(s) aluno(a)(s) foi(ram) arguido(a)(s) pelos membros da Banca de Avaliação abaixo assinados. Após deliberação a Banca de Avaliação considerou o trabalho _____.

| | |
|---|---|
| _____<br>**Prof. Marcelo Mikosz Gonçalves**<br>(Presidente - UTFPR/Curitiba) | _____<br>**Profa. Myriam Regattieri de Biase da Silva Delgado**<br>(Avaliador(a) 1 - UTFPR/Curitiba) |
| _____<br>**Prof. João Alberto Fabro**<br>(Avaliador 2(a) - UTFPR/Curitiba) | _____<br>**<Profa. Leyza Baldo Dorini>**<br>(Professora Responsável pelo TCC – UTFPR/Curitiba) |
| _____<br>**Prof. Marcelo Mikosz Gonçalves**<br>(Coordenadordo curso de Bacharelado em Sistemas de Informação – UTFPR/Curitiba) | |

“A Folha de Aprovação assinada encontra-se na Coordenação do Curso.”

# RESUMO

HEIMOSKI, Ricardo. **Uso do *Ant Colony Optimization* como Solução para o Problema de Discagem para Caronas.** 2019. 60 f. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

Este trabalho apresenta uma solução para um modelo do problema de discagem para caronas através do uso da meta-heurística baseada no comportamento das formigas, conhecida como *Ant Colony Optimization* ou ACO. Sabendo-se que vários algoritmos baseados em comportamentos da natureza são utilizados para resolução de problemas em grafos, e que problemas de locomoção urbana podem ser modelados como problemas em grafos, percebeu-se a possibilidade da utilização de uma técnica dessas para a aproximação de um problema real relacionado à discagem de caronas. Foi utilizada uma abordagem extremamente elitista que somente atualiza com feromônio a melhor solução encontrada. Com a solução proposta foi possível resolver o modelo do problema proposto para instâncias de tamanhos variados, com melhorias na qualidade e velocidade da resposta com iterações sobre resultados anteriores seguindo a meta-heurística. Os resultados mostram a importância da calibragem da parametrização a ser utilizada, pois impacta diretamente na qualidade das soluções e na velocidade das mesmas. Conclui-se que a meta-heurística de ACO pode ser utilizada para resolução desse tipo de problema e que trabalhos posteriores podem buscar otimizações sobre esse modelo de resolução.

**Palavras-chave**: *Ant, Colony, Optimization*, Formiga, ACO, DARP, Carona, *Ride*, Meta-heurística.

# ABSTRACT

HEIMOSKI, Ricardo. **Using Ant Colony Optimization as a Solution to the Carpool Dialing Problem**. 2019. 60 p. Course Completion Work (Undergraduate in Information Systems) – Federal Technological University of Paraná. Curitiba, 2019.

This work proposes a solution for a model of the dial a ride problem through the use of the metaheuristic based on ant's behavior, Ant Colony Optimization or ACO. From the observation that several algorithms based on behaviors found in nature are used to solve problems in graphs, and that problems of urban locomotion can be modeled as problems in graphs, the possibility of using such a technique to approximate a real problem was perceived. An extremely elitist approach was used, which only updates with pheromone the best solution found. With the proposed solution it was possible to solve the model problem for instances of varied sizes, with improvements in the quality and speed of the response with iterations on previous results following the metaheuristic. The results show how the calibration of the parameterization directly impacts the quality and speed of the solution. We concluded that the metaheuristic can be used to solve this type of problem and that later works look optimizations on this resolution model.

**Keywords:** Ant, Colony, Optimization, Formiga, ACO, DARP, Carona, Ride, Meta-heurística.

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ACO | *Ant Colony Optimization* |
| DARP | *Dial A Ride Problem* |
| CPP | *Car Pooling Problem* |
| DCPP | *Daily Car Pooling Problem* |
| LTCP | *Long Term Car Pooling Problem* |

# SUMÁRIO

# 1 INTRODUÇÃO

A urbanização com excessiva densidade populacional traz desafios com relação à locomoção nos espaços urbanos. Várias soluções coexistem em diferentes modais de transporte como bicicletas, metros, ônibus, e em um módulo extremamente comum que é o carro. Uma série de motivos culturais garantem um grande foco nesse modal de transporte que frequentemente é utilizado individualmente por grande parte da população. Somado a isso há a capacidade limitada nas vias de transporte. Diante desses problemas, é cada vez mais comum que sejam geradas situações de intenso congestionamento no trânsito dos grandes centros urbanos. Várias alternativas e vantagens existem para incentivar o uso compartilhado de carros em vez de seu uso individual, como *carpool*, que são as vias exclusivas para as pessoas que não estão sozinhas no veículo e a divisão de gastos de transporte. É comum o uso de táxis ou de empresas privadas para efetuar transporte de pessoas nas mais diversas localidades do planeta, frequentemente de forma individual, consistindo apenas do motorista e do cliente. Todavia existe uma variação em que o transporte apesar de ser feito em carro privado é coletivo, ou seja, várias pessoas com diferentes rotas compartilham o mesmo veículo, e as principais vantagens são, normalmente, a redução dos custos da viagem, do número de veículos nas ruas e, consequentemente, uma pequena melhoria no trânsito que deve se intensificar à medida que essa prática se torne mais comum (Moura, Rodrigues, 2013).

Com o surgimento de empresas que exploram esse tipo de serviço fazendo uso de tecnologias que permitam que os passageiros solicitem seu transporte em qualquer lugar de uma cidade, há também a necessidade de criar soluções computacionais para gerar os mapas de rotas otimizadas, para que assim, motoristas saibam quais são os melhores caminhos para que a adição de novos passageiros, com destinos distintos, não cause impactos negativos nos passageiros atuais.

Esse cenário, em que um carro deve ser responsável pelo embarque e transporte de passageiros até seus destinos, é frequentemente conhecido como Problemas de Carona ou “*Car Pooling Problem*” (CPP). Existem inúmeros trabalhos

na literatura de pesquisa computacional, buscando gerar soluções mais próximas das ótimas para o CPP (Hartman et al. 2014). Esses problemas podem ser subdivididos em subtipos de acordo com suas características que podem variar consideravelmente dependendo dos elementos disponíveis, dos padrões e dos objetivos dos participantes. Todos esses problemas são uma variação do problema do roteamento de carros, que é por si uma variação do problema do caixeiro viajante (Hartman et al. 2014).

Na história da computação o caixeiro viajante é um problema clássico, com soluções propostas de longa data (Shen, 1965), que nos remete a uma situação semelhante em termos gerais, de criar uma rota dentre várias possíveis para gerar um ciclo hamiltoniano, ou seja, uma rota ou ciclo em que todos os vértices de um grafo são visitados utilizando a menor rota possível.

Problemas de carona podem ser relacionados ao caixeiro viajante parcialmente por pertencerem a mesma ordem de complexidade, são problemas NP–completos (Yan, Chen 2011). Nos problemas de carona , a partir de um vértice em um grafo, tem-se que escolher quais são os vértices que devem ser visitados para maximizar a capacidade do veículo, ao mesmo tempo em que se busca minimizar o tempo de espera e desvios de rotas para os passageiros, pois para cada novo passageiro adicionado ao veículo tem-se o compromisso de chegar ao seu vértice de destino.

Na literatura científica existe uma classe de problemas relacionados ao cenário de caronas e suas características e nomenclaturas podem variar de acordo com seus elementos. O nome genérico de CPP é utilizado para representar a classe geral do problema, e muitas vezes o termo é usado livremente sem especificar o subtipo do problema que está sendo trabalhado. É importante salientar que existem pequenas diferenças entre esses problemas derivadas de seus elementos fundamentais.

O problema no qual uma pessoa liga para um serviço de caronas que pode lhe fornecer um veículo qualquer, e no qual todos os veículos são iguais, é frequentemente conhecido como problemas de discagem para caronas (DARP - *Dial a Ride Problem*) (Cordeau, Laporte, 2007). É também comum a aplicação de algum mecanismo de escolha prévia pelos participantes relacionados à carona, como

preferências musicais, em que a formação de grupos de caronas acontece por pessoas que não se conhecem e pretendem compartilhar rotas por algum período regular de tempo, ou seja não é um evento único. Esse cenário é caracterizado como problema da carona diário (DCPP - *Daily Car Pooling Problem*) (Guo, Goncalves, Hsu, 2011). Existem outros casos em que há a adição de rotinas ao longo de períodos de tempo maiores, ou seja, problemas em que há o mesmo trajeto rotineiramente, e muito frequentemente com um trajeto de ida e outro trajeto de volta. Nesse tipo de cenário a intenção dos pesquisadores é buscar mapear situações de transporte semelhantes a casos reais em que colegas de trabalho procuram uma forma de locomoção comum, para o ambiente de trabalho na mesma empresa (Yan, Cheng, 2011). Esses problemas são classificados como problemas de caronas de longo prazo (LTCP – *Long Term Car Pooling Problem*).

A adição de elementos como rotinas, tabelas de preferências ou limites de tempos especificam pequenas distinções entre os problemas de carona que são abordados mais profundamente no Capítulo 2. Dentre tantas variações do problema, o foco desse trabalho é uma generalização do DARP, primeiramente em uma instância estática simples, ou seja, com um cliente apenas, sem adições em tempo de execução. Posteriormente com a adição de múltiplos clientes, todavia em um cenário ainda estático. Nesse problema, tem-se um conjunto de vértices alvos em que sua geração pode ser aleatória ou predefinida, adicionados a uma entidade que se desloca no grafo, até um limiar, que representa a capacidade do veículo. Pode-se escolher quais serão essas adições origem-destino, buscando a rota mais otimizada.

Pode-se representar o problema DARP como: existindo uma população de carros (C) e uma população de indivíduos (P), em que cada indivíduo em P possui uma tupla composta de local de origem (LO) e um local de destino (LD) que são mapeados como vértices de um grafo, deve-se escolher quais elementos de C que podem atender a um ou mais elementos de P. Assim a resolução do problema é a busca por soluções que permitam um elemento de C percorrer a menor rota no grafo a partir de um número de locais de origem (LOs) de indivíduos (P) que maximizem sua capacidade e minimizem o trajeto até os diferentes locais de destino (LDs)

existentes. Uma explicação mais elaborada do problema é apresentada no Capítulo 3 desse trabalho.

