UNIVERSIDADE
DE PERNAMBUCO

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Programa de Pós-Graduação Acadêmica em Engenharia de Computação

Renan Costa Alencar

Otimizando o Team Ant Colony System para cálculo de múltiplas rotas para a distribuição de materiais e medicamentos

Dissertação de Mestrado

Recife, outubro de 2018

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Programa de Pós-Graduação Acadêmica em Engenharia de Computação

Renan Costa Alencar

Otimizando o Team Ant Colony System para cálculo de múltiplas rotas para a distribuição de materiais e medicamentos

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-Graduação acadêmico em ENGENHARIA DE COMPUTAÇÃO da Universidade de Pernambuco como requisito parcial para obtenção do título de Mestre em Engenharia de Computação.

Prof. DSc. Carmelo José Albanez Bastos Filho
Orientador

Recife, outubro de 2018

Dados Internacionais de Catalogação-na-Publicação (CIP)
Universidade de Pernambuco

Alencar, Renan Costa
A368o Otimizando o Team Ant Colony System para cálculo de múltiplas rotas para a distribuição de materiais e medicamento. / Renan Costa Alencar. – Recife: UPE, Escola Politécnica, 2018.
62 f.: il.

Orientador: Prof. Dr. Carmelo José A. Bastos Filho

Dissertação (Mestrado – Computação Inteligente) Universidade de Pernambuco, Escola Politécnica de Pernambuco, Programa de Pós-Graduação em Engenharia da Computação, 2018.

1. Problema de Múltiplos Caixeiros Viajantes. 2. Problema da Mochila. 3. Inteligência de Enxames. 4. Logística Hospitalar. I. Engenharia da Computação - Dissertação. II. Bastos Filho, Carmelo José Albanez (orient.). III. Universidade de Pernambuco, Escola Politécnica, Mestrado em Computação Inteligente. IV. Título.

CDD 006.3

UNIVERSIDADE DE PERNAMBUCO

Dissertação do Mestrado apresentada por **Renan Costa Alencar**, à Pós-Graduação em Engenharia de Computação da Escola Politécnica de Pernambuco da Universidade de Pernambuco, sob o título **"Otimizando o Team Ant Colony System para cálculo de múltiplas rotas para a distribuição de materiais e medicamentos"**, orientado pelo Prof. Carmelo José Albanez Bastos Filho – Doutor, onde foi aprovado pela Banca Examinadora formada pelos professores:

Bruno José Torres Fernandes - Doutor
(primeiro examinador)

Hugo Valadares Siqueira- Doutor
(segundo examinador)

Carmelo José Albanez Bastos Filho - Doutor
(terceiro examinador)

Visto e permitido a impressão.

Recife, 26 de outubro de 2018.

Prof. Byron Leite Dantas Bezerra
Coordenador da Pós-Graduação em Engenharia de Computação da
Escola Politécnica de Pernambuco da Universidade de Pernambuco

Dedico este trabalho primeiramente ao Pai Celestial, por ser essencial em minha vida, meu guia, socorro presente na horas de angústia, aos meus pais, aos meus irmãos, aos amigos que fiz nesta jornada e ao meu amor.

"Deus nos concede, a cada dia, uma página de vida nova no livro do tempo.
Aquilo que colocarmos nela, corre por nossa conta."
Chico Xavier

Resumo

A distribuição de remédios e suprimentos hospitalares é considerado um problema complexo e difícil de resolver. Dentro da logística hospitalar, esse problema está associado ao MTSP (*Multiple Traveling Salesman Problem*) e ao KP (*Knapsack Problem*). Os problemas do MTSP visam minimizar o deslocamento total dos caixeiros, com uma restrição de que os caminhos devem começar e terminar no depósito e todos os nós intermediários devem ser visitados uma única vez. Por outro lado, as instâncias do KP visam maximizar a capacidade de uma mochila através de objetos previamente selecionados. Esses problemas podem ser resolvidos usando o *Team Ant Colony Optimization* (TACO), uma variação do algoritmo *Ant Colony System* (ACS), baseado no comportamento das colônias de formigas. O TACO possui N equipes com m elementos, em que cada formiga de uma equipe corresponde a um vendedor na construção de uma solução para os problemas de MTSP ou KP. No que toca aos resultados, a abordagem acima foi promissora para dois cenários: minimizar a maior rota, alocar uniformemente a carga de trabalho a todos os caixeiros e minimizar o custo total das rotas, ou seja, a soma de todos os custos de rota de caixeiros individuais. No entanto, esses objetivos são concorrentes. O presente trabalho propõe o uso de algoritmos de otimização de enxames como otimizadores globais para obter melhores resultados do que os achados anteriormente. O algoritmo TACO usa esses algoritmos como otimizadores globais para ajustar os seus parâmetros e, consequentemente, melhorar os resultados para os objetivos já mencionados. Os resultados para o uso de otimizadores globais foram promissores para a otimização dos objetivos abordados pela TACO.

Palavras-Chave: Problema de Múltiplos Caixeiros Viajantes, Problema da Mochila, Inteligência de Enxames e Logística Hospitalar.

Abstract

Distributing medicine and hospital supplies is considered a complex and hard problem to solve. Within hospital logistics, that problem is associated with the Multiple Traveling Salesman Problem (MTSP) and the Knapsack Problem (KP). MTSP problems aim to minimize the total displacement of the salesmen, with a constraint that the paths must begin and end in the depot and all intermediate nodes should be visited once. In the order hand, KP instances aim to maximize the capacity of a sack through previously selected objects. Those problems can be solved using the Team Ant Colony Optimization (TACO), a variation of the Ant Colony System (ACS) algorithm, based on ant colony behavior. TACO has N ant teams with m elements, where each ant of a team corresponds to a salesman in the construction of a solution to the MTSP or KP problems. In the results, the above approach was promising for two scenarios: minimizing the largest route, uniformly allocating the workload to all salesmen, and minimizing the total cost of routes, i.e., the sum of all route costs of individual salesmen. However, these objectives are concurrent. This work proposes the use of swarm optimization algorithms as global optimizers to obtain better results than those previous findings. TACO algorithm uses those algorithms as global optimizers to adjust its parameters and consequently to improve the results for those objectives already mentioned. The results for the use of global optimizers were promising for the optimization of the objectives tackled by TACO.

Keywords: Multiple Salesmen Travelling Problem, Knapsack Problem, Swarm Intelligence and Hopistal Logistics.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de símbolos

$\tau_{ij}(t)$	Quantidade de feromônio depositado no caminho para o ACO
α	Coeficiente de peso do feromônio para o ACO
β	Coeficiente de visibilidade do feromônio para o ACO
$q0$	Porcentagem de escolhas aleatórias para o ACO
ρ	Coeficiente de persistência do feromônio para atualização global para o ACO
ξ	Coeficiente de persistência do fermônio para atualização local para o ACO
$CP(k)$	Custo Parcial Total para o TACO
P_{best}	Melhor posição local para o PSO
G_{best}	Melhor posição global para o PSO
$\vec{x}_i$	Vetor posição da particula no espaço de busca para PSO, FSS e MOFSS
$\vec{v}_i$	Vetor velocidade da particula no espaço de busca para PSO
Δf_i	Variação da função de aptidão para o FSS
$\vec{B}$	Baricentro do cardume no FSS
D_i	Grau de dominância do MOFSS
$\vec{I}$	Vetor de influência coletiva instintiva para o FSS
$\vec{n}_i$	Nova posição do peixe para o FSS
S_i	Passo inicial para o FSS
S_{ind}	Passo individual para o FSS
S_f	Passo final para o FSS
V_i	Fator de influência coletiva volitiva para o MOFSS
w_i	Peso do peixe i no FSS

Tabela de Símbolos e Siglas

AFSA	Artificial Fish-Swarm Algorithm
ACO	Ant Colony Optimization
ACS	Ant Colony System
CAF	Centro de Abastecimento Farmacêutico
CEC	Conference on Commerce and Enterprise Computing
CTR	Custo Total das Rotas
DP	Desvio Padrão
EA	Arquivo Externo
FSS	Fish School Search
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedures
IE	Inteligência de Enxames
KP	Knapsack Problem
MGA	Modified Genetic Algorithm
MOEA/D	Multiobjective Evolutionary Algorithm Based on Decomposition
MOFSS	Multi-Objective Fish School Search
MTSP	Multiple Travelling Salesmen Problem
OG	Otimizador Global
PRV	Problema de Roteamentode Veículos
PSO	Particle Swarm Optimization
RAF	Regra de Atualização do Feromônio

RTE	Regra de Transaçãode Estado
RML	Rota Mais Longa
SMPSO	Speed-constrained Multi-objective PSO
SPEA2	Strength Pareto Evolutionary Algorithm
STACS	Single Team Ant Colony System
STRBAS	Single Team Rank-Based Ant System
SUS	Sistema Único de Saúde
TACO	Team Ant Colony Optimization
TSP	Travelling Salesman Problem
TSPLIB	Travelling Salesman Problem Library
UKP	Unbounded Knapsack Problem

Agradecimentos

Primeiramente ao Pai Celestial que permitiu que tudo isso acontecesse, ao longo de minha vida, e não somente nestes anos como mestrando, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

Meus estimados agradecimentos:

À UPE pelo ambiente criativo e amigável que proporciona.

Ao meu orientador prof. Carmelo José Albanez Bastos Filho, pelo empenho dedicado à elaboração deste trabalho, pelas palavras amigas, pela força e motivação que me fizeram enxergar a luz no fim do túnel e, principalmente, a paciência dispendida durante o processo.

Agradeço a minha mãe, heroína que me dá apoio, incentivo nas horas difícies, de desânimo e cansaço.

Ao meu pai, por quem tomo como exemplo de vida, quem me ensinou a sempre fazer o correto e por ter me dado a melhor herança da vida terrena: o conhecimento.

Obrigado aos meus irmãos, que nos momentos de minha ausência dedicados à vida acadêmica e ao trabal, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente.

Aos amigos que fiz durante esta jornada e aos amigos que sempre me acompanham nas minhas jornadas de vida.

Ao meu amor que é a minha base mais sólida, que sempre me motiva e que sempre divido os bons e os momentos difíceis.

Capítulo 1

Introdução

O Problema do Caixeiro Viajante (*Travelling Salesman Problem* - TSP) e o Problema da Mochila (*Knapsack Problem* - KP) são dois dos problemas de otimização combinatória mais estudados nas últimas quatro décadas, de acordo com Augustine (2002) [1] Bektas (2006) [2], Lin (2007) [3] e Vallivaara [4]. Estes autores afirmam que tais problemas se destacam de outros problemas combinatórios por serem fáceis de projetar, mas difíceis de resolver. Assim, o TSP e o KP pertencem ao conjunto de problemas NP-completos. Embora sejam difíceis de resolver, mesmo separadamente, existem alguns problemas reais que podem ser mapeados como uma combinação TSP-KP, resultando em uma tarefa complexa.

De acordo Bektas (2006) [2], há uma infinidade de problemas da vida real que podem ser encarados como problemas MTSP (*Multiple Trvaelling Salesmen Problem*, uma variação multi-objetiva do problema clássico TSP), e.g., na resolução de PRVs (Problema de Roteamento de Veículos) com adição de restrições às soluções. Na definição básica do MTSP, há $m > 1$ caixeiros inicialmente posicionados numa mesma cidade, definida como depósito. As demais cidades da instância são denominadas de intermediárias. O MTSP tem por objetivo minimizar o deslocamento total dos caixeiros, com a restrição de que os caminhos devem começar e terminar no depósito e todas as cidades intermediárias devem ser visitadas uma única vez. No caminho de cada caixeiro deve existir ao menos uma cidade além do depósito.

O KP é um problema combinatório, que aloca espaço em uma mochila, com antecedência, de acordo com uma seleção de objetos. Assim, o valor total de todos os objetos escolhidos é maximizado na mochila. Lin (2007) [3] afirma que o KP é um problema muito frequente e que aparece no mundo real em diversos seguimentos, e.g., no planejamento econômico e na indústria, como problemas de carga, corte de estoque e empacotamento de lixo. Martello e Toth (1990) [5] definem o problema do KP como um vetor n-objeto de variáveis binárias x_i ($i = 1, \ldots, n$), no qual o objeto i tem um peso w_i e a mochila tem uma capacidade M. Se uma fração x_i; $0 \leq xi \leq 1$ é colocado na mochila, então um lucro, $g_i x_i$, é o ganho. O KP visa encontrar uma combinação de objetos que maximize o lucro total de todos os objetos escolhidos na mochila. Enquanto a capacidade da mochila é M, o peso total de todos os objetos escolhidos deve ser no

máximo M.

Embora ter esses dois problemas associados não seja incomum, eles podem aparecer em cenários complexos. Por exemplo, a distribuição de medicamentos em grandes centros hospitalares pode ser vista como uma instância do MTSP-KP. Tanto a separação (KP) quanto a distribuição (MTSP) têm um alto número de combinações e, mesmo separadas, essas tarefas são difíceis de serem otimizadas, exigindo ferramentas sofisticadas para lidar com elas. Como garantidor da segurança do paciente, Ribeiro (2005) [6] explica que a logística hospitalar é um dos maiores desafios encontrados pelos gestores dos hospitais, principalmente no que diz respeito ao atendimento das necessidades organizacionais de forma rápida, correta e eficiente. Além disso, Souza et al. (2013) [7] explica que, para um hospital público no Brasil, em que o orçamento se restringe aos recursos financeiros disponibilizados pelo Sistema Único de Saúde (SUS), é fundamental que esses recursos sejam implantados com eficiência.

Algumas abordagens, baseadas em métodos matemáticos, algoritmos evolutivos e inteligência de enxames, são usadas para otimizar instâncias de MTSP e KP. É possível citar os seguintes exemplos: Algoritmo Genético - GA (HOLLAND, 1992) [8], *Ant Colony Optimization* - ACO (DORIGO et al., 2008) [9] e *Artificial Bee Colony* - ABC (KARABOGA, 2005) [10]. Essas abordagens visam resolver problemas de produção de aço (TANG et al., 2000) [11], distribuição de cigarros (LIU et al., 2009) [12], ordens de serviço (BARBOSA; KASHIWABARA, 2015, 2016) [13, 14], roteamento de redes de sensores (WANG et al., 2007) [15], entre outros.

Durante a revisão sistemática sobre o tema, não foram encontrados estudos relacionados ao uso de processos de otimização global para melhorar o desempenho de meta-heurísticas implementadas para resolver o problema MTSP-KP. Assim, ainda há espaço para otimizar os parâmetros do MTSP-KP com algoritmos meta-heurísticos visando atingir objetivos específicos, especialmente quando é necessário balancear o tamanho das rotas e o número de entregas por caixeiro simultaneamente. Logo, este trabalho propõe uma metodologia para otimizar os parâmetros do MTSP através de otimizadores baseados em população global, e.g., algoritmos baseados em inteligência de enxames. Utilizamos como estudo de caso um processo de distribuição de medicamentos com múltiplas rotas.

1.1 Motivação

Uma forma de resolver problemas combinatórios é simplesmente enumerar todas as soluções possíveis e guardar aquela de menor custo. Entretanto, essa é uma ideia ingênua pois para qualquer problema NP-difícil, i.e., um problema útil e real, esse método torna-se impraticável, já que o conjunto de soluções possíveis é inviável de ser calculado. Mesmo que seja utilizado um supercomputador para resolver o problema, o tempo de processamento pode levar várias horas, ou vários dias ou até anos. Como consequência, os problemas práticos dessa natureza fazem parte do cotidiano da sociedade moderna. Os problemas aqui tratados são

conhecidos tecnicamente por NP-difícil. Portanto, técnicas computacionais mais apuradas são necessárias para resolver os problemas desta classe.

Há uma infinidade de tarefas da vida real que podem ser encarados como problemas de Otimização Combinatória. A Otimização Combinatória tem uma gama de aplicações possíveis como na resolução de problemas de Roteirização de Veículos, Ensalamento em Escolas e Universidades (*Timetabling*), Escalonamento de Trabalho Humano, Escalonamento de Tarefas em Máquinas, Projeto de Circuitos Integrados, Projeto de Redes com Restrições de Conectividade, dentro outros citados por Papadimitriou e Steiglitz (1998) [16]. Todavia, problemas do mundo real tendem a envolver vários objetivos que precisam ser resolvidos ao mesmo tempo, e que, em geral, são conflitantes entre si. Nestes casos, de acordo Branke et al. (2008) [17], não será encontrada apenas uma única solução, mas sim varias soluções possíveis, que representarão uma curva, chamada de Soluções de Pareto, definida por Collette e Siarry (2013) [18]. Esta definição é detalhada no capítulo seguinte.

A resolução de um problema de otimização normalmente passa por duas fases. A primeira fase consiste em transformar o problema em um modelo. Posteriormente, na segunda fase, um algoritmo deve ser implementado para resolvê-lo. Nem sempre é possível encontrar a melhor solução de um problema de otimização em tempo razoável por meio de algoritmos exatos. Nesses casos, um bom candidato pode ser suficiente para a aplicação que se tem em mãos. De acordo com Murakami 2008 [19], os Métodos Heurísticos são algoritmos que não garantem encontrar a solução ótima de um problema, mas são capazes de retornar uma solução de qualidade em um tempo adequado para as necessidades da aplicação.

Meta-heurísticas são paradigmas de desenvolvimento de algoritmos heurísticos. Diversas propostas surgiram nos últimos anos impulsionadas pelos problemas pertencentes à classe NP-difícil. Dentre as mais conhecidas podemos destacar Algoritmos Genéticos (GA, *Genetic Algorithms*), inspirados na evolução natural dos seres vivos; Otimização por Colônia de Formigas (ACO, *Ant Colony Optimization*), algoritmos baseados no comportamento das formigas para encontrar comida; *Simulated Annealing*, baseada originalmente em conceitos de Mecânica Estatística considerando a analogia entre o processo físico annealing de sólidos e a resolução de problemas de otimização combinatória; Busca Tabu, utiliza mecanismos de memória e diversificação das soluções como recursos para encontrar a solução ótima; e GRASP (*Greedy Randomized Adaptive Search Procedures*), técnica baseada no paradigma da busca gulosa (ou míope).

O algoritmo FSS (*Fish School Search*), criado por Bastos-Filho [20], foi introduzido em 2008 com o objetivo de abordar tarefas de otimização em espaços de busca multimodais. A principal vantagem deste algoritmo é a capacidade de autorregulação, o trade-off entre exploração e explotação. Alguns resultados preliminares, publicados por Bastos-Filho (2009) [21], indicam que o FSS pode superar alguns dos algoritmos baseados em enxames em funções de benchmark com espaços de busca multimodais. Entretanto, não há variante do algoritmo que trate da

resolução de problemas combinatórias.

1.2 Objetivos

O problema de logística na distribuição de medicamentos em hospitais de grande porte apresenta múltiplos demandantes (farmácias menores) bem como múltiplos despachantes (entregadores), tendo um único depósito chamado de CAF – Centro de Abastecimento Farmacêutico. Logo, pode ser abordado de três modos distintos: agrupando os demandantes e em seguida encontrando os caminhos para distribuição; encontrando os caminhos de distribuição primeiro e depois distribuindo os demandantes; ou realizando um processo de agrupamento e roteamento em conjunto. Para este trabalho será utilizado a primeira abordagem devido a sua facilidade na resolução da tarefa foco deste trabalho haja vista que a segunda abordagem faz a associação entre a resolução de problemas MKP (Problema de Múltiplas Mochilas) com instâncias MTSP que será tratado em trabalhos futuros.

Neste trabalho é proposto o uso de uma metodologia para resolução de problemas de MTSP, que visa a distribuição de medicamentos, otimizando a programação de percursos para a entrega. A partir de um cenário hospitalar, são construídas instâncias de MTSP. As posições das entregas a serem executadas representam as cidades da instância e os entregadores de pedidos representam os caixeiros.

Estas instâncias são otimizadas pela abordagem TACO – *Team Ant Colony Optimization*, algoritmo proposto por Vallivaara (2008) [4], para distribuição de medicamentos a material hospitalar bem como os percursos a serem realizados pelos entregadores. As melhores soluções geradas pela TACO são utilizadas para definir a distribuição das ordens entre os entregadores e os percursos que devem ser realizados.

Ao fim do processo de encontro das soluções, um otimizador global é utilizado, baseado em algoritmos de base populacional como os algoritmos de enxames. Esta associação visa otimizar soluções previamente encontradas em relação minimização do custo no deslocamento dos entregadores (minimização do tempo de entrega) bem como o balanceamento de atendimentos realizados por eles (maximização do número de atendimento).

A abordagem TACO foi escolhida devido aos resultados encorajadores para o MTSP e vem sendo aplicado com sucesso para vários problemas NP-difíceis (problemas não polinomiais) de otimização combinatória; e a fim de explorar novas formas de aplicá-la ao MTSP. Também os algoritmos de enxames foram escolhidos devido as suas características de grande poder de otimização dos parâmetros inerentes ao TACO.

A partir do problema exposto na seção anterior e da proposta acima descrita, esta pesquisa tem como objetivo fomentar os estudos escassos na resolução de problemas de logística voltados a área hospitalar e que possa contribuir para a melhoria da gestão. É objetivo também desta

pesquisa contribuir com a resolução de problemas combinatórios como o MTSP e o campo dos métodos computacionais, como a Inteligência de Enxames.

1.3 Estrutura da Dissertação

Esta dissertação está estruturada em 7 capítulos, iniciando neste Capítulo 1 com uma introdução sobre a proposta apresentada. No Capítulo 2, é introduzido uma descrição básica dos problemas TSP e KP e suas variantes. No Capítulo 3, é descrito alguns algoritmos baseados em inteligência de enxame para resolver problemas combinatórios. No Capítulo 4, é apresentado os trabalhos relacionados envolvendo algoritmos meta-heurísticos para resolver instâncias de MTSP e KP. No Capítulo 5, são explanados o modelo proposto, a instância do problema a ser otimizada e os cenários e suas configurações para minimizar a instância. No Capítulo 6, é apresentada uma discussão dos resultados otimizados fazendo uma comparação entre os otimizadores globais baseados em população. Finalmente, no Capítulo 7, o último deste trabalho, é destacado algumas conclusões sobre o modelo proposto e trabalhos futuros.

Capítulo 2

Problemas Combinatórios

Lacerda (2012) [22] define que os problemas de Otimização Combinatória, em geral, se resumem a encontrar, dentre todos os possíveis subconjuntos de soluções, aquela cujo o custo seja o menor possível. Isto é, suponha que há um conjunto de itens e uma série de regras que podem ser usadas para selecionar alguns elementos (itens desse conjunto). Usando essas regras, há várias maneiras diferentes de escolher os elementos e criar outros conjuntos menores (subconjuntos). Se cada elemento estiver associado à um custo, os subconjuntos criados também terão um custo que é dado, por exemplo, pela soma dos custos de seus elementos.

Uma forma de resolver tais tarefas seria simplesmente enumerar todas as soluções possíveis e guardar aquela de menor custo, segundo Bektas (2006) [2]. Entretanto, essa é uma ideia ingênua pois para qualquer problema de um tamanho interessante (e útil), esse método torna-se impraticável, já que o número de soluções possíveis é muito grande. Mesmo que, para a sua resolução, se utilize um supercomputador, haveria um custo computacional muito elevado.

Segundo Martello e Toth (1990) [5], modelos baseados em grafos são imensamente utilizados em muitos exemplos de otimização combinatória. Grafo é uma forma de representar um conjunto de elementos e suas relações. Esse recurso é muito utilizado para modelagem de instâncias por ser uma forma bastante intuitiva para representá-las. Além disso, na literatura podem ser encontrados algoritmos para resolver diversos otimizações combinatórias em grafos. Um exemplo de problema modelado em forma de grafo é o do Caixeiro Viajante (do inglês, *Travelling Salesman Problem*). O caixeiro visa encontrar a menor rota entre cidades onde obedeçam às restrições de ser o menor caminho e que ele venha passar no máximo uma única vez em cada cidade do seu trajeto.

Segundo Papadimitriou e Steiglitz (1998) [16], otimização combinatória é a ciência de encontrar a melhor solução de um conjunto grande, mas finito de soluções possíveis. Este grande número de possibilidades vem da necessidade de encontrar um bom subconjunto de elementos de um dado conjunto sujeito a algumas regras. O custo ou qualidade de uma solução é geralmente definido como uma função desses elementos.

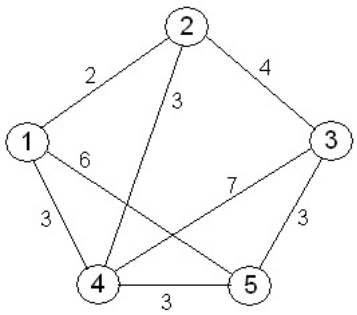
Enquanto na maioria das aplicações práticas de exploração através de todos os casos é apenas uma possibilidade teórica, devido ao seu número imenso, a otimização combinatória tenta oferecer métodos e algoritmos mais sofisticados, resultando em tempos de execução razoável. Algumas das ferramentas mais poderosas para encontrar soluções ótimas são programação linear e integral e programação de restrição.

Para aqueles casos em que essas técnicas não conseguem encontrar boas soluções em tempo razoável, algumas outras abordagens são possíveis, como algoritmos de Aproximação, heurística e métodos de limites duplos, como aponta Malaquias (2006) [23].

2.1 O Problema do Caixeiro Viajante (TSP)

De acordo com Bektas (2006) [2] e Silva (2016) [24], o Problema do Caixeiro Viajante (TSP) é considerado um dos exemplos de otimização combinatória mais estudado por pertencer à classe dos problemas NP-completos. O TSP é de fácil descrição, mas de difícil resolução devido a sua explosão combinatória, ou seja, há zilhões de possibilidades de se resolvê-lo. Tem como desafio encontrar o caminho mais curto visitando cada membro de uma coleção de locais e retornando ao seu ponto de partida (Figura 1). Logo, o torna intratável na obtenção de soluções exatas. Bektas afirma que o problema tem uma aplicabilidade em diversas áreas tais como: indústrias (minimizar custo por meio da melhor rota utilizada por equipamentos no processo de montagem), empresas (transportadoras, entrega e coleta de cargas), automobilística (redução de custo de viagem), transporte de passageiros, roteirização de serviços de reparos ou serviços públicos (como coleta de lixo, entrega postal), sequenciamento de genoma.

Figura 1 – Problema do caixeiro-viajante

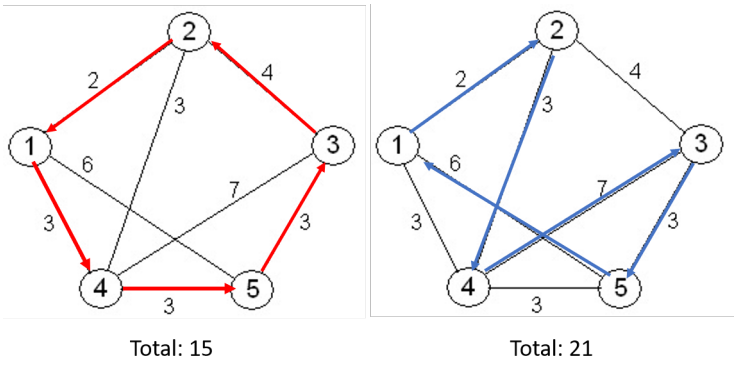


Fonte: Google Images

O TSP pode ser representado por um grafo completo $G = (N, A)$, com N sendo o conjunto de nós, também chamado cidades, e A sendo o conjunto de arcos totalmente conectados aos nós. Cada arco (i, j) pertencente A é atribuído um valor d_ij que representa a distância entre as cidades i e j.

Tomando o exemplo da Figura 1, temos um grafo de cinco vértices em que cada aresta, associada a um vértice, possui um peso. Pensando em encontrar o percurso que passe em todos nós uma única vez, temos duas possíveis soluções ilustradas na Figura 2. É possível perceber que, ao somar o peso de todas as arestas do percurso, a rota ilustrada a esquerda possui menor tamanho se comparado a da direita.

Figura 2 – Visualização de soluções em grafo.



Fonte: O próprio autor

Logo, o TSP é um problema de encontrar um caminho fechado mais curto visitando cada um dos $n = |N|$ nós de G exatamente uma vez. Para insâncias de TSP simétricos, as distâncias entre as cidades são independentes da direção transversal dos arcos, isto é, $D_ij = d_ji$ para cada par de nós. No TSP assimétrico pelo menos para um par de nós temos $d_ij \leq d_ji$.

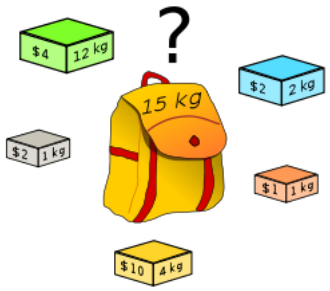
2.2 O Problema da Mochila (KP)

O problema da mochila (KP) pode ser enunciado da seguinte forma (MARTELLO; TOTH, 1990) [5]:

Um viajante levará consigo apenas uma mochila para sua viagem (Figura 3). Sua mochila possui uma dada capacidade e deve ser preenchida com alguns objetos que lhe sejam úteis durante a viagem. Cada objeto possui um peso e um dado valor para o viajante e este possui apenas uma unidade de cada objeto a ser escolhido. Quais objetos devem ser levados pelo viajante em sua mochila de forma a maximizar o valor da mochila?