Para gerar as soluções desse problema, foi escolhida a meta-heurística de otimização por Colônia de Formiga (ACO – A*nt Colony Optimization*) (Colorni, Dorigo, Maniezzo,1991), devido à extensa literatura confirmando seu potencial para resolver problemas de roteamento (Vendramim et al. 2012), variações do caixeiro viajante (Guo, Goncalves, Huo, 2011) e problemas de carona a longo prazo (Maniezzo et al. 2004). Assim o trabalho visa à utilização de um algoritmo vinculado à meta-heurística ACO para solucionar problemas DARP estáticos.

## 1.1 TEMA DE PESQUISA

Este trabalho busca verificar a capacidade de um algoritmo baseado na meta-heurística ACO em resolver o problema DARP, em sua forma estática, com passageiros únicos e múltiplos, com geração aleatória ou predefinida em grafos fortemente conexos ou não.

## 1.2 OBJETIVOS DA PESQUISA

### 1.2.1 Objetivo Geral

Este trabalho busca verificar a capacidade de um algoritmo baseado na meta-heurística ACO em resolver o problema DARP, em suas variações, estática e de passageiros únicos e múltiplos.

### 1.2.2 Objetivos Específicos

- Identificar algoritmos utilizados para solucionar o problema de carona ou problemas semelhantes;
- Propor um algoritmo baseado na meta-heurística de otimização por colônia de formigas para o problema DARP estático;
- Expandir a solução para um problema DARP estático com múltiplos passageiros.

## 1.3 PROCEDIMENTOS METODOLÓGICOS

No presente trabalho será implementado um algoritmo com a heurística ACO para resolução do problema DARP. Posteriormente, serão feitas análises comparativas dos custos de rotas obtidas, tempo de convergência do algoritmo, padrão de mudança das soluções encontradas relativa à convergência e tempo computacional utilizado para execução do algoritmo. Essa abordagem é frequente em trabalhos semelhantes da área (Maniezzo, Mingozzi, 2004) independentemente da existência de dados reais para teste, pois uma abordagem simulada permite testes em escalas menores e também por não se possuir para esse trabalho uma base de dados reais para utilização.

Devido à complexidade da concepção da solução, a proposta de trabalho é voltada para resolver os problemas em ordem de complexidade, ou seja, resolver inicialmente uma instância do problema DARP estático com apenas um cliente, ampliando a resolução para uma instância múltipla.

Para todas as variações do problema é necessária uma simulação que consiste da emulação de cinco elementos fundamentais: o mapa (grafo), os carros servidores, os clientes, o algoritmo de teste e os relatórios de dados de execução.

A representação de um mapa de uma cidade qualquer é feita através da geração de um grafo conexo orientado, que é representado textualmente. Os valores de quantidades de vértices e arestas podem ser customizados ou definidos, e um algoritmo vai criar um grafo com esses valores e orientações incluindo valores de deslocamento nas arestas que devem ser orientadas.

No DARP estático, os elementos não se movem no mapa e apenas a solução é apresentada com seus parâmetros.

Um vértice pode ser visitado pelos agentes que representam os carros, esses criam um caminho, verifica-se a viabilidade desse caminho para resolver um conjunto problema que é representado por uma ou mais duplas origem/destino, que representa um cliente em seu ponto de origem pedindo um deslocamento até o ponto de destino.

Os servidores, ou carros para nosso problema, são gerados após o mapa ser criado, e são alocados aleatoriamente em vértices do mapa. Eles geram caminhos que simulam como se eles fossem ao encontro de um vértice que possua um cliente,

tentando utilizar uma rota curta definida pelo algoritmo que está sendo testado, com possibilidade de efetuar um trajeto que contemple vários clientes caso essa solução se apresente como viável pelo algoritmo utilizado. Sua movimentação deve seguir um padrão que represente um modelo viável para locomoção de veículos em uma cidade, ou seja, mudanças bruscas de direção não devem ocorrer. Dessa forma o grafo utilizado é um grafo dirigido, também conhecido como grafo orientado, em que cada aresta permite a movimentação em somente um sentido (do nó A para o nó B). Assim na simulação do problema o veículo fica com um movimento simulado “para frente” e “para os lados” que é mais condizente com um deslocamento real, ao contrário do que é possível em um grafo de simplesmente retroceder na existência de aresta de retorno no vértice em que estiver posicionado. A locomoção deve obedecer aos valores de custo de locomoção nas arestas.

Heurísticas distintas são aplicadas para customizar a estrutura geral de uma solução baseada na meta-heurística ACO para buscar resultados mais condizentes com o cenário proposto. A heurística é relativa à necessidade de balancear a capacidade de exploração do grafo com os caminhos encontrados, isto é presente em vários trabalhos da literatura (Hartman et al. 2014). O algoritmo obtém resultados melhores quando é customizado para o problema que se está estudando.

Uma estratégia fortemente elitista é utilizada como fator de fitness levando consideração a necessidade de resolução em tempo adequado e a característica da heurística de iterar sobre soluções anteriores.

Para cada execução do algoritmo faz-se necessário a gravação de dados para posterior análise de padrões e comparação entre índices que indiquem maior ou menor efetividade e eficácia do algoritmo e das variáveis escolhidas. Assim a codificação do artefato de teste deve conter ferramentas de registro em documentos de texto sobre os valores de variáveis chave. As variáveis que serão utilizadas para contemplar variações de performance entre os algoritmos testados, são o tempo computacional para encontrar soluções, o tempo total para a conclusão de cenários, o tempo de convergência das soluções e a diferença entre as soluções encontradas durante o período de convergência, para podermos analisar se com o passar do tempo em um mesmo grafo há ganhos de eficiência sem comprometimentos no tempo de computação da solução. Apesar dessa abordagem de verificar o tempo de máquina

ser vulnerável a várias questões técnicas como escalonamento de sistema operacional, e os requisitos de hardware utilizados, é abordagem comum presente na literatura e sua análise é mais simples e direta que abordagens direcionadas para complexidade computacional. Pela característica de calibração das variáveis em uma meta-heurística ACO a geração dessa documentação é fundamental para que o trabalho possa demonstrar os seus resultados.

### 1.3.1 TECNOLOGIAS

O trabalho foi desenvolvido com a linguagem de programação JAVA por questão de familiaridade do pesquisador com essa linguagem. Existem exemplos de trabalhos que utilizaram essa linguagem para a mesma meta-heurística no site do ACO e não há obrigatoriedade na escolha da linguagem de programação.

A IDE utilizada para o trabalho é o NETBEANS em sua versão 8.2, novamente pela familiaridade do pesquisador com a plataforma e por não haver necessidades específicas relativas à tecnologia relacionadas ao ACO.

## 1.4 ESTRUTURA DO PROJETO DE PESQUISA

O desenvolvimento deste Trabalho de Conclusão de Curso apresenta a seguinte estrutura:

O primeiro capítulo, nomeado como INTRODUÇÃO, inicia apresentando brevemente: 1) a Delimitação do Tema; 2) os Problemas e Premissas, onde são expostos os problemas que podem ser resolvidos ou melhorados com a proposta do trabalho; 3) Objetivos do Estudo; 4) a Justificativa, que tem como objetivo mostrar a importância do estudo para fins sociais, econômicos e ambientais; 5) os Procedimentos Metodológicos, que apontam os meios como os resultados do estudo serão obtidos.

O segundo capítulo, intitulado como REFERENCIAL TEÓRICO E ESTADO DA ARTE, é composto por conceitos e definições dos temas que compõem o estudo,

como sistemas inteligentes, algoritmo ACO (*Ant Colony Optimization*) e sistema de caronas CPP (*Car Pooling Problem)*.

No terceiro capítulo, chamado de RECURSOS DE HARDWARE E SOFTWARE, são apresentados os aspectos relacionados ao ambiente de desenvolvimento, físico e virtual, necessários para a implementação do estudo.

No quarto capítulo, denominado IMPLEMENTAÇÃO, a arquitetura geral da solução e os elementos gerais da implementação são analisados.

No quinto capítulo, nomeado de RESULTADOS, são apresentados os resultados da implementação.

Por fim, o sexto capítulo, designado como CONCLUSÃO, apresenta as considerações finais do estudo.

Ao final da pesquisa estão dispostas as referências bibliográficas utilizadas.

# 2 REFERENCIAL TEÓRICO E ESTADO DA ARTE

Este capítulo apresenta brevemente um resumo da literatura científica recente que fundamentou teoricamente esse trabalho. Primeiramente, salienta-se a importância da área de sistemas inteligentes que é responsável pela criação do algoritmo baseado em colônias de formigas (ACO). Posteriormente, são mencionadas pesquisas envolvendo a classe de problemas relativos a caronas de carros ou CPP e que são utilizadas para definir o problema DARP.

## 2.1 SISTEMAS INTELIGENTES

Uma das curiosidades de cientistas envolvidos com a computação foi de criar uma máquina capaz de pensar. Essa busca cria um campo da ciência, a inteligência artificial, que ao longo das décadas passou por diversos momentos, até que na década de 1990 houve um momento de grande interesse pela área com consideráveis progressos e novas técnicas e algoritmos sendo usados (Russel, 1995).

Entre as linhas de pesquisa em inteligência artificial algumas delas focaram aspectos da natureza em que pesquisadores buscaram inspiração ou replicação de mecanismos biológicos. Essa grande área ficou conhecida como Computação Natural (Ballard, 1999).

Assim vários pesquisadores focaram em pesquisar sobre elementos mais fundamentais da natureza que podem inspirar sistemas inteligentes. Dessa forma, a inspiração para reproduzir um sistema semelhante ao forrageamento de formigas surgiu com pesquisadores italianos que desenvolveram o algoritmo ACO (Ant Colony Optimization) (Colorni, Dorigo, Maniezzo, 1991).

## 2.2 *ANT COLONY OPTIMIZATION* – ACO

Biólogos descobriram que várias espécies de formigas utilizam uma substância química chamada feromônio quando saem em busca de alimento. Quando uma

formiga encontra uma fonte de alimento ela libera mais desse feromônio e retorna para a colônia, outras formigas buscam caminhos que contenham mais feromônio para seguir e esse ciclo acaba reforçando caminhos nos quais uma fonte de comida foi encontrada através de um reforço positivo (Dorigo, Stutzle, 2009).