Seja w_j o peso do j-ésimo objeto e p_j seu valor. Se o objeto x_j aparece na mochila, então $x_j = 1$ caso contrario $x_j = 0$. Se denotarmos por c a capacidade da mochila, e por n a quantidade de objetos disponíveis para escolha e $F_(c)$ como o maior valor obtido para a mochila de capacidade c usando os n objetos, então o problema pode ser formulado algebricamente como a seguir:

Figura 3 – Problema da mochila: Como maximizar o valor com um peso máximo?



Fonte: Google Images

$$F_{(c)} = max \sum_{j=1}^{n} p_j x_j \quad (p_j > 0) \tag{2.1}$$

$$sujeito \quad a \sum_{j=1}^{n} w_j x_j \leq c \quad (w_j, c > 0) \tag{2.2}$$

onde: $x_j = 0$ ou $x_j = 1$.

Esta versão do problema é conhecida como problema da mochila binária (*Binary KP* ou *0-1KP*); Martello e Toth 1990 [5].

Se permitirmos a x_j assumir qualquer valor inteiro maior que zero então temos a forma mais genérica da versão unidimensional do problema da mochila. De acordo com a definição presente no trabalho de Martello e Toth 1990 [5], tal problema é conhecido como problema da mochila ilimitada (*Unbounded Knapsack Problem* ou UKP).

Martello e Toth afirmam que, apesar do nome, o problema da mochila não é útil apenas na solução de problemas de empacotamento. Existem também aplicações na área da criptografia, engenharia naval, gerenciamento de projetos e finanças entre outras.

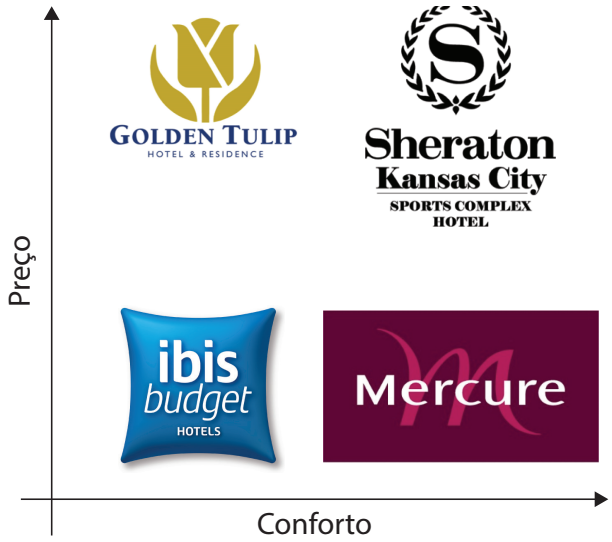
2.3 Otimização e Problemas Multi-Objetivos

É imprescindível o estudo de problemas multi-objetivo visto que a maior parte dos problemas do mundo real tendem a ser classificados desta forma. Augusutine (2002) [1] explica que a complexidade dos problemas reais que surgem nas sociedades tecnológicas modernas é caracterizada pela existência de múltiplas perspectivas de análise, refletindo aspectos econômicos, sociais, políticos, físicos, de engenharia, administrativos, psicológicos, éticos, estéticos, etc.

Bastos-Filhos e Guimarães (2013) [25] resumem o conceito de otimização como sendo "a busca por soluções para determinado problema (ou problemas) que o satisfaçam da melhor maneira possível". Collette e Siarry (2013) [18], em seu livro "Multiobjective Optimization:

Principles and Case Studies", definem um problema de otimização como sendo a busca de um mínimo ou máximo (o ótimo) de uma função. O autor também descreve um caso específico de otimização chamado problema de otimização restrita. Este tipo ocorre quando problemas de otimização para os quais as variáveis da função a serem otimizadas são restritas para evoluir em uma área precisamente definida do espaço de pesquisa.

Figura 4 – Dilema de reserva de um quarto de hotel em função de dois objetivos: preço e conforto.



Fonte: o próprio autor.

Branke et al. (2008) [17] explicam que um problema de otimização mono objetivo envolve uma única função objetivo (ou função *fitness*) e geralmente resulta em uma solução única, chamada de solução ótima. Por outro lado, uma tarefa de otimização multi-objetivo considera vários objetivos conflitantes simultaneamente. Nesse caso, geralmente não há solução ótima única, mas um conjunto de alternativas com diferentes compensações, chamadas de soluções ótimas de Pareto, ou soluções não dominadas.

A Figura 4 apresenta um problema para a reserva de hotel, levando em consideração dois objetivos: preço e conforto. Se a pretensão for por um quarto com maior conforto, o preço será mais alto. Já, se pretende um quarto com menor preço, o conforto também será afetado, diminuindo-o.

Capítulo 3

Algorítmos Meta-Heurísticos e Inteligência de Enxames

As técnicas de Inteligência de Enxames (IE) estão inseridas dentro Computação Evolutiva, apesar de trazer uma metáfora diferente da evolução das espécies. Na IE, o comportamento biológico de seres vivos serve de inspiração o que torna as técnicas de otimização mais simples e fáceis de serem implementadas do que as técnicas evolucionárias.

Chan e Tiwari (2007) [26], em seu livro "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization", expõem que a complexidade crescente dos problemas exigiu que os pesquisadores encontrassem o possível formas de facilitar a solução dos problemas. Segundo os autores, isso motivou os pesquisadores a compreender ideias da natureza e implantá-lo nas ciências da engenharia. Esse modo de pensar levou a surgimento de muitos algoritmos biologicamente inspirados que provaram ser eficientes em lidar com os problemas computacionalmente complexos com competência, tais como Algoritmo Genético (GA), Otimização de Colônia de Formigas (ACO), Otimização de Enxame de Partículas (PSO), etc.

Na IA, de acordo com Eberhart e Kennedy (2001) [27], o comportamento global do enxame é um efeito emergente das interações locais dos membros do enxame. De acordo com Filho et al. [20], as informações pessoais e sociais guardam semelhança com operadores de recombinação e de cruzamento, enquanto que a conservação de movimento da partícula atua como uma espécie de mutação direcional. Há diferenças, no entanto, sendo a maior delas o papel da seleção natural: enquanto que métodos evolutivos tem como parte essencial a morte dos indivíduos menos aptos, em um processo social os indivíduos são preservados durante a execução do algoritmo, de forma que o próprio indivíduo se adapta no decorrer do tempo.

Há uma série algoritmos de otimização de problemas que abordam diferentes grupos de seres vivos, desde pássaros a vagalumes para resolver problemas de otimização. Engelbrecht (2006) [28] exemplifica alguns problemas resolvidos pela IE como aproximação de funções, agrupamento, otimização de estruturas mecânicas, resolução de sistemas de equações, otimização de roteamento em redes de telecomunicações, coloração de grafos, programação e resolução do

problema de atribuição quadrática, dentro outros. Os problemas aqui tratados são conhecidos tecnicamente por NP-difícil. Portanto, técnicas computacionais mais apuradas são necessárias para resolver esses problemas.

Dentro da IE, há várias técnicas que abordam diferentes grupos de seres vivos desde pássaros a bactérias. Entretanto, este trabalho faz uso das seguintes abordagens: *Ant Colony Optimization* (ACO), *Particle Swarm Optimization* (PSO) e *Fish School Search* (FSS). O ACO, Otimização por Colônia de Formigas, tem a sociedade organizada das formigas como modelo para busca de alimentos através de rastros deixados pelas formigas. Já o PSO, Otimização por Enxame de Partículas, tem como inspiração os bandos de pássaro, tendo como a forma de voar em grupo na busca de alimento e mecanismo de interação social. Por fim, o FSS [21], Busca em Cardume de Peixes, tem no movimento do nado dos peixes a inspiração para proteção e busca de locais mais favoráveis para sobrevivência.

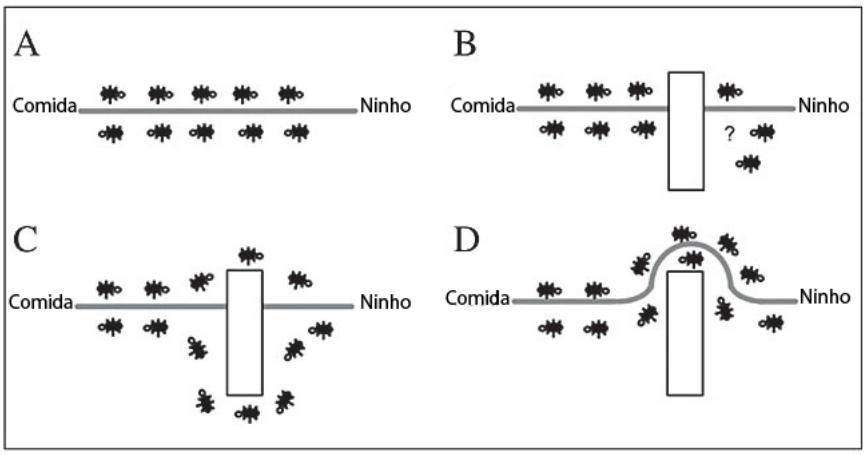
3.1 ACO

O ACO, desenvolvido por Dorigo e sua equipe em 1996 [29], inspirou-se no comportamento de colônias de formigas reais, em particular, por seu comportamento de forrageamento. Uma das principais ideias é a comunicação indireta entre os indivíduos de uma colônia de agentes, chamados de formigas, com base em uma analogia com trilhas de uma substância química, chamada feromônio, que as formigas reais utilizam para a comunicação. As trilhas (artificiais) de feromônio são um tipo de informação numérica distribuída que é modificada pelas formigas para refletir sua experiência acumulada ao resolver um problema particular.

Como descreve Mulati, Constantino e da Silva (2013) [30], foi descoberto que a comunicação entre as formigas que caminhavam pela trilha ocorria por meio de uma substância química, denominada feromônio, depositada por elas próprias. Enquanto as formigas caminham por uma trilha, inicialmente de forma aleatória, elas depositam uma certa quantidade de feromônio no solo. Assim, as próximas formigas tomam a decisão de seguir um caminho com probabilidade proporcional à quantidade feromônio depositada anteriormente. Ao decidir seguir um caminho com a presença da substância, ocorre então um reforço do caminho com o seu próprio feromônio. Este comportamento é denominado de auto-catalítico por ser um processo que reforça a si mesmo. O feromônio depositado tende a evaporar com o tempo, então quanto maior é a concentração de formigas passando pelo mesmo lugar, mais atrativo ele se torna para as próximas formigas. A Figura 5 ilustra um experimento com formigas reais.

As formigas conseguem obter um bom caminho entre dois pontos. Na primeira decisão após a inserção do obstáculo, as quantidades de formigas a escolher a rota mais curta e a mais longa devem ser aproximadamente a mesma, de modo que elas fazem suas escolhas com base no feromônio encontrado, sendo que as probabilidades de ambos os lados devem ter valores próximos um do outro. Porém, na segunda decisão, as formigas que percorreram o trajeto mais

Figura 5 – Ilustração do experimento com formigas



Fonte: Google Images

curto já estão voltando, depositando ainda mais feromônio, enquanto as que foram pelo perscurso mais longo ainda estão completando a primeira transição. Desta forma, as formigas tendem a seguir o mais curto.

De forma análoga, um indivíduo da colônia artificial constrói soluções candidatas, começando com uma solução vazia e, em seguida, adicionando componentes solução de forma iterativa até uma solução candidata completa ser gerada. Após a construção da solução estar completa, as formigas dão feedback das soluções que elas construíram depositando feromônio nos componentes da solução que elas usaram em sua solução. Tipicamente, os componentes da solução que fazem parte de melhores soluções ou são usados por muitas formigas receberão uma maior quantidade de feromônio. Portanto, provavelmente estes componentes serão usados pelas formigas em iterações futuras do algoritmo. Para evitar que a busca fique estagnada, geralmente antes das trilhas de feromônio serem reforçadas, todas as trilhas de feromônio são diminuídas por um fator ρ.

Em geral, todos os algoritmos ACO para combinação combinatória estática seguem um esquema algorítmico específico mostrado no Pseudocódigo 1. Após a inicialização das trilhas de feromônio e alguns parâmetros, um loop principal é repetido até uma condição de término - que pode ser um certo número de construções de soluções ou um determinado limite de tempo de CPU - é cumprido. No loop essencial, primeiro, as formigas constroem soluções viáveis, então as soluções geradas podem ser melhoradas pela aplicação busca local e, finalmente, as trilhas de feromônios são atualizados.

Algoritmo 1 Ant Colony Optimisation

1: Configurar parametros, inicializar as trilhas de feromônio
2: Enquanto o critério de parada não é satisfeito:
3: Para cada uma das formgias(em parelelo):
4: Construi soluções
5: Aplique Busca Local (opcional)
6: Atualize as trilhas

Na ACO, as soluções são geradas num processo iterativo no qual agentes inteligentes, as formigas, percorrem todas as cidades de uma instância de TSP, por exemplo, escolhendo a cada iteração a próxima cidade de sua rota de maneira estocástica através da Regra de Transição de Estado (RTE). A RTE [29] do Ant Colony Optization é mostrada na Equação 3.1:

$$p_i^k = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum [\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}, & \text{se } j \text{ é uma aresta permitida à } k \\ 0, & \text{caso contrário} \end{cases} \tag{3.1}$$

Na Equação 3.1, que determina a probabilidade de um vértice ser escolhido por uma formiga, i representa o vértice no qual a formiga k se encontra, e j um dos possíveis vértices para o qual ela pode se deslocar. Os vértices permitidos à formiga são aqueles ainda não visitados até a fase atual da construção da solução. Cada formiga armazena os nós já visitados em sua memória, que é denominada como lista tabu. $\tau_{ij}(t)$ corresponde à quantidade de feromônio presente na aresta entre os vértices i e j no instante t. η_{ij} corresponde à visibilidade da aresta, que é definida pelo valor $1/c_{ij}$, onde c_{ij} é o custo de deslocamento da aresta. α e β são parâmetros que definem o peso da trilha de feromônio e da visibilidade, respectivamente, na escolha do próximo vértice pela formiga [29].

3.2 ACS

O algoritmo *Ant Colony System*, ou Sistema de Colônia de Formigas, é uma abordagem derivada da Otimização de Colônia de Formigas (ACO) também desenvolvida por Dorigo et al. (2006) [9]. O ACS difere do ACO em três pontoss: utiliza uma RTE mais agressiva; a RAF (Regra de Atualização de feromônio) determina que a evaporação e o depósito de feromônio, ao final de cada ciclo, acontecem apenas nas arestas pertencentes à *best-so-far solution*, ou seja, a melhor solução encontrada até o momento em uma execução do algoritmo; e, a cada vez que uma formiga utiliza uma aresta numa solução, uma quantidade pré-definida de feromônio é removida da aresta, para aumentar a exploração de caminhos alternativos. Outra diferençaa importante é em relação ao número N de formigas que percorrem o grafo e constroem suas próprias soluções simultaneamente: enquanto no ACO o melhor valor encontrado experimentalmente é $N = n$, para o ACS o melhor valor encontrado é $N = 10$ [9].

$$j = \begin{cases} \text{argmax}_{l \in J_k}\{\tau_{il}[\eta_{il}]^{\beta}\}, & \text{if } q \leq q_0 \\ \text{J}, & \text{caso contrário} \end{cases} \tag{3.2}$$

A RTE do ACS está mostrada na equação 10, na qual j representa a cidade escolhida por uma formiga k que se encontra no vértice i ao se movimentar. Esta equação é chamada de regra proporcional pseudo-aleatória, pois o parâmetro q_0 define a porcentagem das escolhas que serão feitas de forma determinística pelas formigas, onde $0 \leq q0 \leq 1$. Sendo q uma variável aleatória uniformemente distribuída em $[0, 1]$ e atualizada a cada movimento, se $q_0 = 1$, todas as escolhas das formigas serão realizadas de forma determinística, pelo valor máximo de $\tau_{il}[\eta_{il}]^{\beta}$

, onde $l \in S_k^i$ corresponde ao conjunto de cidades permitidas à formiga no movimento. Se, ao contrário, $q_0 = 0$, todas as escolhas serão realizadas de forma estocástica, pois J representa uma cidade escolhida através das probabilidades calculadas pela Equação 3.1, com a única diferença que o parâmetro α foi excluído, ou seja, seu valor foi fixado em 1. O melhor valor definido experimentalmente é $q_0 = 0.9$, o que significa que 90% das escolhas serão determinísticas [9].

De acordo com Dorigo et al [9], a RAF do ACS define que a atualização de feromônio ocorre em dois momentos: uma atualização global, ao final de cada ciclo do algoritmo; e uma atualização local, logo depois que uma formiga se move de uma cidade a outra, ou seja, inclui mais uma aresta na sua rota. A regra para Atualização Global de Feromônio (AGF) está apresentada nas Equações 3.3 e 3.4, onde ρ é o coeficiente de persistência do feromônio, cujo melhor valor encontrado para o ACS é $\rho = 0.1$. T^{bs} corresponde ao conjunto de arestas que fazem parte da melhor solução encontrada até o momento atual da execução do algoritmo e C^{bs} corresponde ao custo de T^{bs}.

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \forall(i, j) \in T^{bs}, \tag{3.3}$$

$$\Delta\tau_{ij}^{bs} = 1/C^{bs} \tag{3.4}$$

A regra de Atualização Local de Feromônio (ALF) [9] está apresentada na Equação 3.5, na qual ξ é um parâmetro cujo melhor valor experimentalmente calculado é $\xi = \rho = 0.1$. τ_0 também é um parâmetro que, além de ser utilizado na atualização local de feromônio, também corresponde à quantidade de feromônio depositada em todas as arestas no início da execução do algoritmo. O melhor valor encontrado para τ_0 é calculado a partir do custo de uma solução criada aplicando um algoritmo do vizinho mais próximo à instância em análise. A fórmula para cálculo de τ_0 é mostrada na Equação 3.6, na qual n é igual ao número de cidades da instância e C^{nn} é igual ao custo da solução obtido utilizando um algoritmo do vizinho mais próximo sobre a instância.

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0, \tag{3.5}$$

$$\tau_0 = 1/nC^{nn} \tag{3.6}$$

3.3 TACO

A Otimização de Colônia de Formigas por Times (TACO), proposta por Vallivaara (2008) [4], é baseada no ACS (*Ant Colony System*) para resolver instâncias de MTSP (*Multiple Travelling Salesmen Problem*). Essa generalização básica é feita substituindo as N formigas ACS, que constroem soluções para o TSP, com N equipes de m membros, fazendo um total de formigas igual a $N \times m$. Uma equipe de formigas representa um vendedor na construção da solução MTSP e cada equipe tem sua lista de tabu.

A única diferença entre o ACS e o TACO com $m = 1$ é que no primeiro as formigas podem ser colocadas nos nós aleatoriamente no estágio inicial. Como isso não é possível no TACO e no MTSP, é natural que as formigas sejam colocadas inicialmente nos pontos de partida iniciais [4]. Para distribuir a carga de trabalho, uma formiga com a rota parcial mais curta escolhe sua próxima cidade j, em qualquer momento do processo de construção, de acordo com a equação da Regra de Estado de Transição (RET), conforme mostrado em 3.7.

$$j = \begin{cases} \text{argmax}_{l \in J_k}\{\tau_{il}[\eta_{il}]^\beta\}, & \text{if } q \leq q_0 \\ \text{J}, & \text{caso contrário} \end{cases} \tag{3.7}$$

Durante a fase de inicialização, o nível de feromônio é ajustado para $\tau_0 = 1/nC^{nn}$ em cada aresta, como sugerido no ACS. Antes de começar a construção da rota, todos os nós são configurados como não-visitados por cada equipe. Em seguida, cada membro m de uma equipe N_i é colocado na cidade inicial correspondente v_k, e todas as cidades iniciais são marcadas como visitadas para este mesmo time [4].

Estando uma formiga k posicionada em um vértice v_k; sendo v_0 o depósito da instância; $c(v_i, v_j)$ o custo de deslocamento entre dois vértices v_i e v_j quaisquer; o Custo Parcial total $CP(k)$ da rota já percorrida pela formiga k; e tendo sido escolhido o vértice v_j como seu próximo vértice de acordo com a equação 3.7; o método consiste em verificar, antes da formiga de mover, se existe qualquer outra formiga l da instância que respeite a inequação 3.8. Se houver, então é permitido que a formiga l escolha seu próximo vértice novamente de acordo com a RET presente em 3.7, e se movimente para o novo vértice escolhido [4].

$$c(v_l, v_j) + c(v_j, v_0) + CP(l) < c(v_k, v_j) + c(v_j, v_0) + CP(k) \tag{3.8}$$

TACO tem vários parâmetros que são responsáveis pelo seu comportamento durante a construção de soluções. A probabilidade inicial q_0 determina se a inicialização das formigas tem apenas escolhas determinísticas ou aleatórias $(0 < q_0 < 1)$. Os parâmetros de feromônio α e β definem o peso da trilha de feromônio e a visibilidade, respectivamente, na escolha do próximo nó pela formiga. O parâmetro ξ controla a persistência do feromônio quando a Regra de Atualização de Feromônio (RAF) ocorre localmente, logo após uma formiga se mover de uma cidade para outra, ou seja, inclui mais uma borda em sua rota. Da mesma forma, ρ regula a persistência de feromônios para RAF global, ou seja, no final de cada ciclo do algoritmo.

Finalmente, o algoritmo usa uma abordagem de pesquisa local para melhorar as soluções construídas. A pesquisa local opt-2 altera quaisquer duas arestas de uma solução e verifica se ela obtém alguma melhoria. O procedimento opt-2 é aplicado a todas as soluções construídas [4].

3.4 PSO

Kennedy e Eberhart (1999) [31] propuseram o método *Partition Swarm Optimization* (PSO) baseado no revoada das aves. O PSO é adequado para a otimização de variáveis contínuas em um espaço de busca de alta dimensão e apresenta alta precisão. Ele realiza a pesquisa por meio de um enxame de partículas por meio de um processo de iteração.

Cada partícula se move em direção a sua melhor posição (P_{best}) anterior e a melhor posição global (G_{best}) no enxame para alcançar a solução ótima, como mostrado na Equação 3.9.

$$\vec{x}_{i(t+1)} = \vec{x}_{i(t)} + \vec{v}_{i(t)} \tag{3.9}$$

A solução representa a posição da partícula no espaço de busca, um vetor $\vec{x}_i$. Para cada etapa, as partículas têm suas posições de acordo com seu vetor de velocidade $\vec{v}_i$, como mostrado na Equação 3.10, encontrado no trabalho de Shi [32].

$$\vec{v}_j = \omega\vec{v}_j + c_1 r_1(P_{best} - \vec{x}_j) + c_2 r_2(G_{best} - \vec{x}_j) \tag{3.10}$$

onde c_1 e c_2 são números positivos constantes (sendo c_1 é referente ao comportamento cognitivo e c_2 ao comportamento social); r_1 e r_2 são números aleatórios entre o intervalo $[0, 1]$.

A velocidade de fixação, um limite superior para o parâmetro de velocidade evita que as partículas voem para fora do espaço de busca. Da mesma forma, a estratégia do "coeficiente de constrição", proposta por Clerc e Kennedy (2008) [33], constrói as velocidades através da análise dinâmica de enxames.

A inércia, mostrada na primeira parte da Equação 3.11, representa a velocidade anterior, que fornece o momento necessário para as partículas percorrerem o espaço de busca.

$$\vec{v}_{ij}(t+1) = \chi[\vec{v}_{ij}(t+1) + \varphi_1(\vec{v}_{ij}(t) - \vec{x}_{ij}(t)) + \varphi_2(\hat{y}_{ij}(t) - \vec{x}_{ij}(t))] \tag{3.11}$$

Por outro lado, o componente cognitivo, a Equação 3.12, determina o movimento individual de cada partícula. Ele incentiva as partículas a se moverem em direção às suas melhores posições atuais.

$$\chi = \frac{2}{4 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{3.12}$$

A último a Equação 3.13, o componente "social", implica o efeito colaborativo das partículas para encontrar a solução global ótima.

$$\varphi = \varphi_1 + \varphi_2 \tag{3.13}$$

onde $\varphi_1 = c_1$ e $\varphi_2 = c_2$.

3.5 FSS

Bastos-Filho et al. (2008) [20] desenvolveu um algoritmo de busca de base populacional inspirado no comportamento de natação de peixes, que se expande e se contrai enquanto procura comida. O algoritmo *Fish School Search* (FSS) considera os movimentos individuais e coletivos dos peixes. Esse algoritmo de otimização não apresenta a mesma capacidade de exploração do PSO, mas tem a capacidade de encontrar boas soluções em um espaço de pesquisa com muitos mínimos locais.

Cada peixe, em localização n-dimensional, representa uma solução viável para o problema. Seu sucesso é medido pelo seu peso, uma conta cumulativa do sucesso da busca por cada peixe na escola. Os peixes não apenas armazenam informações sobre seu peso, mas também se posicionam no espaço de busca.

O FSS consiste em operadores de movimentação e alimentação. No movimento individual, cada peixe se desloca aleatoriamente em direção a uma posição em sua vizinhança à procura de regiões promissoras. Este componente é calculado usando a Equação 3.14.

$$\vec{n}_i(t) = \vec{x}_{i(t-1)} + step_{ind} \times rand[-1, 1] \tag{3.14}$$

onde $\vec{n}_i(t)$ representa o novo vetor posição do peixe na dimensão i, $\vec{x}_{i(t)}$ a posição atual, $step_{ind}$ sendo o passo individual e $rand[-1, 1]$ um número aleatório gerado por uma distribuição uniforme no intervalo $[-1; 1]$.

Após o movimento individual é necessário calcular a diferença entre o *fitness* dos indivíduos, de acordo com 3.15:

$$\Delta f_i(t) = f[\vec{n}_i(t)] - f[\vec{x}_i(t-1)] \tag{3.15}$$

onde $f[\vec{x}_i(t)]$ é o *fitness* do indivíduo atual e $f[\vec{n}_i(t)]$ é o fitness do próximo indivíduo.

Depois de se mudar para novas posições, todos os peixes têm seus pesos atualizados de acordo com a Equação 3.16. A atualização de peso é determinada pelo sucesso do movimento individual, que é calculado através da adequação das posições atual e nova.

$$W_{i(t)} = W_{i(t-1)} + \frac{\Delta f_{i(t)}}{max[\Delta f_{i(t)}]} \tag{3.16}$$

onde $W_i(t)$ é o peso atual do indivíduo.

Depois de alimentar todos os peixes, ocorre o movimento coletivo-instintivo. Todos os peixes se movem em direção a um vetor de influência, como mostrado nas Equações 3.17 e 3.18. Os peixes que melhoraram sua aptidão na iteração atual e geram esse vetor.

$$\vec{m}_{i(t)} = \frac{\sum N_i = \vec{x}_{i(t)} \times W_{i(t)}}{\sum N_i = W_{i(t)}} \tag{3.17}$$

onde $\Delta x_i(t)$ é o deslocamento do peixe gerado pelo movimento individual.

$$\vec{x}_{i(t)} = \vec{x}_{i(t-1)} + \vec{m}_{i(t)} \tag{3.18}$$

No final da iteração atual, o cardume se contrai ou se expande de acordo com o operador do movimento volitivo-coletivo. A contração da escola resulta em uma busca de exploração, enquanto sua expansão faz com que a escola explore a área de busca, evitando os mínimos locais. Assim, o operador volitivo calcula o baricentro da escola como mostrado na Equação 3.19 e atualiza o seu movimento, de acordo com a Equação 3.20. Este último operador fornece ao FSS a capacidade de auto-ajustar a granularidade da pesquisa ao longo do processo de otimização.

$$\vec{B}_i = \frac{\sum \vec{x}_i(t) W_i(t)}{\sum W_i(t)} \tag{3.19}$$

$$\vec{x}_i(t) = \vec{x}_i(t) \pm step_{vol} \frac{\vec{x}_i(t) - \vec{B}_i}{dist(\vec{x}_i(t), \vec{B}_i)} \tag{3.20}$$

onde $dist[]$ é uma função que retorna a distância euclidiana entre a posição do peixe e o baricentro, e $step_{vol}$ é um passo para controlar o deslocamento do movimento.

3.6 MOFSS

A *Multi-Objective Fish School Search* (MOFSS) é uma generalização FSS que lida com problemas com múltiplas restrições objetivas. Ele usa um Arquivo Externo (EA) para armazenar as melhores soluções não dominadas encontradas durante o processo de busca. As soluções no EA são usadas para guiar os movimentos dos peixes no espaço de busca. O método *Crowding Distance* trunca o EA para lidar com seu limite de tamanho. O MOFSS difere do FSS nos operadores de seleção de recursos que são adaptados para resolver problemas multi-objetivos.

No movimento individual, o peixe sempre se move em direção a novas posições, mesmo que sejam piores que as anteriores ou as posições sejam indiferentes, considerando o critério de dominância. Soluções no EA levam os movimentos individuais. Para cada peixe, um guia é selecionado usando o operador de seleção de líder. A seleção é executada por um torneio que seleciona dois peixes e executa uma seleção aleatória para retornar um guia.

A nova posição do pexei é calculada através da Equação 3.21:

$$\vec{n}_i(t) = \vec{x}_i(t) + \Delta\vec{x}_i(t) \tag{3.21}$$

Em que $\vec{x}_i(t)$ é a posição atual do peixe i, $\vec{n}_i(t)$ é a nova posição e $\Delta\vec{x}_i(t)$ é o deslocamento, calculado através da Equação 3.22:

$$\Delta\vec{x}_i(t) = S_{ind}(t) \times U[0,1] \times \frac{\vec{x}_g(t) - \vec{x}_i(t)}{DIST[\vec{x}_g(t), \vec{x}_i(t)]} \tag{3.22}$$

Onde $S_{ind}(t)$ é o passo individual, $U[0,1]$ é uma distribuição uniforme aleatória entre 0 e 1 e $DIST[\vec{x}_g(t), \vec{x}_i(t)]$ é uma função que calcula a distância euclidiana.