Buscando verificar se esse tipo de comportamento poderia ajudar na resolução de problemas computacionais um grupo de pesquisadores italianos criou o ACO para resolver o problema clássico do caixeiro viajante.

O algoritmo ACO é um algoritmo estocástico probabilístico, em que agentes inteligentes de complexidade simples, se comunicam indiretamente para gerar conjuntos de soluções que competem entre si em busca de uma convergência de uma solução boa o suficiente para o problema. Ou seja, as decisões dos agentes individuais são governadas por um cálculo baseado em valores e possibilidades aleatórias, que define qual será o caminho a ser tomado entre os possíveis, sendo que a escolha do caminho normalmente é feita entre: uma heurística vinculada ao problema analisado, ou o valor de concentração de feromônio na trilha.

A combinação desses dois fatores é fundamental para o sucesso do ACO, pois uma abordagem heurística, como por exemplo, uma escolha puramente gulosa não é otimizada necessariamente e reduz o conjunto solução dos caminhos possíveis em um grafo. A adição dos valores do feromônio permite a exploração de novos caminhos, considerando um valor basal naturalmente presente em todos os caminhos, e posteriormente é fator responsável para que novas soluções sejam constituídas de subconjuntos de outras soluções já encontradas. Assim, há iteração sobre um subconjunto inicial através do reforço positivo, ao mesmo tempo em que permite exploração de novas opções não abrangidas pela heurística escolhida, e assim, espera-se varrer o conjunto de soluções possíveis de um grafo encontrando-se um resultado melhor (Dorigo, Stutzle, 2009).

O algoritmo ACO funciona implementando “formigas virtuais” depositadas em diversos pontos de um grafo conexo, que fazem uma busca por algum elemento de interesse. Ao longo dessa busca seu caminho percorrido é salvo, e caso um conjunto

solução seja encontrado, gera-se uma marcação na sua trilha, simulando a deposição do feromônio, em um caminho de volta. Caso múltiplos caminhos sejam encontrados, por dois agentes por exemplo, a taxa de deposição de feromônio pode ser diferente entre eles, normalmente isso ocorre para que a trilha seja marcada mais fortemente no caminho menor, dessa forma pode-se encontrar rotas otimizadas para algum problema que possa ser representado em um grafo conexo. Uma das grandes vantagens do ACO é que ele pode atualizar as trilhas dinamicamente, assim mudanças no grafo levam os agentes a se adaptarem aos novos cenários (Dorigo, Gambardella, 1997).

Uma representação simples do ACO em pseudocódigo pode ser a seguinte (Dorigo, Stutzle, 2009):

***Procedimento ACO***
***Inicialização***
Enquanto (condição de parada não encontrada) faça
***ConstruaFormigasSolução***
***ApliqueBuscaLocal(opcional)***
***AtualizaçãoGlobalFeromonio***
***Fim***
***Fim ACO***

O Procedimento ACO é subdividido em quatro passos fundamentais, o primeiro deles é Inicialização, que consiste na inicialização do algoritmo, que cria as variáveis a serem utilizadas como os feromônio e as inicia com valor zero.

A chamada para ConstruaFormigasSolução estabelece a inicialização do conjunto de formigas que vão através de modo estocástico percorrer diversos caminhos no grafo, a cada passo adicionando uma possível aresta levando em consideração valores de feromônio ou de informação da heurística relativa ao problema.

A chamada para ApliqueBuscaLocal é feita para poder analisar entre o conjunto de soluções criadas pelas formigas e ver se alguma atinge o objetivo procurado ou aplicar outras ações. Tem-se como exemplo dessas ações, fazer uma troca entre duas arestas e verificar se o resultado obtido é melhor que o atual.

A chamada final dentro do ciclo é a de AtualizaçãoGlobalFeromonio que define a atualização dos valores das trilhas de feromônio buscando favorecer trilhas para outras formigas, e inclusive levando em conta fatores como “evaporação” para remover trilhas com menor trânsito. Essa atualização pode ocorrer de diferentes formas dentro da meta-heurística, sendo comumente aplicada após uma solução ter sido gerada.

O ACO como meta-heurística apresenta certa liberdade em suas implementações, e assim, existem variações na literatura com camadas de complexidade e mudanças devido a especificidades de problemas. São variações conhecidas do ACO: *rank-based Ant System* (AS-*rank*), MAX-MIN *Ant System* (MMAS), *Ant Colony System* (ACS) (Dorigo, Blum, 2005).

O AS-*rank* é uma variação do ACO em que a atualização de feromônio é feita com uma estratégia elitista, ou seja, há uma priorização das soluções encontradas, e apenas algumas soluções podem depositar feromônio (Dorigo, Blum, 2005). Assim nessa variação dentre os resultados encontrados, é gerada uma comparação e um sistema de classificação das melhores soluções, e somente as soluções até um certo número (por exemplo as 3 melhores soluções) atualizam os seus caminhos.

A variação MMAS define valores máximos e mínimos para o feromônio nas trilhas, e parte de valores iniciais máximos com decaimento de evaporação para todos os caminhos exceto o melhor encontrado, assim com uma abordagem de atualização de feromônio extremamente elitista. A quantidade de feromônio depositada é também uma função da distância percorrida: g(fd) = 1/f(dp), assim quantidades maiores do feromônio são depositadas em soluções melhores favorecendo uma convergência para elas (Dorigo, Blum, 2005).

O algoritmo ACS favorece o aproveitamento do conhecimento adquirido pelos caminhos percorridos anteriormente, e utiliza esse conhecimento para evitar construir soluções sem expandir a exploração do ambiente e gerar uma convergência muito prematura de resultados. O ACS gera um novo fator de escolha entre as arestas baseado em um valor relativo ao produto do valor heurístico e da taxa de feromônio, ou seja, quanto maior o valor dessa multiplicação maior a possibilidade de a aresta ser escolhida pela formiga. O algoritmo faz a atualização das taxas de feromônio com uma tendência fortemente elitista, assim somente o melhor caminho encontrado deposita feromônio, e o caminho de feromônio é atualizado durante a sua execução antes de encontrar uma solução. Isso permite que uma formiga reduza a quantidade de feromônio na sua rota para evitar que outras a sigam, reduzindo o valor da multiplicação da heurística e do valor de feromônio. Este cenário é especialmente útil para formigas que percorrem caminhos ruins, pois juntamente com a taxa de evaporação faz com que trilhas já percorridas com soluções ruins não sejam seguidas. Outra particularidade desse algoritmo é a atualização das trilhas de feromônio enquanto constrói as soluções (Dorigo, Blum, 2005).

A meta-heurística ACO (Dorigo, Birattari, Stutzle, 2006) é utilizada nesse trabalho e proporciona um modelo de solução para problemas de otimização em cenários NP-difícil com elementos que podem ser mudados para melhor caracterizar o problema específico em questão. Essas variações devem levar em conta a exploração do ambiente de busca contra a convergência da solução, o tempo para encontrar a solução e a heurística de escolha do problema passo a passo, podendo assim um algoritmo baseado na meta-heurística ACO conter elementos diversos dos algoritmos ACO-rank, MMAS, ACS, entre outras abordagens.

Faz-se importante salientar que os problemas que o ACO lida são NP-difícieis e que, dessa forma, a solução ótima é de difícil obtenção. O algoritmo se baseia em encontrar uma solução boa o suficiente para o problema em um tempo de convergência aceitável. Considerações teóricas a respeito da capacidade do algoritmo encontrar a solução ideal (convergência de solução) e em tempo aceitável são expostas por (Dorigo, Birattari, Stutzle, 2006) e não são o alvo desse trabalho, que se

preocupa com a implementação do algoritmo em um cenário simulado próximo ao real em busca da demonstração da resolução do problema em questão.

Todavia a convergência entre as soluções encontradas é um detalhe a ser lidado com cautela. Assim, deve-se gerar um conjunto de variáveis que permitam ao algoritmo percorrer uma quantidade de caminhos distintos grande o suficiente para que ache boas soluções, e permitir que haja convergência entre as soluções em tempo computacional adequado, sendo assim, valores como taxa de deposição de feromônio, taxa de evaporação de feromônio, valores mínimos e máximos do feromônio, quantidade de formigas geradas por ciclo do algoritmo, e chance probabilística de percorrer novos caminhos são fatores essenciais para o bom funcionamento do ACO em condições de resolver problemas práticos.

Apesar das pesquisas mostrarem que os resultados iniciais utilizando ACO não foram ótimos para a resolução do caixeiro viajante (Dorigo, Stutzle 2009), a iniciativa se mostrou válida, especialmente com posteriores estudos que melhoraram a solução para problemas específicos. Assim outros pesquisadores buscaram verificar se essa estrutura de algoritmo pode ser utilizada para resolver outros tipos de problemas com características semelhantes, e obtiveram sucesso para problemas tais como: variações do caixeiro viajante (Dorigo, Gambardella, 1997), indexação em banco de dados (Vendramin, Pereira, Pozo, 2010), de roteamento de redes (Vendramin et al. 2012) e também para uma classe de problemas de “carona” conhecida como CPP (Hartman et al. 2014).

## 2.3 CAR POOLING PROBLEM – CPP

Várias cidades no mundo sofrem com problemas consideráveis de excesso de veículos em suas vias causando congestionamentos gigantes. Esses congestionamentos fazem com que pessoas percam inúmeras horas de suas vidas em trânsito, que em caso na inexistência da aglomeração de veículos poderiam fazer em uma fração do tempo. Dessa forma inúmeros pesquisadores buscam um conjunto de soluções que podem melhorar de alguma forma o problema de transporte em grandes centros urbanos.

A definição geral básica de um problema de carona consiste no compartilhamento de um veículo (frequentemente carro) entre duas ou mais pessoas com o objetivo de fazer uma rota em comum, ou muito semelhante. As motivações para isso são diversas: redução de custos, compartilhamento de recursos, redução de veículos nas vias e do tráfego, socialização, melhoria da qualidade do ar através da redução da emissão de poluentes, entre outras vantagens.