O operador de alimentação é quem define o conhecimento do cardume, atualizando o peso de cada peixe conforme seu movimento individual e, consequentemente, atualizando o peso total do cardume. Diferentemente da versão mono-objetiva do FSS (*Fish School Search*), nesta abordagem há varios objetivos conflitantes entre si e por isso é preciso utilizar um conceito de dominância para determinar qual e a melhor solução [25]. Para tanto, é necessário possuir dois elementos, a posição atual do peixe $x_i(t)$ e a nova posição do peixe $n_i(t)$, determinada pelo operador de de movimento individual [25].

Para determinar se o peixe ira sofrer um aumento ou um diminuição de peso, foi introduzido a verificação do valor de *Crowding Distance* da solução do arquivo externo que está mais proxima da posição atual do peixe e da solução do arquivo externo que esta mais proxima da nova posição deste [25].

Um fator de grau de domínio $D_i(t)$ também foi introduzido ao cálculo do peso do peixe. Este grau de domínio é determinado através da análise da quantidade de soluções que um peixe domina e da quantidade de soluções que dominam este mesmo peixe. Com isso, cria-se uma

variável que prioriza o aumento do peso dos peixes que estão mais próximos do arquivo externo e penaliza os que se encontram mais distantes [25].

O fator de dominância $D_i(t)$ é cálculo de acordo com a Equação 3.23:

$$D_i(t) = 1 - \frac{R_i}{MaxR} \tag{3.23}$$

em que $R(i)$ representa a aptidão (ou *fitness*) bruta do peixe e $MaxR$ é o máximo valor de aptidão bruta encontrado na iteração. A aptidão bruta $R(i)$ é calculado através da equação Equação 3.24,

$$R_i = \sum_{j \in N, j \prec i} S(j) \tag{3.24}$$

onde $S(j)$ é um valor de forçaa e representa o número de soluções que peixe j domina, calculado conforme a Equação 3.25:

$$S(i) = |j | j \in N \wedge i \prec j| \tag{3.25}$$

Depois do cálculo do grau de domínio, cada peixe tem seu valor de peso atualizado conforme a Equação 3.26:

$$w_i(t+1) = w_i + \Delta w_i(t) \times D_i(t) \tag{3.26}$$

E o peso total do cardume na iteração atual é calculado através da Equação 3.27:

$$w(t+1) = \sum_{i=1}^{N} w_i(t+1) \tag{3.27}$$

O movimento volitivo multi-objetivo leva o conjunto de soluções não dominadas da EA (Arquivo Externo) como pontos de referência para contratar ou expandir o cardume, enquanto o FSS usa o baricentro escolar para determinar esses pontos. Cada peixe da escola seleciona um líder, usando o operador de seleção de líderes, e se move em direção a ele. Assim, os peixes tendem a se mover para as soluções não dominadas [25].

O operador de movimento coletivo instintivo é responsável por fazer o cardume sofrer um deslocamento, influenciado pelo desempenho geral obtido por cada peixe em seu operador de movimento individual. Dois modos distintos são utilizados para determinar o vetor de influência coletiva instintiva. No primeiro modo, este vetor é calculado (ver Equação 3.28) igualmente para todo o cardume, ou seja, todos os peixes possuem o mesmo valor de infuência:

$$\vec{I}(t) = \sum_{i=1}^{N} \frac{\Delta w_i(t) \times \Delta \vec{x}_i(t)}{MAX[1, (\Delta w)]} \tag{3.28}$$

Já na segunda opção, cada peixe tem um vetor de influência determinado individualmente (ver Equação 3.29), como base no seu ganho de peso na iteração atual:

$$\vec{I}(t) = S_{vol}(t) \times U[0,1] \times \Delta w_i(t) \frac{\vec{x}_i(t) - \vec{x}_g(t)}{DIST[\vec{x}_i(t), \vec{x}_g(t)]} \tag{3.29}$$

Sendo as posições dos peixes atualizados através da Equação 3.30 em ambas abordagens.

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{I}(t) \tag{3.30}$$

Por fim, o operador de movimento coletivo volitivo determina se o cardume deve se movimentar para realizar uma busca em amplitude ou uma busca em profundidade. Para tanto é calculado um vetor de influência volitiva (ver Equação 3.31), que é adicionado a cada peixe do cardume, fazendo que eles se desloquem para uma nova posição [25].

$$V_i(t) = S_{vol}(t) \times SINAL(\Delta w) \times U[0,1] \tag{3.31}$$

onde $SINAL(\Delta w)$ é uma função que determina o sinal dos valores informados, retornando 1 caso o valor seja positivo ou -1 caso o valor seja negativo.

Como consequência, as posições dos peixes atualizados através da Equação 3.32.

$$\vec{x}_i(t+1) = \vec{x}_i(t) + V_i(t) \times \frac{\vec{x}_i(t) - \vec{x}_g(t)}{DIST[\vec{x}_i(t), \vec{x}_g(t)]} \tag{3.32}$$

Finalmente, o operador de turbulência ocorre para evitar que o enxame fique preso nos mínimos locais. O operador cria novas posições vizinhas que são avaliadas em cada iteração. Se as novas posições forem soluções não dominadas, elas serão incluídas no EA.

Capítulo 4

Trabalhos Relacionados

Na literatura existem diversos algoritmos utilizados para a resolução de problemas MTSP (*Multiple Travelling Salesmen Problem*) e KP (*Knapsack Problem*). Todos são variações de algoritmos genéticos ou de inteligência de enxames. Apesar de não haver, até o presente momento, uma abordagem para resolução do problema de distribuição de medicamentos, há um conjunto de problemas computacionais que são transversais a este. Tais problemas são estudados visando melhorar o atendimento comercial das empresas de distribuição sob um ou mais objetivos. Logo a seguir, são descritas brevemente as referências aos trabalhos transversais aos problemas MTSP e as metodologias empregadas.

Dentre os trabalhos pesquisados relacionados ao MTSP, Somhom et al. (1991) [34] cria uma Rede Neural Baseada em Competição (*Competition-Based Neural Network*) para minimizar a maior rota individual das soluções dentro de um MTSP com único depósito e rotas fechadas. Utilizam instâncias modelo da TSPLIB (*Travelling Salesman Problem Library*) [35] para comparar seu algoritmo com um algoritmo de Rede Elástica (*Elastic Net*) e com a heurística 2-opt generalizada. Implementa os 3 algoritmos para realizar os experimentos.

Tang et al. (2000) [11] busca melhorar a programação (*scheduling*) da produção por *hot rolling* (moldagem por meio de rolos da espessura de uma chapa de metal em altas temperaturas) em uma indústria chinesa de ferro e aço de grande porte. Os autores modelam o problema real como como um MTSP com depósitos iniciais e finais distintos. A minimização do custo total das soluções é realizada pelo Algoritmo Genético Modificado (MGA). A avaliação do seu algoritmo é efeito comparando os resultados com os do algoritmo exato de Volgenant e Jonker [11]. É utilizado instâncias aleatórias, considerando o custo das rotas e os tempos computacionais. Também é feito uma comparação do seu método com o utilizado na empresa a partir de dados reais.

Carter e Ragsdale (2006) [36] aplica um algoritmo genético com um novo cromossomo e operadores relacionados ao MTSP. O cromossomo é divido em duas partes, a primeira contendo uma permutação das cidades 1 a n; a segunda de tamanho m representa o número de cidades designada para cada caixeiro. Comparam os resultados obtidos com seu novo cromossomo com

outros dois cromossomos já existentes em instâncias aleatórias com 51, 100 e 150 cidades. Os autores fixam o tempo dos experimentos e analisam os custos das soluções.

Wang et al. (2007) [15] busca o agrupamento e roteamento de nós em redes de sensores multimídia sem fio visando o consumo eficiente de energia através da aprendizagem. Aplicam a otimização por colônia de formigas utilizando as regras do Ant System. O número máximo de cidades que que caixeiro pode visitar está limitado a um intervalo pré-definido, como em Junjie e Dingwei (2006) [37]. Comparam seu algoritmo ACO com um algoritmo de Simulated Annealing e com um algoritmo genético. Os aplicam a uma instância de 200 sensores e 3 grupos e analisam o consumo de energia quantizado na instância.

Vallivaara (2008) [4] trata de um tipo específico de MSTP para planejamento de rotas com múltiplos robôs em um ambiente hospitalar. Tem por objetivo a minimização da maior rota individual das soluções e minimização do custo total das soluções. Aplica otimização por colônia de formigas com regras do *Ant Colony System*. O abordagem constitui-se de N times com m formigas que geram, cada uma, uma única solução. Cada time de formigas corresponde a um caixeiro. Aplica a lista de nós candidatos. A formiga com menor rota parcial se move, após verificado se não há um movimento melhor. Aplica a 2-opt para todas as soluções geradas e a 3-opt na melhor solução do ciclo. Compara seus resultados com os apresentados em Somhom et al. (1991) e Zhu e Yang (2003) [38], obtidos sobre instâncias da TSPLIB criado por Reinelt (1995).

Liu (2009) tem como problema a distribuição de cigarros em uma empresa chinesa de grande porte. Buscam a minimização do custo total e minimização da maior rota individual das soluções MTSP com único depósito e rotas fechadas. Aplicam a otimização por colônia de formigas com Regra de Transição de Estado (RET) do Ant Colony System e Regra de Atualização de Feromônio (RAF) do Max-Min Ant System para o mecanismo de reinicialização de feromônio. Aplicam 4 heurísticas de busca local a todas as soluções criadas. Tem como experimento dados reais com 2153 pontos de entrega. Diminuem de 8 para 7 o número de veículos necessários. Fazem outro experimento comparando com os resultados de Vallivaara (2008), Somhom et al. (1991) e Zhu e Yang (2003).

Yousefikhoshbakht e Sedighpour (2012) [39] buscam Minimizar do custo total das soluções MTSP básico. Utilizam a Otimização por Colônia de Formigas com regras do Elitist Ant System num algoritmo para melhoramento das rotas individuais de soluções MTSP criadas por um algoritmo construtivo denominado Sweep Algorithm. A busca local 3-opt também é aplicada como heurística de melhoramento. Comparam seus resultados com os de Junjie e Dingwei (2006) e Tang et al. (2000) para os custos totais das soluções e os tempos computacionais.

Ji e Huang (2012) [40] utilizam um algoritmo híbrido de duas etapas com agrupamento através de k-Means e *Artificial Fish-Swarm Algorithm* (AFSA) para uma variação do MTSP com tempo limite (VRP). A primeira etapa utiliza o k-Means para dividir as regiões atendidas em diferentes blocos com cada cliente sendo visitado uma única vez e os carros tem tempo limite.

Na segunda etapa o AFSA busca a solução ótima. Realizam vários experimentos com instâncias aleatórias comparando o seu algoritmo com o ACO e *Hybrid Genetic Algorithm*.

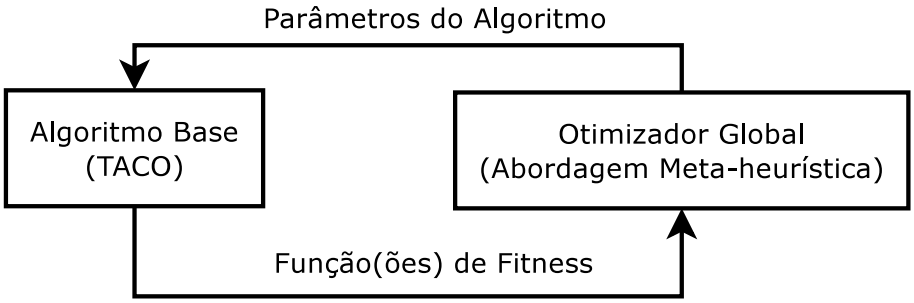
Barbosa e Kashiwabara (2015) [13] tem como problema a distribuição de ordens de serviço nas concessionárias de energia elétrica. Buscam a minimização do custo total e minimização da maior rota individual das soluções num MTSP básico. Aplicam duas versões modificadas do *Team Colony Optimization* (TACO) proposto por Vallivaara (2008), com regras do ACS (Single Team Ant Colony System - STACS) e com regras do Rank-Based Ant System (*Single Team Rank-Based Ant System* - STRBS). Fazem experimentos comparando STACS, STRBAS e TACO sob as instâncias da TSPLIB. Por fim, comparam STACS e STRBAS com as instâncias reais e analisam o tempo computacional. Utilizam teste de Wilcoxon pareado para análise estatística dos resultados.

Capítulo 5

Uma Proposta de Otimização dos Parâmetros do MTSP com Otimizadores Globais

O modelo proposto consiste em usar um algoritmo de base para calcular as melhores soluções para uma instância do MTSP, enquanto um algoritmo otimizador global procura os melhores conjuntos de parâmetros para o algoritmo base. Algoritmo otimizador global tanto pode ser mono objetivo como pode trabalhar com multiplos objetivos. Neste trabalho, o *Team Ant Colony Optimization* (TACO) é utilizado como algoritmo de base. Além disso, a Otimização de Enxame de Partículas (PSO), a Busca em Cardume de Peixes (FSS) e sua versão multi-objetivo (MOFSS) participam da otimização dos valores dos parâmetros TACO.

Figura 6 – Modelo de otimização associado a um otimizador global (OG)



Fonte: o próprio autor.

Conforme mostrado na Figura 6, o Otimizador Global (OG) começa a gerar um conjunto de valores para os parâmetros do TACO. Neste caso, os parâmetros otimizados do TACO são α, β, ξ e ρ. Então, a TACO constrói um conjunto de melhores soluções baseadas nos parâmetros otimizados. Em seguida, o TACO retorna ao OG o conjunto das melhores soluções, que são avaliadas para o algoritmo meta-heurístico. Conforme as iterações acontecem, o OG mantém um registro do melhor conjunto de parâmetros, que foram encontrados até o momento. A execução

termina quando o OG atinge o limite de iterações.

5.1 O Problema: distribuição de medicamentos

Uma CAF (Central de Abastecimento Farmacêutico) usualmente executa os pedidos de forma manual, tanto a separação da carga quanto o percurso estabelecido, independente do tempo gasto pelos entregadores (ver Figura 7). Devido à esta grande variação de tempo necessário no atendimento de pedidos, é difícil estabelecer uma capacidade para cada entregador, no intuito de mitigar os custos individuais dos percursos estabelecidos.

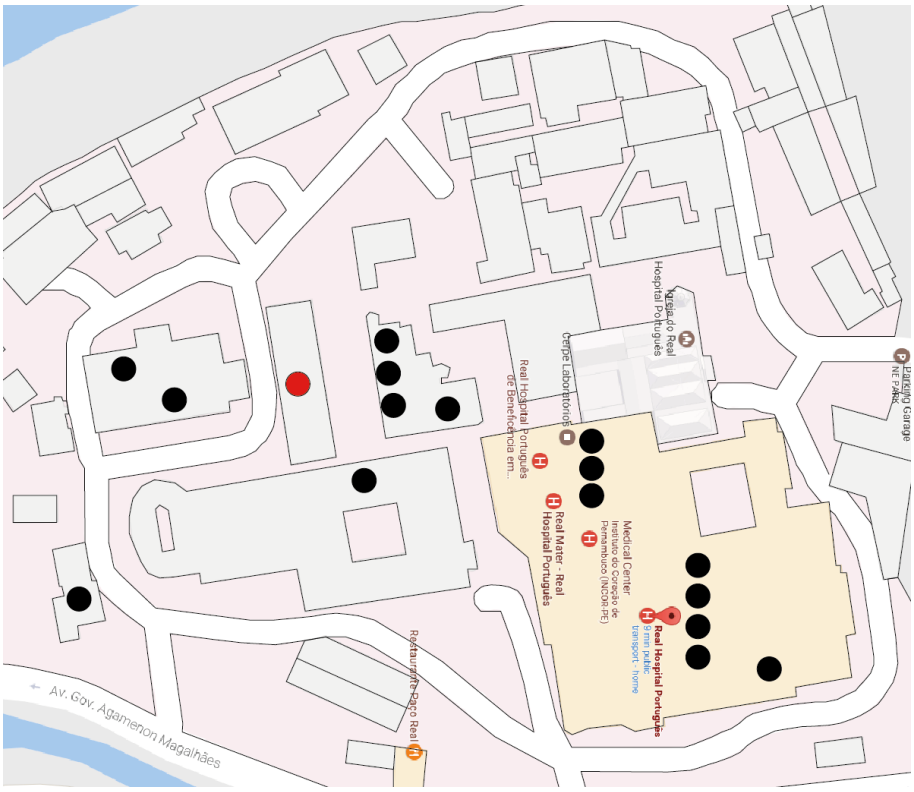
Figura 7 – Fluxograma do processo de pedido, separação e entrega da CAF.

Fonte: o próprio autor

A metodologia deste trabalho consiste na criação de instâncias de MTSP a partir do ambiente real hospitalar e na aplicação de algoritmos baseados no ACO para a construção otimizada de soluções na distribuição de medicamento entre os entregadores. Para este trabalho, foi desenvolvido uma instância MTSP tendo como base um hospital de grande porte, contendo farmácias satélites (FS) e uma CAF. Estas FSs estão distribuídas em diferentes blocos dentro do hospital e em diferente pavimentos. Na Figura 8 estão destacadas em preto as 16 farmácias em uma simulação para entregas durante o período diurno de um dia típico de trabalho. O ponto em cinza claro está destacado a CAF, local onde os entregadores iniciam e concluem os seus percursos.

Já na Figura 9 mostra o grafo com os vértices representando cada farmácia satélite incluindo o centro de distribuição representado pelo vértice V_1, com os respectivos custos em suas arestas, cada mensageiro sai do vértice V_1, realiza a rota determinada pelo algoritmo e retorna para o centro de distribuição, V_1. Neste grafo há 17 nós (vértices) sendo cada nó n é interligado aos outros $17 - n$ nós. Além disso, para esta abordagem, o problema é considerado como estático, isto é, todos os pedidos são feitos até as 8h da manhã pelas outras farmácias não considerando pedidos feitos depois do tempo limite.

Figura 8 – Representação gráfica das farmácias e o CAF num grande hospital



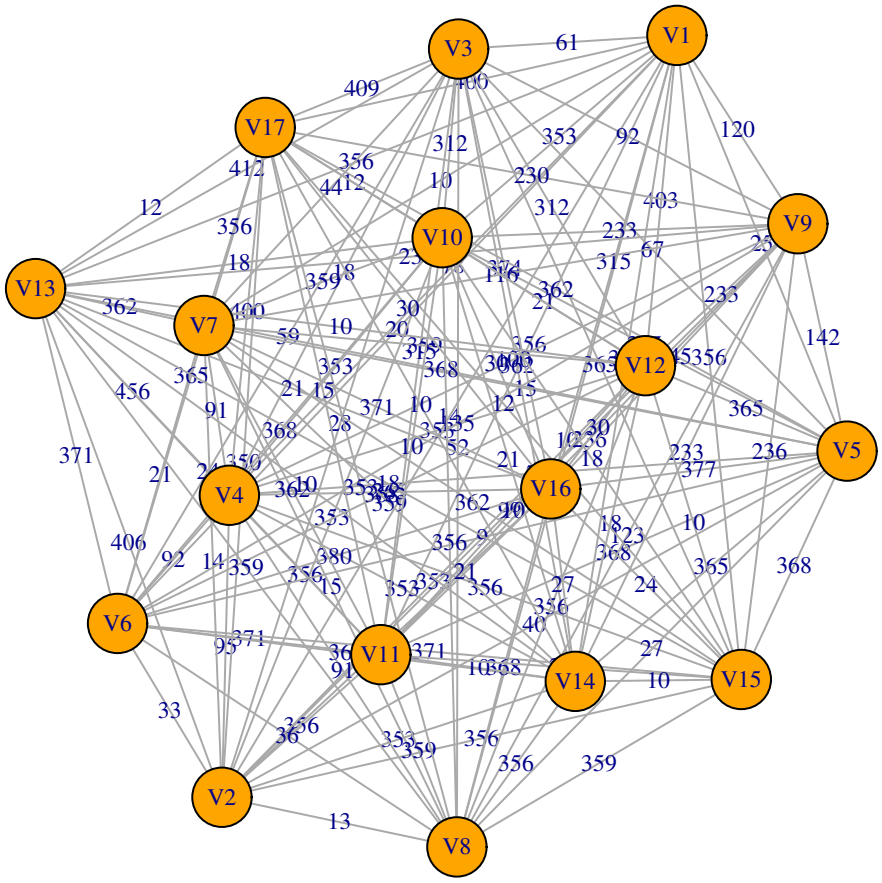
Fonte: o próprio autor

Uma matriz, chamada matriz de custo, contém os dados de cada farmácia, como número de registro (id), latitude, longitude e sua distância. A matriz ajuda a calcular o custo da rota. Da mesma forma, outra matriz, chamada de matriz de dados, contém as ordens de entrega para cada nó no modelo construído. Cada pedido contém registros do remetente e da farmácia, as distâncias iniciais e finais, quando o remetente deixou o CAF e a duração do pedido. A matriz de dados ajuda a simular um dia de pedido no depósito. Além disso, há quatro entregadores para entregar os pedidos às farmácias.

5.2 Parâmetros a serem otimizados

Como afirmado anteriormente, um otimizador global, baseado em inteligência de enxame, também é usado para melhorar os resultados do TACO, otimizando seus parâmetros. FSS, MOFSS e PSO são considerados como um otimizador global dos parâmetros TACO. Os valores padrão para o conjunto de parâmetros utilizados são: número de entregadores $M = 4$, número de equipes $N = 10$, probabilidade inicial $q_0 = 0,5$, relevância de feromônio $\alpha = 0,5$, relevância de visibilidade $\beta = 1,0$, persistência de feromônio para atualização local $\xi = 0,1$, persistência de feromônio para atualização global $\rho = 0,1$. Esses valores padrão foram obtidos por um teste de parâmetro. O critério de parada é de 1000 iterações para cada execução independente. Os otimizadores globais otimizam o subconjunto de parâmetros $P = \{\alpha, \beta, \xi, \rho\}$ em um intervalo

Figura 9 – Modelo representado em forma de grafo com as suas arestas contendo as distâncias



Fonte: o próprio autor.

de $0 \leq P \leq 2$.

5.3 Metodologia Experimental

Os experimentos apresentados nesta seção têm como objetivo mostrar a eficácia da metodologia desenvolvida aplicada ao problema real. No primeiro cenário, os algoritmos são configurados para minimizar o custo total das soluções, sem considerar a distribuição da carga de trabalho entre as equipes, como na descrição geral do MTSP. No segundo cenário, os algoritmos minimizam o custo da maior rota individual das soluções, visando a construção de soluções formadas por rotas com custos iguais entre os entregadores, como no MTSP com balanceamento de carga de trabalho.

5.3.1 Ambiente de Simulação

Todos os experimentos foram executados em um MacBook Pro (13 polegadas, final de 2011) com CPU Intel Core i5 de 2,4 GHz, 16 GB de RAM (1333 MHz DDR3) e sistema

operacional MacOS High Sierra (versão 10.13.4). Usamos o banco de dados apresentado na Seção 5.1. Em seguida, realizamos 30 execuções independentes dos algoritmos para cada experimento. O algoritmo TACO foi codificado em Java com base no algoritmo proposto por Valivaara (2007) e está disponível em <https://github.com/renanalencar/taco-mofss.git>. O FSS é baseado na versão de Verçosa, Bastos-Filho e Monteiro (2017) [41]. O PSO de objetivo único foi retirado da estrutura do jMetal (DURILLO; NEBRO, 2011) [42]. O MOFSS utilizado é baseado na versão de Bastos-Filho e Guimarães (2015) [25] utilizando o jMetal como framework para visualizações.

O TACO desenvolvido por Valivaara foi utilizado por ter apresentado bons resultados na resolução de problemas do tipo MTSP durante experimentos prévios no desenvolvimento deste trabalho. O algoritmo foi codificado em Java para facilitar a integração com outras ferramentas já disponíveis e algoritmos utilizados como OGs. Já o FSS desenvolvido foi escolhido Verçosa, Bastos-Filho e Monteiro por ter como base a versão FSS Vanilla (disponível em <http://www.fbln.pro.br/fss/versions.htm>) e apresentar estruturação das classes compatível com os problemas da Conference on Commerce and Enterprise Computing (CEC).

O jMetal é um repositório criado por Durillo et al. em 2010, e que utiliza linguagem de programação Java, e fornece a implementação de vários algoritmos existentes na literatura, tanto mono quanto multi-objetivo. A utilização desta ferramenta contribuiu, de forma significativa, no auxilio do desenvolvimento deste trabalho, visto que algumas necessidades já estavam implementadas, poupando tempo de desenvolvimento. Além disso, o PSO utilizado neste trabalho provem deste framework cuja a implementação corresponde ao PSO padrão.

Por fim, O MOFSS baseado na versão de Bastos-Filho e Guimarães foi escolhido por ter sido desenvolvido utilizando o framework JMetal o que facilita na integração com os outros algoritmos desenvolvidos em Java. Além disso, esta implementação gera gráficos dos operadores utilizados pelo MOFSS e o gráfico contendo as soluções não-dominadas.

5.3.2 Métrica de Avaliação

A comparação de desempenho de um ou vários métodos de otimização mono e multi-objetivo requer parâmetros que possam ser relacionados e comparáveis. Neste trabalho, a métrica utilizada foi o desvio padrão (DP) calculado em cima da média de resultados de 30 simulações independentes. O DP é um parâmetro muito usado em estatística que indica o grau de variação de um conjunto de elementos, i.e., é uma medida que expressa o grau de dispersão de um conjunto de dados. Logo, o desvio padrão indica o quanto um conjunto de dados é uniforme. Quanto mais próximo de 0 for o desvio padrão, mais homogêneo são os dados.

5.3.3 Configuração dos experimentos

O TACO participa do experimento como um algoritmo de base. Antes de iniciar os experimentos do modelo proposto, um teste de base deve ser realizado para comprovar a eficácia

e robustez do algoritmo escolhido. Os valores padrão para ambos os cenários são indicados na Seção 5.2. O FSS como um GO é executado com um critério de parada de 1000 iterações por execução e possui 30 execuções independentes. Os valores dos parâmetros do FSS foram retirados do trabalho de Verçosa, Bastos-Filho e Monteiro (2017). Da mesma forma, o PSO tem os mesmos valores de critério de parada e execuções independentes. Os valores do parâmetro PSO permaneceram os mesmos que no framework jMetal. Por fim, foram mantidos os valores padrões apresentados no trabalho de Bastos-Filho e Guimarães para a abordagem MOFSS.

Capítulo 6

Resultados

O primeiro conjunto de experimentos foi realizado para minimizar o Custo Total de Rotas (CTR), ou seja, a soma da rota de cada equipe. Ao comparar as soluções, que possui o menor custo total, é considerada a melhor. Este cenário pode ser aplicado à vida real quando pretendemos reduzir a quantidade total de rotas de entrega em vez de priorizar o equilíbrio de trabalho.

O segundo conjunto de experimentos visou minimizar a Rota Mais Longa (RML) das soluções mantendo os mesmos valores dos parâmetros no primeiro cenário. Esse caso é adequado para situações reais quando priorizamos o equilíbrio entre as rotas individuais (*workbalance*) em vez da soma total das rotas.

Os resultados mostrados aqui são referentes ao experimentos somente envolvendo o MTSP (*Multiple Travelling Salesmen Problem*). O tratamento subsequente para soluções envolvendo tanto o MTSP quanto KP (*Knapsack Problem*) não foi tratado nestes experimento devido a ajustes que entrarão na continuação deste trabalho.

6.1 Resultados sem Otimizadores Globais

Foi utilizado a abordagem TACO com regras do ACS para otimização da instância vista na Seção 5 para quatro entregadores. Foram realizados dois experimentos: minimização do custo total das soluções e minimização da maior rota individual das soluções.

A Tabela 1 mostra a distância média de execução do algoritmo (em quilômetros) para a instância. Este valor foi obtido a partir da média da distância encontrada (tanto para Custo Total de Rotas - CTR quanto para Rota Mais Longa - RML) nos 1000 ciclos TACO e nas 30 execuções do algoritmo, por um dia útil. Os resultados provam a robustez dao algoritmo é minimizar os resultados para CTR e RML.

O TACO apresentou resultados satisfatórios para as duas variações do MTSP: minimizar o custo total da solução e minimizar o custo da maior rota de solução individual (Tabela 1). Os

Tabela 1 – Resultados para o TACO sem Otimizadores Globais

Média de 30 simulações com 1000 iterações							
Minimizando o Custo Total das Rotas (CTR)				Minimizando a Rota Mais Longa (RML)			
CTR (km)	D.P.	RML (km)	D.P.	CTR (km)	P.D.	RML (km)	D.P.
1,8	0,25	0,721	0,003	1,223	0,050	0,756	0,012

pequenos valores dos desvios padrão nas duas tabelas confirmam a robustez do algoritmo ao gerar soluções com custos próximos da média das 30 execuções. Embora a RL esteja sendo minimizada, o TCR varia ao longo das iterações, conforme observado no Figura 10. A mesma situação ocorre quando a TACO minimiza o TCR (ver Figura 11). Tendo isso em mente, ainda há espaço para melhorias nos resultados de ambos os objetivos.