Na literatura existem variações de nomenclatura para essa classe de problemas de acordo com suas especificidades. Pode-se sumarizar essas diferenças nos seguintes tipos: problema de caronas que envolvem informações prévias, problemas de caronas que envolvem sazonalidade, problemas que incluem limitações temporais e se o veículo é pessoal ou privado (de uma companhia que presta serviços de transporte).

O problema de carona em carros, o CPP (*Car Pooling Problem*) é normalmente descrito quando duas ou mais pessoas compartilham um veículo. Todavia, o veículo é de posse de uma das pessoas, assim a rota é de seu interesse de forma não comercial, ou seja, não é o seu trabalho transportar pessoas, e a carona não foi baseada em informações disponíveis previamente, não havendo informação sobre a sazonalidade nessa modalidade.

Quando há informação sobre a sazonalidade, o CPP é redefinido comumente em dois tipos: o de curto prazo ou diário, conhecido como *Daily Car Pooling Problem* - DCPP (Guo, Goncalves, Hsu, 2011), e o de longo prazo ou *Long Term Car Pooling Problem* - LTCPP (Guo, Goncalves, Hsu, 2013). No DCPP a sazonalidade é curta, podendo ser uma viagem de ida e volta em um único dia ou em um período similar de natureza efêmera. Quando há um acordo de longo prazo entre os participantes, normalmente devido a uma rotina em comum, como deslocamento para o trabalho ou uma instituição de ensino, o problema é redefinido como LTCPP, e frequentemente existem informações prévias dos participantes, fato que pode gerar mudanças em casos específicos de conflitos ou mudanças de rotina de um dos participantes alterando a dinâmica do problema.

A literatura costuma classificar o problema de “disque-carona” ou *Dial a Ride Problem* - DARP quando a posse do veículo é de uma companhia privada que efetua transporte de passageiros e existe uma população de veículos (que para todos os efeitos do problema são iguais) que atende à demanda de uma população de usuários de forma normalmente individual (Yan, Chen, 2011) (um cliente por veículo) com poucos exemplos na literatura do problema buscando a lotação máxima do veículo ou seja uma resolução em que um carro atenda vários clientes. O DARP também pode apresentar variações na camada temporal, porém não relativo à sazonalidade e sim à estruturação do problema em um ambiente dinâmico ou estático. Em um ambiente dinâmico, novos clientes e carros podem surgir enquanto o problema é executado e as posições são atualizadas no mapa para simular uma situação próxima a real.

Na Tabela 1 são apresentados exemplos de características de cada tipo de problema discutido nesse trabalho, sendo que outras categorias de problemas similares existem, porém, não foram consideradas pertinentes a este projeto (Hartman et al. 2014).

**Tabela 1 – Tipos de problemas de caronas.**

| Nome | Informações Prévias | Sazonalidade | Limitações Temporais | Veículo Pessoal |
|---|---|---|---|---|
| **CPP** | | | | X |
| **DCPP** | | X | | X |
| **LTCPP** | X | X | | X |
| **DARP** | | | X | |

**Fonte: Autoria própria, 2019.**

# 3 RECURSOS DE HARDWARE E SOFTWARE

Nessa seção são abordados aspectos relacionados ao ambiente de desenvolvimento tanto físico como virtual necessários para a implementação desse trabalho.

## 3.1 RECURSOS DE HARDWARE

Não há nenhuma necessidade extrema em termos de recursos de hardware para o desenvolvimento do trabalho, assim serão utilizadas duas máquinas com características distintas. Um laptop com 2gb de RAM e um processador Intel Core 2 Duo @ 2.13 GHz com dois sistemas operacionais instalados, Windows 7 e Ubuntu 16, e um computador desktop com processador Intel(R) Core (TM)2 Quad Q6600 @ 2.40GHz com 8gb de RAM e sistema operacional Windows 8.1.

## 3.2 RECURSOS DE SOFTWARE

Para a formulação prática do problema utilizaremos a linguagem orientada a objetos JAVA. O desenvolvimento do código ocorrerá através do software NetBeans que permite a criação, edição e compilação de códigos fonte Java. A instalação do JDK Java Development Kit 8.0 é necessária para o funcionamento adequado da ferramenta NetBeans.

O código será salvo em disco nos dois computadores disponíveis e também em servidor remoto através da ferramenta Dropbox que permite compartilhamento e controle de arquivos entre diversas máquinas.

# 4 IMPLEMENTAÇÃO

Neste capítulo serão apresentadas a Arquitetura Geral e os Elementos Gerais da Implementação.

## 4.1 ARQUITETURA GERAL

A solução implementada consiste em codificação em Java no paradigma de Orientação a Objeto. Em que se busca uma solução em um grafo, utilizando um conjunto de formigas, que percorrem um caminho de vértices e arestas. O conjunto problema deve estar contido no caminho das formigas, o que torna uma solução a ser analisada entre as possíveis soluções das diversas formigas. No algoritmo proposto as soluções são comparadas e somente são consideradas para atualizações de trilhas de feromônio, aquelas em que o somatório dos valores das arestas seja um valor menor do que da solução encontrada anteriormente.

Estruturou-se a codificação com as seguintes classes:

- Main
- ACOCustom
- Formiga
- Grafo
- Vertice
- Aresta
- Caminho
- Problema
- Solucoes

## 4.2 ELEMENTOS GERAIS DA IMPLEMENTAÇÃO

A implementação desse trabalho pode ser descrita em 3 passos principais: o primeiro sendo a geração de elementos necessários ao processamento, utilizando valores de entrada de dados para gerar estruturas de dados que serão utilizadas posteriormente; o segundo sendo o processamento dos dados gerados ou inseridos

na entrada; e o terceiro sendo os resultados da execução, ou seja a saída da implementação.

Cada um dos elementos será explicado em alto nível aqui, e uma cópia do código está disponível em https://github.com/heimoski/ACO-for-Darp para consulta.

O código foi estruturado em sua classe principal para ter dois modos distintos de execução. O primeiro modo gera um grafo e um conjunto de problemas aleatórios a cada execução. Esse modo gera um grafo direcionado conexo, em que os valores das arestas são definidos aleatoriamente, e que todos os vértices têm conexões a todos os vértices do grafo e um conjunto de problemas P em que a quantidade de elementos em P é igual a uma variável de controle. O outro modo de execução utiliza elementos fixos em código tanto na geração do grafo, como no conjunto problema. Nesse caso um grafo é representado por uma matriz de inteiros que é uma variável estática. A principal vantagem e a diferença entre esses dois modos é que no modo de geração de grafos e problemas aleatórios pode se testar se a implementação funciona com conjuntos diferentes e o caso estático permite a análise de outros valores parametrizáveis e a comparação de execuções utilizando variações desses parâmetros.

## 4.3 ELEMENTOS GERAIS DE ENTRADA/PARAMETRIZAÇÃO

Os elementos de entrada permitem a parametrização de variáveis utilizadas na execução da implementação. São importantes devido às possibilidades que geram ao permitir comparar como valores diferentes influenciam saídas para conjuntos de problemas.

Como elementos de entrada, há valores que são utilizados na construção dos elementos necessários à geração da resposta. Cada elemento é parametrizável na classe principal da implementação e são valores estáticos, seus nomes e seus objetivos são listados abaixo:

- Número de Vértices(V) – define a quantidade de vértices que o grafo a ser gerado deve possuir;

- Número de Formigas(F) – Quantidade de agentes a percorrer o mapa a cada geração;
- Alfa(A) – Constante utilizada para ponderar o valor do feromônio na escolha da formiga;
- Beta(B) – Constante utilizada para ponderar sobre o valor heurístico na escolha da formiga;
- Taxa de Evaporação (TeV) – Valor que reduz a quantidade de feromônio nos vértices;
- Valor Inicial Feromônio (ViF) – Valor inicial do feromônio nas arestas;
- Número de Problemas(P) – Quantidade de problemas que a solução gerada deve resolver simultaneamente;
- Gerações(G) – Quantidade de problemas que a solução gerada deve resolver simultaneamente;
- Modo Aleatório (MA) – Variável que controla se o grafo gerado será aleatório ou se é estático;
- Matriz de Valores do Grafo (MVG) – Matriz representando um grafo que deve ser utilizada caso não se utilize o modo aleatório, ou seja, o valor da variável “modo aleatório” seja “falso”.
- Quantidade de Iterações(Qi) - Valor que controla a quantidade de ciclos de resoluções, ou seja, se esse valor for 2, o programa roda duas iterações em que cada iteração resolve os problemas na quantidade de gerações fornecida.
- Limite Passos Vértices Para Achar Solução (PVS) - Este valor é um valor limítrofe para controlar as iterações dos passos das formigas, relativo à quantidade de vértices. É um indicie multiplicativo (x1, x2) em que atingindo o número de passos relativo à quantidade de vértices, a condição de parada da geração é aceita.
- Limite Mínimo De Soluções Para Ir Para Próxima Geração (SPG) - Este valor é um valor limítrofe para controlar as iterações em uma geração através das soluções encontradas pelas formigas. Caso tenha-se 5 formigas, e para agilizar a execução do algoritmo, considerarmos que quando 2 encontraram soluções

podemos ir para a próxima geração, essa variável pode ser atribuída com esse valor, para esse efeito.

## 4.4 EXECUÇÃO DA SOLUÇÃO

A proposta do pseudocódigo simplificado do ACO (Dorigo, Stutzle, 2009) é a seguinte:

***Procedimento ACO***
***Inicialização***
Enquanto (condição de parada não encontrada) faça
***ConstruaFormigasSolução***
***ApliqueBuscaLocal(opcional)***
***AtualizaçãoGlobalFeromonio***
***Fim***
***Fim ACO***

A implementação feita nesse trabalho segue estrutura semelhante, considerando o fato de a meta-heurística permitir pequenas modificações, sendo a seguinte:

***Início do Programa***
***Inicialização***
*Construção do conjunto problema*
*Construção do Grafo*
***Enquanto*** *a quantidade de gerações não for atingida*
*Alocação das entidades no Grafo*
***Enquanto*** *a condição de parada não for atingida (Número de passos* [PVS]*, ou Quantidade de Soluções Encontradas* [SPG]*)*
*Movimentar as entidades segundo a heurística ACO*
*Comparar os caminhos para verificar se algum contém o conjunto solução*

***Fim Enquanto***

*Comparar soluções e eleger a melhor*

*Atualizar os valores de feromônios no grafo*

***Fim Enquanto***

***Fim do Programa***

## 4.5 DETALHES DA IMPLEMENTAÇÃO

Para facilitar a compreensão da implementação, detalhes mais específicos de cada passo do algoritmo são abordados nessa seção utilizando como base os passos do pseudocódigo.