Figura 10 – Curva de convergência quando minimizado o CTR

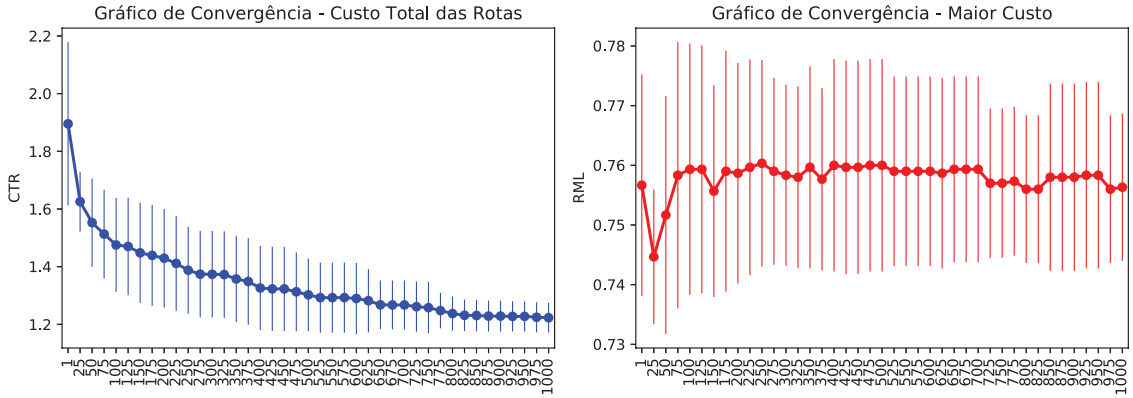
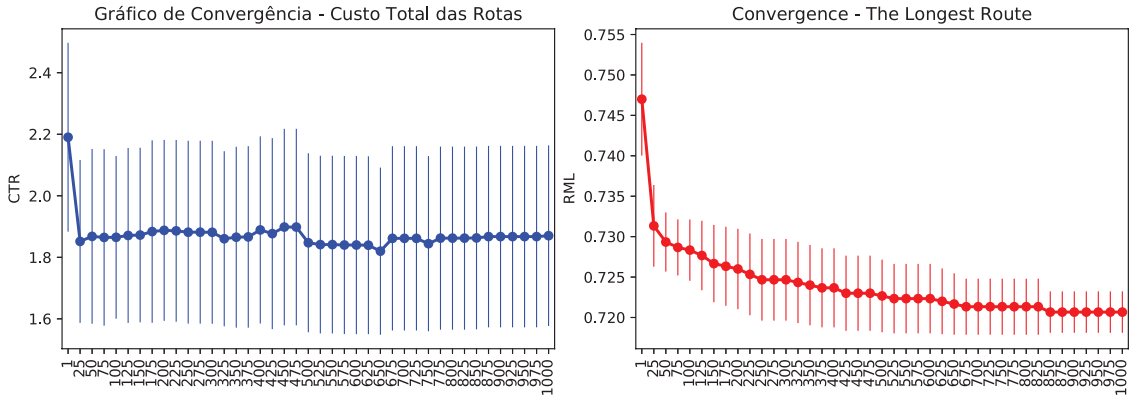


Figura 11 – Curva de convergência quando minimizado RML



6.2 Resultados com Otimizadores Globais

Os experimentos realizados com o otimizador externo mostraram melhores resultados para ambos os cenários, mostrados nas Tabelas 2 e 3. Comparando as Tabelas 1 e 2, o FSS tem uma melhora melhor ao minimizar a rota mais longa em comparação ao algoritmo base. Esse

resultado é devido à capacidade do FSS de explorar o espaço de pesquisa. Além disso, o FSS apresenta um desvio padrão menor, o que corrobora a sua robustez.

Tabela 2 – Resultados para TACO com FSS sendo o Otimizador Global

Média de 30 simulações com 1000 iterações							
Minimizando o Custo Total das Rotas (CTR)				Minimizando a Rota Mais Longa (RML)			
CTR (km)	D.P.	RML (km)	D.P.	CTR (km)	D.P.	RML (km)	D.P.
1,0797	0,002	0,747	0	1,944	0,335	0,719	0,001

Figura 12 – Curva de convergência FSS-TACO quando minimizado o CTR

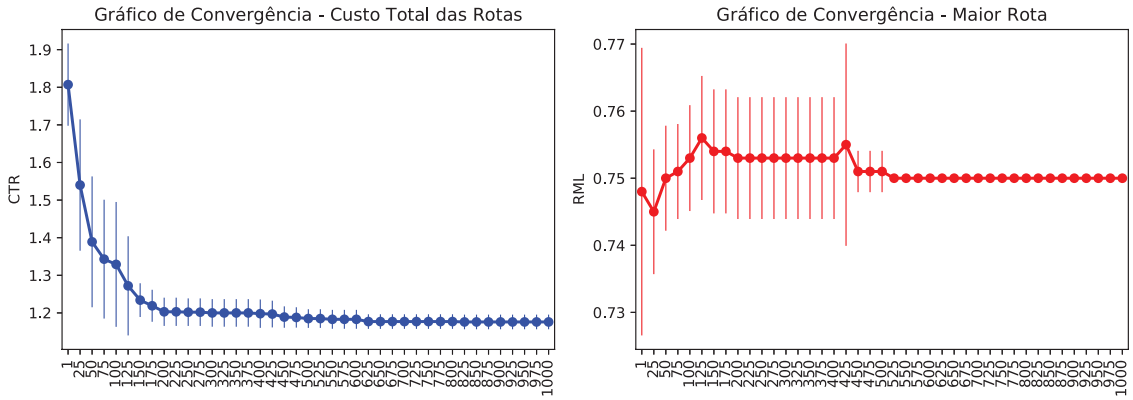
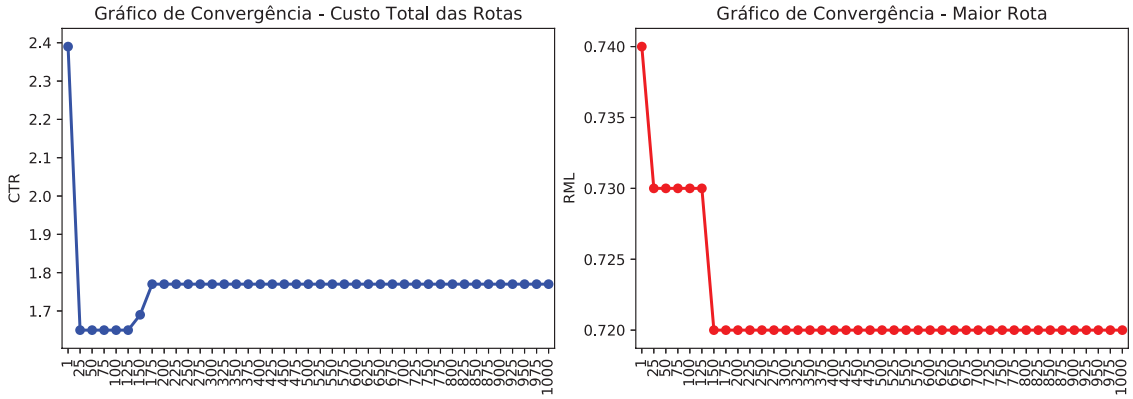


Figura 13 – Curva de convergência FSS-TACO quando minimizado a RML



PSO também teve melhores resultados em comparação com abordagem TACO sem otimizadores externos (ver Tabela 3). Seu desvio padrão também mostra a robustez e eficácia ao otimizar TCR e LR.

Tabela 3 – Resultados para TACO com PSO sendo o Otimizador Global

Média de 30 simulações com 1000 iterações							
Minimizando o Custo Total das Rotas (CTR)				Minimizando a Rota Mais Longa (RML)			
CTR (km)	D.P.	RML (km)	D.P.	CTR (km)	D.P.	RML (km)	D.P.
1,115	0,032	0,750	0,003	2,048	0,333	0,720	0,005

Figura 14 – Curva de convergência PSO-TACO quando minimizado o CTR

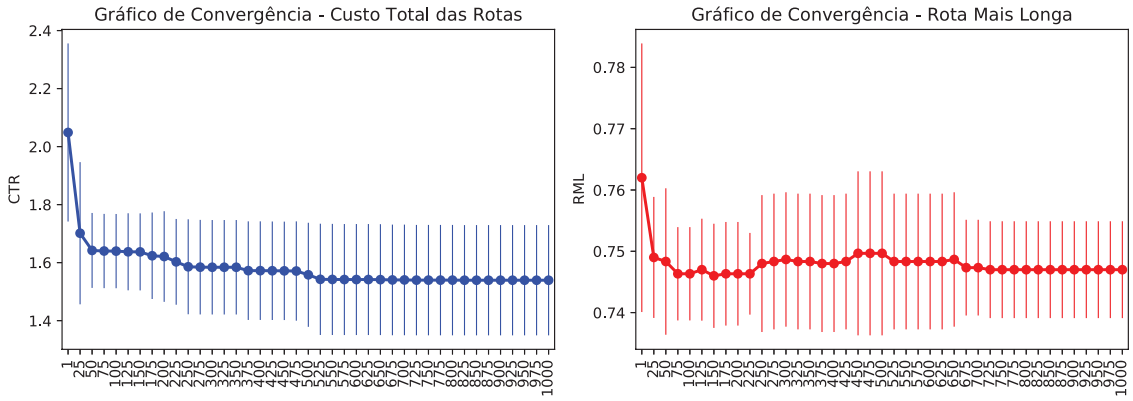
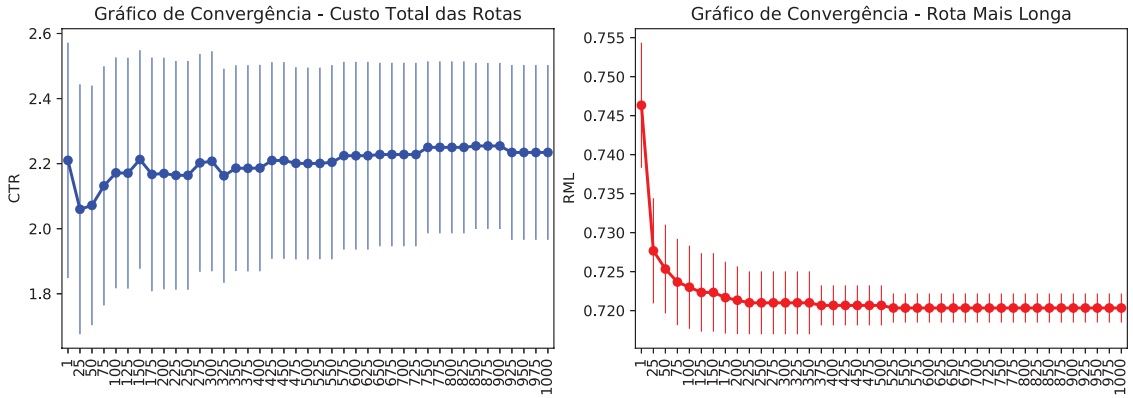


Figura 15 – Curva de convergência PSO-TACO quando minimizado a RML



Os experimentos realizados com o otimizador externo mostraram melhores resultados para ambos os cenários, mostrados nas Tabelas 2 e 3. Conforme apresentado nas Tabelas 1 e 2, o FSS tem uma melhora melhor de minimizar a rota mais longa em comparação ao algoritmo base. Esse resultado é devido à capacidade do FSS de explorar o espaço de pesquisa. Além disso, o FSS apresenta um desvio padrão menor, o que corrobora sua robustez.

6.3 Comparativo entre os experimentos

Todos os resultados anteriores são compilados na Tabela 4 para um melhor entendimento. Comparando as três abordagens, o FSS obteve melhores resultados ao minimizar tanto o CTR quanto a RML. Também teve o menor resultado com um desvio padrão próximo de zero. Como visto na Figura 16, em ambos os cenários, o FSS convergiu mais cedo do que o PSO e o algoritmo de base. Os resultados do PSO foram melhores que os resultados do algoritmo de base e convergiram mais cedo do que o TACO como esperado.

A Figura 17 mostra o desvio padrão para ambos os CTR e RML. Ao minimizar o CTR, podemos notar que os resultados do TACO variam mais do que o FSS e PSO. Por outro lado, PSO tem a menor variação de seus resultados. Ao minimizar a RML, vale ressaltar que para

Figura 16 – Comparação entre as convergências de TACO, FSS-TACO e PSO-TACO

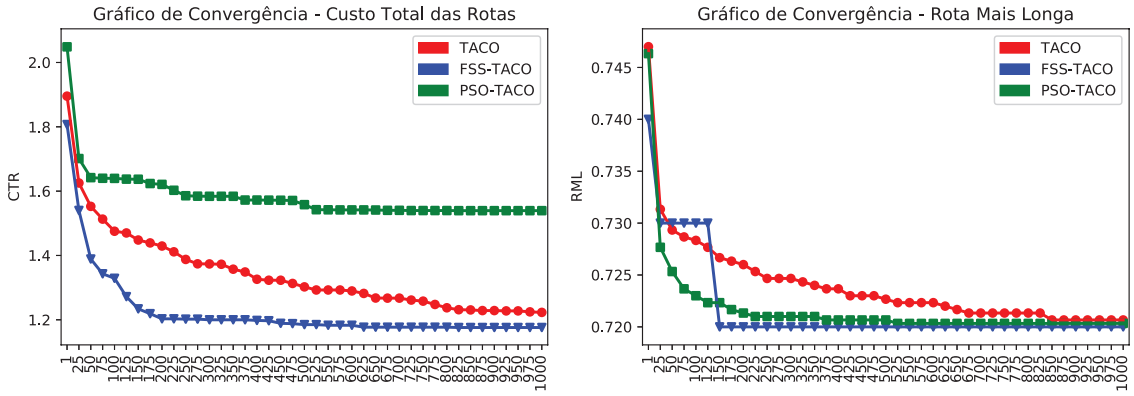
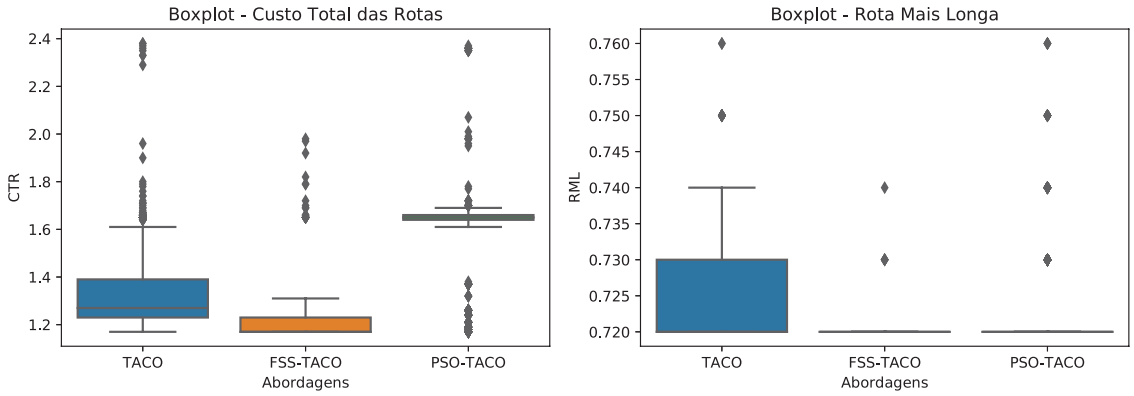


Tabela 4 – Comparação entre resultados com 30 simulações independentes

Abordagem	Minimizando o CTR				Minimizando a RML			
	CTR (km)	D.P.	RML (km)	D.P.	CTR (km)	D.P.	RML (km)	D.P.
TACO	1,8	0,25	0,721	0,003	1,223	0,050	0,756	0,012
FSS-TACO	**1.0797**	**0.002**	0,747	0	1,944	0,335	**0.719**	**0.001**
PSO-TACO	1,115	0,032	0,750	0,003	2,048	0,333	0,720	0,005

todas as abordagens (TACO, FSS e PSO) existe uma pequena ou nenhuma variação entre seus resultados.