### 4.5.1 Construção do Conjunto Problema

O conjunto problema é composto de valores de origem e destino que podem ser criados aleatoriamente ou definidos manualmente caso a execução não seja de modo aleatório (MA). Os valores devem ser de vértices válidos, ou seja, existentes no grafo, porém é possível que ciclos sejam gerados com múltiplos problemas, o único controle existente é para que o vértice de destino seja diferente do vértice de origem.

### 4.5.2 Construção do Grafo

O grafo utilizado na resolução dos problemas é construído recebendo como parâmetros uma quantidade de vértices(V), formigas(F), problemas(P), e gerações. Se o programa estiver executando no modo aleatório (MA) o grafo gera arestas entre todos os seus vértices, sendo assim um grafo fortemente conexo. As arestas possuem valores aleatórios de 1 até 10, dessa forma o menor caminho entre dois pontos não é necessariamente uma única aresta (O caminho AB com custo 10 é pior que o caminho ACB, sendo que A-> C tem custo 1 e C->B tem custo 5, para um custo total de 6 ao invés de 10). O grafo pode ser também fornecido de maneira estática como uma matriz para testes específicos, considera-se o valor de aresta 0 como uma desconexão, assim permitindo a utilização de grafos não fortemente conexos). O grafo também

controla a inicialização e distribuição da quantidade de formigas parametrizada em seus vértices. Tentando distribuir inicialmente uma formiga por vértice, mas sendo o número superior, atribuí uma quantidade maior por vértice.

### 4.5.3 Enquanto a quantidade de gerações não for atingida

Essa condição força a geração de soluções ser executada na quantidade de vezes que for estipulado o valor de gerações(G). A cada geração são atribuídas formigas ao grafo para serem colocadas em seus vértices e após isso um ciclo é iniciado.

### 4.5.4 Alocação das entidades no Grafo

Além das formigas, variáveis de controle são também inicializadas e após isso um ciclo que busca soluções começa.

### 4.5.5 Enquanto a condição de parada não for atingida (Número de passos, ou Quantidade de Soluções Encontradas)

Essa condição de parada do ciclo de busca de soluções é baseada na análise de duas variáveis de inicialização:

- Limite Passos Vértices Para Achar Solução (PVS)
- Limite Mínimo De Soluções Para Ir Para Próxima Geração (SPG)

PVS é um valor que é multiplicado a quantidade de vértices para gerar uma quantidade de passos que as formigas podem dar até a condição de parada. Assim um grafo com 10 vértices, podemos estipular um valor de PVS de 10 para estipular que no pior caso vamos ter 100 passos dados pelas formigas antes de encerrar uma geração e verificar seus resultados.

SPG é o valor utilizado caso se queira controlar o ciclo de busca de soluções através da quantidade de formigas que encontraram soluções. Por exemplo se tivermos 10 formigas tentando resolver P, podemos atribuir o valor de SPG como

5, e a partir do momento que tivermos 5 formigas que tenham soluções podemos encerrar um ciclo de busca de soluções e iniciar uma outra geração.

O intuito de ter esses dois valores sendo verificados é contornar o problema da parada. Sendo problemas de carona NP-Completos não temos como saber quando uma solução será encontrada, assim, para otimizar a solução, limites podem ser colocados através de PVS e SPG para evitar de ter uma geração em que a condição de parada não seja atingida em tempo razoável, pois no pior caso a busca de soluções pode ser infinita.

### 4.5.6 Movimentar as entidades segundo a heurística ACO

Este passo compreende o processo de escolha de um vértice pelas formigas no grafo. Cada formiga tem uma posição de vértice, e cada vértice tem uma quantidade de arestas. A escolha que a formiga faz entre essas arestas possíveis é baseada no trabalho sobre o ACO (Dorigo, Birattari, Stutzle, 2006) e segue a seguinte fórmula.

$$Ve = (fator\ feromônio^{\alpha} * fator\ heurístico^{\beta}) \div \sum (fator\ feromônio^{\alpha} * fator\ heurístico^{\beta})$$

Onde **α** (Alpha) é uma variável de parametrização para controlar atração de caminhos percorridos e **β** (Beta) é uma variável de controle de atração para caminhos não percorridos por formigas ainda, ou seja, caminhos que não fazem parte da melhor solução já encontrada.

Sendo o Valor de Escolha (Ve), e os valores de fator de feromônio e fator heurístico, que são atribuídos a cada aresta com base nas seguintes fórmulas:

Fator Heurístico = 1.0 / Custo da Aresta

Fator Feromônio = ViF *(atualizado a cada geração)

Alterações no Fator Feromônio são explicadas posteriormente.

A escolha das formigas é feita por um sorteio, em que um valor pseudoaleatório é calculado, e a partir desse valor, faz-se uma verificação tentando encaixar ele em algum dos Valores de Escolha (Ve) possíveis, seguindo uma lógica em que, se o número sorteado, convertido para centesimal, for menor que o valor de (Ve) calculado, a formiga escolhe esse vértice, caso contrário, o valor de (Ve) é reduzido pelo valor da aresta, e utiliza-se esse valor para à próxima comparação. Esse decremento ocorre para evitar que nenhuma aresta não seja escolhida ao aproximar o valor de 0.

### 4.5.7 Comparar os caminhos para verificar se algum contém o conjunto solução

Este passo verifica se entre todos os caminhos das formigas(uma sequência de vértices), há algum que contenha todos os múltiplos caminhos necessários, ou seja as origens e destinos de P. Um caminho somente é considerado válido se contém todos os elementos de P e se eles estão em ordem origem->destino, evitando assinalar um caminho como válido por conter destino->origem de algum elemento de P.

### 4.5.8 Comparar soluções e eleger a melhor

Este passo compara todas as soluções válidas para verificar qual é a melhor solução daquela geração até aquele momento. A definição de melhor solução é aquela que contém o menor custo, ou seja, o valor de custo das arestas utilizadas no caminho.

### 4.5.9 Atualizar os valores de feromônios no grafo

A atualização do Fator Feromônio ocorre de duas formas, quando uma trilha é parte da melhor solução encontrada um valor extra é adicionado segundo a seguinte fórmula:

$$taxa\ de\ update = 1 \div valor\ do\ caminho$$

Sendo que valor do caminho representa o valor total de um caminho, ou seja, a soma do custo de locomoção em todas as suas arestas.

A evaporação é representada por:

$$Valor\ Feromônio\ Final = Taxa\ Feromônio\ Atual * Taxa\ de\ Evaporação;$$

Atualiza-se somente o melhor caminho encontrado até aquele momento entre as gerações de uma iteração. Assim essa é uma estratégia fortemente elitista de escolha para atualização; A quantidade de formigas que encontram caminhos também pode ser definida pelo parâmetro SPG o que pode fazer a escolha ser mais concorrida para um valor alto ou mais branda para um valor baixo (primeiro que termina tem preferência).

## 4.6 VERIFICAÇÃO DA SOLUÇÃO E SAÍDA FINAL

A saída do programa desenvolvido é a melhor solução encontrada que resolve o conjunto problema. Acompanhado de diversos logs que são impressos na tela.

Os logs mostram os caminhos que as formigas percorreram e as soluções encontradas. Eles podem ser desabilitados diretamente no código.

# 5 RESULTADOS

## 5.1 RESULTADOS INICIAIS

A implementação executada nesse trabalho consegue resolver o problema DARP para instâncias estáticas dependendo da parametrização utilizada. E dependendo da parametrização alguns comportamentos podem ser observados com o que foi alterado.

Para cada bateria de testes utilizaremos uma tabela de parametrização referenciando os valores enunciados e definidos na seção 4.3 desse trabalho.

Como os valores de entrada são parametrizáveis, várias resoluções foram executadas. Cada conjunto de execuções salientando algum elemento do trabalho.

Incialmente foi realizada uma demonstração de uma solução simples em um grafo de tamanho reduzido com apenas um problema e a seguinte configuração de parametrização.

**Tabela 2 – Variáveis de Parametrização Para Primeira Bateria de Testes.**

| V | F | A | B | TeV | ViF | P | G | MA | MVG | Qi | PVS | SPG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 0.9 | 0.1 | 0.9 | 1 | 1 | 5 | Sim | N/A | 1 | 3 | 3 |

**Fonte: Autoria própria, 2019.**

A matriz de distâncias gerada aleatoriamente foi a seguinte:

**Tabela 4 – Matriz 1 de Representação de Grafo Utilizada nos Resultados Iniciais.**

| | V0 | V1 | V2 |
|---|---|---|---|
| **V0** | 2 | 1 | 3 |
| **V1** | 5 | 3 | 9 |
| **V2** | 6 | 2 | 6 |

**Fonte: Autoria própria, 2019.**

O problema gerado foi: Origem: 0 - Destino: 1

As soluções encontradas:

Melhor Solução da Geração 0 Custo: 9 - Caminho: 0, 0, 0, 2, 1

Melhor Solução da Geração 1 Custo: 1 - Caminho: 0, 1

Melhor Solução da Geração 2 Custo: 1 - Caminho: 0, 1

Melhor Solução da Geração 3 Custo: 1 - Caminho: 0, 1

Melhor Solução da Geração 4 Custo: 1 - Caminho: 0, 1

**Melhor Solução da Iteração: Geração 1, Custo 1**

Este é um problema muito simples que traz pouco valor, todavia temos uma prova que o artefato codificado nesse trabalho consegue resolver uma instância do problema de carona tal como especificado para um problema (conjunto origem, destino de um usuário potencial) e mostrar uma melhoria nas respostas encontradas em gerações posteriores. Outro fato interessante que pode ser visualizado é a forte convergência da solução, ou seja, após encontrada uma solução a mesma foi sendo repetida por gerações posteriores exemplificando o fenômeno da convergência onde formigas param de explorar soluções alternativas e repetem uma solução já encontrada.

Se alterarmos a parametrização para 4 Vértices, com 4 Formigas e 2 problemas, dessa forma simulando o destino de um veículo até um usuário e até um outro ponto final, por exemplo, utilizando duas iterações. mantendo todos os demais valores, temos a seguinte tabela de parametrização, matriz e os seguintes resultados:

**Tabela 5 – Variáveis de Parametrização Para Segunda Bateria de Testes.**

| V | F | A | B | TeV | ViF | P | G | MA | MVG | Qi | PVS | SPG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 0.9 | 0.1 | 0.9 | 1 | 2 | 5 | Sim | N/A | 2 | 3 | 3 |

**Fonte: Autoria própria, 2019.**

**Tabela 6 – Matriz 2 de Representação de Grafo Utilizada nos Resultados Iniciais.**

| | V0 | V1 | V2 | V3 |
|---|---|---|---|---|
| **V0** | 4 | 3 | 6 | 2 |
| **V1** | 6 | 1 | 5 | 8 |
| **V2** | 8 | 8 | 4 | 9 |
| **V3** | 6 | 9 | 6 | 1 |

**Fonte: Autoria própria, 2019.**

Problemas Gerados Foram:

P1 - Origem: 1 - Destino: 0

P2 - Origem: 3 - Destino: 2

Iteração 1

Melhor Solução da Geração 0 Custo: 52 - Caminho: 1, 3, 2, 2, 3, 2, 2, 3, 0

Melhor Solução da Geração 1 Custo: 33 - Caminho: 0, 1, 3, 1, 2, 0

Melhor Solução da Geração 2 Custo: 39 - Caminho: 0, 1, 3, 1, 2, 1, 0

Melhor Solução da Geração 3 Custo: 31 - Caminho: 0, 1, 2, 3, 2, 0

Melhor Solução da Geração 4 Custo: 23 - Caminho: 0, 3, 0, 1, 0, 2

**Melhor Solução da Iteração: Geração 4, Custo 23**

Iteração 2

Melhor Solução da Geração 0 Custo: 48 - Caminho: 1, 1, 1, 0, 1, 1, 1, 1, 2, 0, 2, 3, 2

Melhor Solução da Geração 1 Custo: 24 - Caminho: 0, 1, 0, 3, 3, 0, 2

Melhor Solução da Geração 2 Custo: 44 - Caminho: 0, 0, 2, 3, 2, 2, 1, 1, 0

Melhor Solução da Geração 3 Custo: 26 - Caminho: 1, 3, 0, 0, 1, 2

Melhor Solução da Geração 4 Custo: 46 - Caminho: 1, 2, 1, 0, 2, 3, 0, 2

**Melhor Solução da Iteração: Geração 1, Custo 24**

Observa-se que, aumentado o número de problemas conseguimos em um novo grafo com novos objetivos, novamente atingir respostas para o problema, porém o comportamento entre as gerações é mais errático, e nem sempre gerações subsequentes obtém resultados melhores. A natureza aleatória da meta-heurística e a parametrização são fatores importantes para esse comportamento. Como problemas NP-Completos são problemas que não temos como saber quando obteremos soluções e se a solução encontrada é a melhor esse tipo de comportamento é aceitável, visto que podemos descartar esses resultados, e somente aplicamos atualizações de trilhas de feromônios no melhor resultado encontrado entre as gerações de uma iteração até o momento. Porém como artefato de estudo esses resultados mesmo que sendo descartados pela heurística são importantes para entender o comportamento das formigas.

## 5.2 RESULTADOS DOS VALORES DE ALFA E BETA

Utilizando um grafo fornecido estaticamente, ou seja, escrito em parâmetro de entrada, para execução em um modo não aleatório, em que os problemas são também fornecidos como parâmetros (P=3, sempre os mesmos problemas) podemos fazer uma análise sobre variação de parâmetros tais como alpha e beta.

Utilizando um grafo não conexo, em que valores de 0 entre vértices representam arestas que não podem ser navegadas, com 3 problemas para a seguinte parametrização, matriz e problemas, obtemos os seguintes resultados.

**Tabela 7 – Variáveis de Parametrização Para Terceira Bateria de Testes.**

| V | F | A | B | TeV | ViF | P | G | MA | MVG | Qi | PVS | SPG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 0.9 | 0.1 | 0.9 | 1 | 3 | 10 | Não | MA | 10 | 3 | 3 |

**Fonte: Autoria própria, 2019.**

MA representa a matriz de entrada para o grafo sendo ela:

**Tabela 8 – Matriz 3 de Representação de Grafo Utilizada nos Resultados Iniciais.**

| | V0 | V1 | V2 | V3 | V4 |
|---|---|---|---|---|---|
| **V0** | 1 | 0 | 3 | 4 | 5 |
| **V1** | 5 | 4 | 0 | 2 | 1 |
| **V2** | 2 | 4 | 6 | 0 | 10 |
| **V3** | 10 | 8 | 6 | 4 | 0 |
| **V4** | 1 | 3 | 5 | 8 | 13 |

**Fonte: Autoria própria, 2019.**

Problemas Gerados Foram:

Origem: 0 - Destino: 1

Origem: 2 - Destino: 3

Origem: 3 - Destino: 4

Executando para 10 gerações em 10 iterações diferentes obtemos os seguintes resultados para a média dos resultados das gerações, o desvio padrão dos resultados encontrados e os melhores resultados de cada geração. A última coluna da tabela

representa se o resultado é menor que a média menos o desvio padrão daquela geração, e também se esse valor é menor que a média e o desvio padrão de todas as iterações daquela execução. Resultando assim em 3 valores possíveis: “Não”, quando o valor não é menor que a comparação em sua geração; “Sim, Local”, quando o valor é menor que as taxas calculadas em sua geração porém não é menor que os valores calculados utilizando todos os resultadas da iteração; “Sim, Total”, quando os valores são inferiores às taxas utilizando todos os resultados de todas as gerações e iterações, sendo assim fora da taxa de desvio esperada naquela execução.

**Tabela 9 – Resultados e Estatísticas da Terceira Bateria de Testes**

| Iteração | Média das Soluções | Desvio Padrão | Média - Desvio Padrão | Melhor Solução | Geração da Melhor Solução | Caminho da Melhor Solução | Resultado Significativo |
|---|---|---|---|---|---|---|---|
| **1** | 46 | 22,89 | 23,11 | 25 | 2 | 0,2,1,3,2,4 | Não |
| **2** | 51 | 11,32 | 39,68 | 30 | 3 | 2,4,0,4,0,3,1,4 | Sim, Total |
| **3** | 56 | 10,10 | 45,90 | 36 | 7 | 0,2,2,4,3,1,4 | Sim, Local |
| **4** | 50 | 16,14 | 33,86 | 28 | 9 | 0,2,0,0,0,3,3,1,1,4 | Sim, Total |
| **5** | 40 | 13,82 | 26,18 | 25 | 7 | 0,0,2,0,3,3,3,1,1,4 | Sim, Total |
| **6** | 46 | 12,28 | 33,72 | 20 | 6 | 0,2,0,3,2,1,4 | Sim, Total |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **7** | 50 | 13,22 | 36,78 | 23 | 9 | 0,3,2,1,4,3 | Sim, Total |
| **8** | 45 | 11,51 | 33,49 | 28 | 8 | 2,2,1,0,3,1,4 | Sim, Total |
| **9** | 48 | 15,75 | 32,25 | 25 | 3 | 0,2,1,4,3,1,4 | Sim, Total |
| **10** | 50 | 13,72 | 36,28 | 26 | 6 | 0,2,1,4,0,3,3,1,4 | Sim, Total |

**Fonte: Autoria própria, 2019.**

**Tabela 10 – Resultados Populacionais da Terceira Bateria de Testes**

| Média das médias | Desvio Padrão Médio | Média das Médias - Desvio Padrão Médio | Média das Gerações com Melhor Solução |
|---|---|---|---|
| 48,20 | 14,07 | 34,12 | 6 |

**Fonte: Autoria própria, 2019.**

**Tabela 11 – Resultados Finais da Terceira Bateria de Testes**

| Iteração com Melhor Solução | Melhor Geração da Iteração com Melhor Solução | Custo da Melhor Solução | Melhor Solução Encontrada |
|---|---|---|---|
| 6 | 6 | 20 | 0,2,0,3,2,1,4 |

**Fonte: Autoria própria, 2019.**

Nessa bateria de 100 execuções distribuída em 10 iterações com 10 gerações, é possível perceber que a média dos resultados obtidos e o desvio padrão médio obtido entre as gerações não tem grande variação. Porém os melhores resultados, podem ser considerados como significativos se comparados as médias reduzidas dos desvios padrões, tanto da geração quanto para todas as execuções, e a sua grande maioria estão abaixo dessa distribuição, salientando o fato da implementação da meta-heurística poder encontrar resultados com uma leve significância.

Os resultados anteriores foram todos executados com os mesmos valores de Alfa, Beta e de taxa de evaporação e inicial de feromônio. Utilizando todas as outras variáveis iguais e invertendo os valores de Alfa e Beta obtemos os seguintes resultados

**Tabela 12 – Variáveis de Parametrização Para Quarta Bateria de Testes.**

| V | F | A | B | TeV | ViF | P | G | MA | MVG | Qi | PVS | SPG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 0.1 | 0.9 | 0.9 | 1 | 3 | 10 | Não | MA | 10 | 3 | 3 |

**Fonte: Autoria própria, 2019.**

**Tabela 13 – Resultados e Estatísticas da Quarta Bateria de Testes**

| Iteração | Média das Soluções | Desvio Padrão | Média - Desvio Padrão | Melhor Solução | Geração da Melhor Solução | Caminho da Melhor Solução | Resultado Significativo |
|---|---|---|---|---|---|---|---|
| 1 | 32 | 19,08 | 12,92 | 24 | 1 | 1,4,0,2,0,3,3,1,4 | Não |
| 2 | 43 | 10,97 | 32,03 | 33 | 2 | 0,0,0,3,2,1,4,1,3 | Sim, Local |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **3** | 44 | 8,2 | 35,80 | 33 | 0 | 0,3,2,1,1,4,0,4,0,0,0,0,3 | Sim, Local |
| **4** | 47 | 8,77 | 38,23 | 32 | 8 | 2,0,3,3,2,2,0,2,1,4 | Sim, Local |
| **5** | 43 | 12,14 | 30,86 | 28 | 2 | 1,4,0,0,3,2,0,3,1,4 | Sim, Total |
| **6** | 38 | 8,53 | 29,47 | 20 | 3 | 0,2,0,0,0,0,3,1,4 | Sim, Total |
| **7** | 43 | 12,95 | 30,05 | 20 | 2 | 0,2,0,0,0,3,1,4 | Sim, Local |
| **8** | 42 | 12,18 | 29,82 | 21 | 5 | 0,0,3,2,1,4,0,0,3 | Sim, Local |
| **9** | 38 | 10,22 | 27,38 | 21 | 3 | 2,1,4,0,0,0,3,1,4 | Sim, Total |
| **10** | 42 | 11,98 | 30,02 | 22 | 6 | 0,2,0,3,3,1,4 | Sim, Total |

**Fonte: Autoria própria, 2019.**

**Tabela 14 – Resultados Populacionais da Quarta Bateria de Testes**

| Média das médias | Desvio Padrão Médio | Média das Médias - Desvio Padrão Médio | Média das Gerações com Melhor Solução |
|---|---|---|---|
| 41,20 | 11,502 | 29,7 | 3 |

**Fonte: Autoria própria, 2019.**

**Tabela 15 – Resultados Finais da Quarta Bateria de Testes**

| Iteração com Melhor Solução | Melhor Geração da Iteração com Melhor Solução | Custo da Melhor Solução | Melhor Solução Encontrada |
|---|---|---|---|
| 6,7 | 3 | 20 | 0,2,0,3,2,1,4 |

**Fonte: Autoria própria, 2019.**

A bateria de testes com valores de Alfa baixos e de Beta altos, gerou para essa bateria de 100 execuções resultados com média e desvio padrões menores que quando esses valores foram trocados. A geração que trouxe melhores respostas também reduziu, porém, a melhor solução encontrada foi de mesmo custo, mas duas iterações distintas obtiveram esse resultado.

Uma outra bateria de testes foi feita com valores equivalentes de Alfa e beta.

**Tabela 16 – Variáveis de Parametrização Para Quinta Bateria de Testes.**

| V | F | A | B | TeV | ViF | P | G | MA | MVG | Qi | PVS | SPG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 1 | 1 | 0.9 | 1 | 3 | 10 | Não | MA | 10 | 3 | 3 |

**Fonte: Autoria própria, 2019.**

**Tabela 17 – Resultados e Estatísticas da Quinta Bateria de Testes**

| Iteração | Média das Soluções | Desvio Padrão | Média - Desvio Padrão | Melhor Solução | Geração da Melhor Solução | Caminho da Melhor Solução | Resultado Significativo |
|---|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| **1** | 40 | 9,62 | 27 | 6 | 2,2,0,4,1,3,1,4 | Sim, Total |
| **2** | 41 | 9,09 | 29 | 4 | 2,2,0,3,2,2,1,4 | Sim, Local |
| **3** | 41 | 13,52 | 27 | 0 | 2,1,0,0,0,0,0,0,3,1,4 | Sim, Total |
| **4** | 41 | 11,26 | 27 | 8 | 2,1,0,0,0,0,0,0,3,1,4 | Sim, Total |
| **5** | 46 | 15,08 | 21 | 2 | 0,2,0,0,3,2,1,4 | Sim, Total |
| **6** | 35 | 11,36 | 16 | 5 | 2,0,0,3,1,4 | Sim, Total |
| **7** | 40 | 6,84 | 28 | 8 | 0,0,0,4,0,0,2,0,0,3,1,4 | Sim, Total |
| **8** | 35 | 9,43 | 22 | 0 | 0,0,3,2,1,4,0,0,3 | Sim, Total |
| **9** | 42 | 8,76 | 25 | 8 | 0,2,0,2,0,3,2,1,4 | Sim, Total |
| **10** | 37 | 7,61 | 22 | 5 | 2,2,0,0,3,1,4 | Sim, Total |

**Fonte: Autoria própria, 2019.**

**Tabela 18 – Resultados Populacionais da Quinta Bateria de Testes**

| Média das médias | Desvio Padrão Médio | Média das Médias - Desvio Padrão Médio | Média das Gerações com Melhor Solução |
|---|---|---|---|
| 39,80 | 10,26 | 29,6 | 3,6 |

**Fonte: Autoria própria, 2019.**

**Tabela 19 – Resultados Finais da Quinta Bateria de Testes**

| Iteração com Melhor Solução | Melhor Geração da Iteração com Melhor Solução | Custo da Melhor Solução | Melhor Solução Encontrada |
|---|---|---|---|
| 6 | 5 | 16 | 2,0,,0,3,1,4 |

**Fonte: Autoria própria, 2019.**

Essa bateria de testes trouxe as menores médias e os menores desvios padrões e o melhor resultado de todas as baterias. Os resultados inclusive todos estão fora da distribuição esperada da média e do desvio padrão local, sendo apenas um valor não aplicável a média e desvio padrão total da iteração. Salientando que não são meramente frutos de uma distribuição aleatória.

## 5.3 RESULTADOS DE TEMPO DE EXECUÇÃO E ESCALABILIDADE

Certos parâmetros são responsáveis por alterações no tempo de máquina utilizado para a execução do artefato de programação. Em especial o foco dos testes foi sobre: Número de vértices (V); Número de formigas (F); Gerações(G) ; Quantidade de Iterações(Qi); Limite Passos Vértices Para Achar Solução (PVS), Limite Mínimo De Soluções Para Ir Para Próxima Geração (SPG). O impacto desses parâmetros e demonstrado nos resultados a seguir

Essa bateria de testes foi feita alterando as variáveis listadas anteriormente (V),(F),(Qi),(PVS),(SPG). Como a análise desses resultados o foco é o tempo de execução não serão mostrados os problemas gerados (somente a quantidade de problemas) nem os grafos resultantes, por não serem pertinentes para essa análise. Dessa forma os parâmetros iniciais utilizados foram os seguintes

**Tabela 20 – Variáveis de Parametrização Para Sexta Bateria de Testes.**

| V | F | A | B | TeV | ViF | P | G | MA | MVG | Qi | PVS | SPG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 1 | 1 | 0.9 | 1 | 3 | 10 | Sim | Não | 10 | 3 | 3 |

**Fonte: Autoria própria, 2019.**

**Tabela 21 – Resultados e Estatísticas da Sexta Bateria de Testes**

| V | F | G | Qi | PVS | SPG | Solução | Tempo de Execução (Segundos) |
|---|---|---|---|---|---|---|---|
| 5 | 5 | 10 | 10 | 3 | 3 | Sim | 0 |
| 10 | 10 | 10 | 10 | 3 | 3 | Sim | 0 |
| 50 | 5 | 10 | 10 | 3 | 3 | Sim | 1 |
| 50 | 10 | 10 | 10 | 3 | 3 | Sim | 1 |
| 50 | 50 | 10 | 10 | 3 | 3 | Sim | 4 |
| 100 | 5 | 10 | 10 | 3 | 3 | Sim | 6 |
| 100 | 10 | 10 | 10 | 3 | 3 | Sim | 6 |
| 100 | 50 | 10 | 10 | 3 | 3 | Sim | 11 |
| 100 | 100 | 10 | 10 | 3 | 3 | Sim | 28 |
| 150 | 5 | 10 | 10 | 3 | 3 | Sim | 17 |
| 150 | 10 | 10 | 10 | 3 | 3 | Sim | 20 |
| 150 | 50 | 10 | 10 | 3 | 3 | Sim | 31 |
| 150 | 100 | 10 | 10 | 3 | 3 | Sim | 42 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 150 | 150 | 10 | 10 | 3 | 3 | Sim | 59 |
| 200 | 5 | 10 | 10 | 3 | 3 | Sim | 31 |
| 200 | 10 | 10 | 10 | 3 | 3 | Sim | 36 |
| 200 | 50 | 10 | 10 | 3 | 3 | Sim | 63 |
| 200 | 100 | 10 | 10 | 3 | 3 | Sim | 101 |
| 200 | 150 | 10 | 10 | 3 | 3 | Sim | 98 |
| 200 | 200 | 5 | 1 | 3 | 3 | Sim | 6 |
| 200 | 200 | 5 | 10 | 3 | 3 | Sim | 71 |
| 200 | 200 | 10 | 1 | 3 | 2 | Sim | 13 |
| 200 | 200 | 10 | 10 | 3 | 2 | Sim | 124 |
| 200 | 200 | 20 | 1 | 3 | 3 | Sim | 27 |
| 200 | 200 | 20 | 10 | 3 | 3 | Sim | 281 |
| 500 | 1 | 1 | 1 | 1 | 1 | Não | 0 |
| 500 | 1 | 1 | 1 | 2 | 1 | Sim | 0 |
| 1000 | 1 | 1 | 1 | 2 | 1 | Não | 9 |
| 1000 | 1 | 1 | 1 | 3 | 1 | Sim | 16 |
| 1000 | 10 | 1 | 1 | 3 | 1 | Sim | 24 |
| 1000 | 10 | 5 | 1 | 3 | 1 | Sim | 89 |

**Fonte: Autoria própria, 2019.**

Estes resultados mostram algumas das interações entre os parâmetros para a velocidade da aplicação em achar soluções, e salientam o fato que dependendo da parametrização existem cenários onde problemas podem não ser resolvidos. Um fator que deve ser também considerado se uma análise for feita sobre a resolução ou não dos problemas é sobre a quantidade de problemas, porém esse trabalho não abordou números de problemas maiores que três.

Os resultados mostram que o problema DARP pode ser resolvido em uma representação simplificada estática, para casos com múltiplos usuários, em instâncias de diversos tamanhos (3-1000) com valores de parâmetros diversos, com algumas consequências da alteração dessa padronização sendo perceptível. Os parâmetros afetam a qualidade das soluções e o tempo de execução. Observa-se também, que houve a utilização de caminhos descobertos previamente na construção de soluções melhores em gerações posteriores.

Como não há tratativa de ciclos na implementação atual, alguns resultados são potencialmente irracionais se aplicados ao comportamento de um motorista. A presença de ciclos em um trabalho que vise uma aplicação real pode ser tratada por uma busca local que é parte da meta-heurística, infelizmente esse trabalho não pode implementar tal característica no artefato codificado por questões de escopo.

O aumento em parâmetros como vértices, formigas, gerações e quantidade de iterações, gerou um aumento no tempo de resolução do problema, tal como esperado da complexidade do problema. O aumento de tempo de execução pode ser balanceado caso se tenha interesse em resolver instâncias com muitos vértices com um número reduzido de formigas para poder gerar resultados dentro de valores menores. Todavia é esperado que a qualidade das soluções seja impactada, isso não foi analisado nesse trabalho.

A utilização de parametrizações distintas para valores de feromônio como alfa e beta, geraram resultados distintos que influenciam a qualidade das respostas obtidas.

# 6 CONCLUSÃO

A utilização da meta-heurística ACO para resolução de problemas DARP estáticos se mostrou viável como um exercício da aplicação de conceitos existentes para a resolução de um problema teórico com potencial aplicações reais e mostrou a importância na utilização de agentes inteligentes para lidar com um problema que é NP-Completo.

A meta-heurística atinge resultados satisfatórios, se considerarmos como tal, resultados que atendam o problema e que sejam melhores do que a média - o desvio padrão das amostras. Fazendo com que os agentes explorem o ambiente sem possuir inteligência individual, porém as marcações nas arestas garantem que as decisões desses agentes criem soluções que representam saída que mostram que o sistema construiu de forma inteligente, com otimizações e atingindo objetivos específicos.

O universo de valores possíveis para todos os parâmetros é grande demais para ser abordado somente por um trabalho, dessa forma a análise dos resultados é baseada em um pequeno subconjunto de dados para analisar apenas alguns aspectos da solução. Os elementos não analisados nesse trabalho podem ser aprofundados em estudos semelhantes por outras partes, ou em trabalhos futuros do autor deste. Buscando dessa forma demonstrar, em maior detalhe, como cada um dos elementos da parametrização podem afetar os resultados.

Mesmo assim algumas questões podem ser abordadas relativas ao impacto de alguns valores de parâmetros. Como, por exemplo, alterações nos valores de Alfa e Beta geraram resultados com médias e desvio padrões diferentes, e como quando Alfa e Beta são definidos o mesmo valor de 1, a bateria de testes apresentou os melhores resultados, potencialmente devido a um balanceamento entre explorar caminhos alternativos e seguir caminhos de gerações anteriores. Inclusive reduzindo a existência de ciclos, que é um fenômeno indesejado da codificação atual a qual não foi tratada, porém um método corretivo não deve ser de difícil concepção havendo mais tempo, e a sua ausência não inviabilizou a demonstração do potencial da aplicação, haja vista que não existe software de complexidade acima de trivial sem

bugs. Valores de alfa mais altos que beta não geram a esperada convergência, este fato pode ser devido a uma grandeza não suficientemente grande do valor, gerando uma discrepância maior, apesar que no primeiro teste uma convergência foi observada e em outros testes valores de resolução se repetiram em alguns momentos. Fatos como o pequeno universo de valores de testes, ou a fórmulas utilizadas na implementação também podem ter sido responsáveis pelo observado. O presente trabalho não busca explicar essas questões, somente salienta que valores diferentes de Alfa e Beta são fatores que impactam a qualidade das respostas encontradas, porém análises mais profundas e específicas da natureza dessas respostas e quais são os melhores valores com base na implementação atual podem ser alvo de estudos futuros.

Outra questão pertinente, é relativa à análise do tempo de execução da solução. Uma correlação se faz presente através da alteração dos parâmetros como, em especial com o número de vértices, número de formigas, gerações, quantidades de iterações e valores limítrofes para condições de parada como quantidade de formigas com conjuntos soluções para os problemas propostos. Esses valores podem aumentar o tempo de execução, como também reduzir, sendo a calibração entre eles importante não somente para a velocidade, pois é esperado que a qualidade de soluções que por exemplo utilizem somente uma formiga seja potencialmente afetada, mesmo que essa análise não seja alvo desse trabalho. Conclui-se apenas a correlação entre aumento do tempo e aumento de parâmetros como vértices, formigas, gerações, iterações, com redução do tempo em caso de alteração de alguns deles, como a quantidade de agentes com um conjunto solução aos problemas apresentados, sendo que uma implementação para resolução de problemas reais deve buscar balancear os parâmetros para o seu caso específico, e buscar um equilíbrio entre a qualidade das soluções e  a velocidade do caso de uso relativo ao tempo de espera para atingir elas.

Apesar de algumas tabelas terem elementos de estatística básica como média e desvio padrão, os resultados apresentados não foram coletados nem avaliados com uma abordagem qualitativa estatística robusta, para tal necessita-se uma

implementação que gere uma saída permitindo esse tipo de análise para quantidades significativas de testes, o que não foi coordenado na arquitetura da codificação, pois não é viável a execução na ordem de milhares ou milhões de resultados e assim fazer uso de ferramentas matemáticas muito mais avançadas. Isto não foi o objetivo nem o escopo desse trabalho, mesmo assim, com os resultados mostrados focando em uma abordagem qualitativa, pode-se mostrar que certos comportamentos estão presentes na meta-heurística e no artefato codificado e dessa forma abrindo portas para novos estudos sobre o tema, caso queiram partir de uma implementação já existente o que pode agilizar muito o tempo de análise.

Otimizações sobre o artefato codificado podem aumentar e melhorar os resultados dos tempos de execução apresentados. Potenciais modificações podem incluir refatorações do código para simplificar métodos e reduzir complexidade, como incluir o processamento paralelizado de comandos para utilizar melhor a capacidade computacional de processadores multi-núcleos.

Este trabalho conclui fundamentalmente que a resolução de problemas de carona do tipo DARP é possível através da meta-heurística ACO, sendo assim uma abordagem válida. Todavia os detalhes de implementação e parametrização são fatores fundamentais que devem ser analisados a fundo para se ter uma profundidade maior sobre os resultados aplicados em cenários específicos, em especial produtos de uso comercial. Sendo que dessa forma as conclusões desse trabalho podem ajudar quem busque resolver problemas semelhantes através dessa meta-heurística.

## REFERÊNCIAS

Ballard, D. H. **An Introduction to Natural Computation (Complex Adaptive Systems)**. MIT Press, Cambridge, Massachusetts, USA: 1999.

Cordeau, Jean-François; Laporte, Gilbert. **The dial-a-ride problem: models and algorithms.** Springer Science_Business Media LLC 5, Maio 2007

Colorni, Alberto; Dorigo, Marcos; Maniezzo Vittorio. **Distributed Optimization by Ant Colonies.** Proceedings of ECAL91 – European Conference on Artificial Life, Paris, France, Elsevier Publishing págs 134-142

Dorigo, Marco; Gambardella, Luca Maria. **Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem.** IEE Transactions on Evolutionary Computation, Vol 1 No 1 Abril 1997

Dorigo, Marco; Blum, Christian. **Ant Colony optimization theory: A survey.** Thoretical Computer Science 344, 2005, págs 243-278

Dorigo, Marco; Birattari, Mauro; Stutzle, Thomas. **Ant Colony Optimization Artifical Ants as a Computational Intelligence Technique.** IEEE Computational Intelligence Magazine, Novembre 2006, págs 28-39

Dorigo, Marco; Stutzle, Thomas. **Ant Colony Optimization: Overview and Recent Advances**, IRIDIA – Technical Report Series 2009-013, Maio 2009

Guo, Yuhan; Goncalves, Gilles; Hsu, Tienté. **A Clustering Ant Colony Algorithm for the Long-Term Car Pooling Problem.** ICSI International Conference on Swarm Intelligence, 2011

Guo, Yuhan; Goncalves, Gilles; Hsu, Tienté. **A Multi-Agent Based Self-Adaptive Genetic Algorithm for the Long-Term Car Pooling Problem** – Springer Science+Bussiness Media, 2013

Hartman, Irith Ben- Arroyo; Keren Daniel; Dbai, Abed Abu; Cohen, Elad; Knapen Luk; Yasar, Ansar-Ul-Haque; Janssens, Davy. **Theory and Practice in Large Carpooling Problems.** 5th International Conference on Ambient Systems, Networks and Technologies, 2014

Lin, Shen. **Computer Solutions of the Traveling Salesman Problem**. The Bell System Technical Journal, Dezembro-1965 , págs 2245-2269

Maniezzo, Vittorio; Carbonaro, Antonella; Vigo, Daniele; Hildman, Hanno. **An ANTS Heuristic for the Long- Term Car Pooling Problem.** Fuzziness and Soft Computing, 2004, Volume 141, págs 411-430

Moura, Keila Rodrigues de; Rodrigues, Sergio Augusto. **Carpooling Como Uma Alternativa Para Melhoria do Trânsito: Aceitabilidade e Características dos Possíveis Usuários**. Tekhne e Logos, Botucatu, SP, Dez-Mar, 2013, v4, n.3

Vendramin, Ana Cristina Barreiras Kochem; Munaretto, Anelise; Delgado, Myriam Regattieri de Biase da Silva; Viana, Aline Carneiro. **CGrAnt: a swarm intelligence-based routing protocol for delay tolerant networks.** Proceedings of the 14th Annual conference on Genetic and evolutionary computation 2012, págs 33-40

Vendramin, Ana Cristina Barreiras Kochem; Pereira, Diogo Augusto B.; Pozo, Aurora. **Inteligência Artificial- Técnica de Busca Local para Melhorar a Meta-heurística de Otimização por Colônia de Formigas no Agrupamento de Instâncias em Bases de Dados.** Congresso Sul Brasileiro de Computação 2010.

Russel, J. Stuart; Norvig, Peter. **Artificial Intelligence - A Modern Approach**, 2002

Yan, Shangyao; Chen, Chun-Ying. **A Model and a Solution Algorithm for the Car Pooling Problem with Pre-Matching Information**. Computers & Industrial Engineering, 2011 Vol 61