Figura 17 – Boxplot com os desvios padrões da minimização do CTR e da RML



6.4 Tempos de Execução dos Algoritmos

A Tabela 5 apresenta o tempo médio de execução do algoritmo para cada instância. Este valor foi obtido a partir da média do tempo (em milissegundos) gasto para execução dos 1000 ciclos de cada abordagem nas 30 execuções independente do algoritmo.

Quanto aos tempos de execução do algoritmo, mostrados na Tabela 5, verifica-se que para a instância experimentada, foram gastos em media aproximadamente 0,6 segundo para execução dos 1000 ciclos do algoritmo, que representam um tempo aceitável para o problema

Tabela 5 – Tempo médio de execução de 1.000 ciclos de TACO, FSS-TACO e PSO-TACO

Abordagem	Minimizando o CTR		Minimizando a RML	
	Média em ms	Média em horas	Média em ms	Média em horas
TACO	$6,185 \times 10^{2}$	$0,017 \times 10^{-2}$	$6,097 \times 10^{2}$	$0,0169 \times 10^{-2}$
FSS-TACO	$1,695 \times 10^{7}$	4,707	$1,570 \times 10^{7}$	4,362
PSO-TACO	$5,21856220 \times 10^{6}$	1,450	$5,161 \times 10^{6}$	1,434

em estudo. Entretanto, ao comparar a abordagem utilizando os OGs PSO e FSS o tempo médio de execução passa 1 hora e 4 horas, respectivamente. Em ambos os cenários de otimização (minimizando CTR ou RML), a abordagem com OG sendo PSO é mais rápida do que o OG sendo o FSS. Ao levar em consideração o tempo que os pedidos são feitos ao CAF (pedidos aceitos até às 8h da manhã), o PSO leva vantagem pois o tempo de processamento é menor, levando a um menor tempo na distribuição dos medicamentos.

6.5 Análise dos Resultados o MOFSS-TACO

Os resultados para o MOFSS sendo o OG do algoritmo base (ver Tabela 6) se mostram bastante interessantes. Ao minimizar simultaneamente CTR e RML, o MOFSS consegue resultados menores que o TACO sem otimizador externo, tendo em vista os valores otimizados do TACO de forma mono-objetiva como visto na Tabela 1. Entretanto, os número obtidos são maiores quando comparados as abordagens PSO-TACO e FSS-TACO. Todavia, deve ser levado em consideração que os objetivos CTR e RML estão sendo minimizados de forma simultânea o que garante um tempo de execução menor dos que os outros dois.

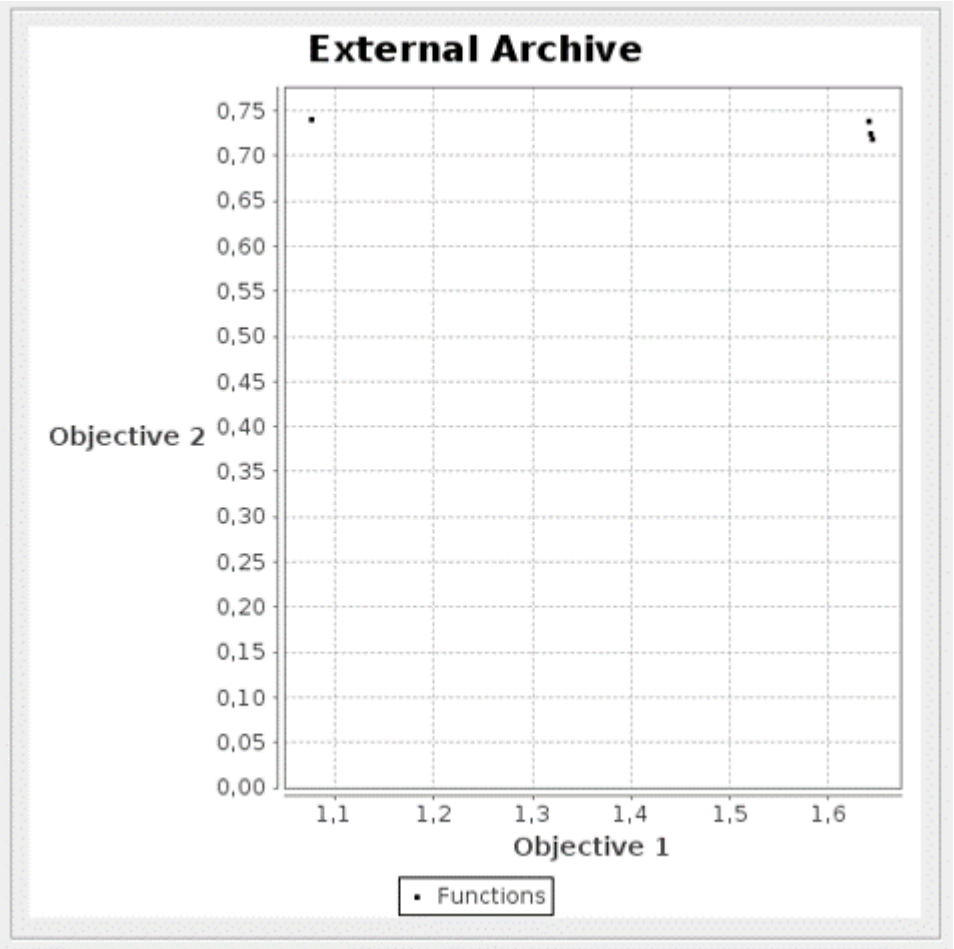
Tabela 6 – Resultados para TACO com MOFSS sendo o Otimizador Global

Média de 30 simulações com 1000 iterações				
Minimizando CTR e RML				
CTR	D.P.	RML	D.P.	Tempo Médio de Execução (horas)
1,398	0,279	0,732	0,011	20,1

Ao fim de cada execução, a abordagem MOFSS-TACO gera um conjunto de soluções ótimas não-dominadas. Como o problema não tem grande complexidade, ao final de cada execução independente gera um conjunto com quatro soluções não dominadas como visto na Figura 18.

Um problema encontrado durante os experimentos com o MOFSS-TACO é o seu tempo de execução visto na Tabela 6. Enquanto as abordagens FSS-TACO e PSO-TACO tiveram tempo médio em torno de 4,3 e 1,4 horas (ver Tabela 5), respectivamente, a abordagem com OG multi-objetivo teve um tempo médio de execução proximo de 20,1 horas. O que torna o processo dispondioso se comparado a rotina de pedidos do CAF, discriminado na Seção 6.5.

Figura 18 – Gráfico do arquivo externo contendo as soluções não dominadas para o MOFSS-TACO.



Fonte: O próprio autor

6.6 Discussão dos Resultados

O resultado do desvio padrão (D.P.) se mostra satisfatório confirmando a robustez do algoritmo para os experimentos contemplando somete o TACO. Comparado com a minimização dos dois objetivos, a minimização da RML foi inferior na minimização do CTR. Essa otimização proporciona um número maior de entregas pelos entregadores visto que o custo da rota individual é balanceado de modo a não sobrecarregar ou ocasionar ociosidade em um dos entregadores. Com os entregadores realizando mais entregas em menos tempo, essa técnica é mais favorável quando o custo total do deslocamento não é o objetivo, mas sim a quantidade de entregas por cada entregador.

Vale ressaltar que, na minimização do CTR, a carga não é distribuída de maneira uniforme para os entregadores podendo um ou mais entregadores ficar com sobrecarga ou ocioso dependendo da configuração. Já na minimização da RML, focado em minimizar a maior rota individual, distribuindo a carga de maneira uniforme para todos os entregadores.

Ao comparar os resultados, sem e com OGs e otimização mono-objetivo, é possível perceber o bom desempenho da abordagem FSS-TACO em face das abordagens TACO e PSO-TACO. Entretanto, o seu uso se torna inviável, dependendo da situação, por requerer um tempo de execução superior aos das outras abordagens presentes na comparação. Para uma abordagem multi-objetiva, o MOFSS sendo o OG do algoritmo base, mostra resultados parciais interessantes visto que obteve resultados melhores para CTR e RML se comparado aos resultados do algoritmo base. Como a instância construída para este trabalho não possui grande complexidade, a Curva de Pareto, com as soluções ótimas, se restringe a quatro soluções não-dominadas.

Capítulo 7

Conclusão

7.1 Conclusão

O problema de Múltiplos Caixeiros Viajantes (MTSP) vem sendo estudado nas últimas duas décadas apresentando várias abordagens para a otimização de suas soluções. Entretanto, ainda é escasso estudos relacionados ao MTSP dentro da área hospitalar. Este trabalho mostra que a distribuição de medicamentos em um hospital de grande porte ou em grande rede de farmácias pode ser realizada de forma eficiente com o auxílio de uma ferramenta computacional.

Este trabalho propôs o uso de uma metodologia para resolução de problemas de MTSP, que visa a distribuição de medicamentos, otimizando a programação de percursos para a entrega. A partir de um cenário hospitalar, são construídas instâncias de MTSP. Estas instâncias são otimizadas pela abordagem TACO ()*Team Ant Colony Optimization*), algoritmo proposto por Vallivaara (2008) [4], para distribuição de medicamentos a material hospitalar bem como os percursos a serem realizados pelos entregadores.

As melhores soluções geradas pela TACO são utilizadas para definir a distribuição das ordens entre os entregadores e os percursos que devem ser realizados. Ao fim do processo de encontro das soluções, um otimizador global é utilizado, baseado em algoritmos de base populacional como os algoritmos de enxames.

O modelo proposto descrito foi eficiente na distribuição das ordens de serviço entre os entregadores e na criação de rotas otimizadas para a realização dos serviços. No entanto, algumas questões precisam ser analisadas, como a otimização multi-objetivo do custo total das rotas e a rota mais longa e a otimização das mochilas dos entregadores.

Para a criação de soluções otimizadas para MTSP, um algoritmo baseado na meta-heurística ACO foi implementado e associado a um otimizador global. O FSS e PSO foram responsáveis por otimizar os parâmetros TACO para melhorar os resultados.

Como mostrado na Seção 6, o FSS alcançou os melhores resultados para ambos os cenários, com os menores resultados de minimização para o Custo Total de Rotas (CTR) e a

Rota Mais Longa (RML) com desvios padrão em torno de zero. Nesses dois casos, o FSS como otimizador global convergiu mais cedo do que as outras duas abordagens.

Outra abordagem possível para o problema do MTSP é otimizar os dois objetivos simultaneamente, CTR e RML. Para atingir este objetivo, foi utilizando um algoritmo de otimização multi-objetivo chamado Multi-Objective Fish School Search (MOFSS). Os resultados parciais mostram um bom desempenho em relação ao algoritmo base (TACO). Ao minimizar CTR RML concomitantemente, a abordagem mostra melhores resultados se comparado ao TACO sem utilização de otimizadores externos. Para o problema do mundo real, é imprescindível este tipo de abordagem para que se possa fazer a mitigação de perdas e custos no processo de distribuição bem como otimizar o capital humano dispendido nos processos de logística.

7.2 Trabalhos Futuros

Como trabalho futuro, pretende-se verificar a utilização de algumas variações de valores para os operadores dos otimizadores globais a fim de encontrar melhores configurações. Também deve ser verifcado a distribuição dos medicamentos reduzindo variando o quadro de entregadores (aqui considerados como caixeiros da instância MTSP).

Outra meta-heurística que tem obtido resultados consistentes para problemas de otimização combinatória são os Algoritmos Genéticos (AG) [34], que serão utilizados como otimizadores globais para comparação com os resultados obtidos pelos experimentos deste trabalho em novos experimentos.

Com os resultados parciais deste trabalho, foi escrito um artigo para a *18th International Conference on Hybrid Intelligent Systems* (HIS 2018 - <http://www.mirlabs.org/his18/>), que será realizado em Porto, Portugal, no período de 13 a 15 de dezembro de 2018. O artigo foi aceito para ser apresentado no evento.

Para análise e efeito comparativo entre a proposta e os algoritmos de estado-da-arte, pretende-se realizar testes estásticos com os dados coletados durante as simulações da instância MTSP criada para a verificação de desempenho. Experimentos com a proposta e abordagens multi-objetivas como SMPSO (NEBRO et al., 2009) [43], MOEA/D (ZHANG; LI, 2007) [44] e SPEA2 (ZITZLER; LAUMANNS; THIELE, 2001) [45] também serão realizados para verificar a robustez e eficiência. Como consequência, haverá realização de um quadro comparativo entre as diferentes instâncias do problema e a aplicação das abordagens citadas neste trabalho.

Por último, outro ponto diz respeito ao caráter dinâmico do ambiente em estudo. Enquanto a metodologia proposta mostra-se viável para cenários estáticos, no qual todas as ordens e as posições dos entregadores são conhecidas antes da otimização, no ambiente real serviços surgem durante o dia de trabalho dos entregadores e podem ter maior prioridade que outros, como as ordens emergenciais. Para tratamento desta demanda, a metodologia desenvolvida será adaptada

para gerar novas soluções otimizadas a cada mudança no ambiente real, como o surgimento de novas ordens.

Referências

[1] AUGUSTINE, J. E. Offline and online variants of the traveling salesman problem. 2002.

[2] BEKTAS, T. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, Elsevier, v. 34, n. 3, p. 209–219, 2006.

[3] LIN, F.-T. Solving the knapsack problem with imprecise weight coefficients using genetic algorithms. *European Journal of Operational Research*, Elsevier, v. 185, n. 1, p. 133–145, 2008.

[4] VALLIVAARA, I. A team ant colony optimization algorithm for the multiple travelling salesmen problem with minmax objective. In: ACTA PRESS. *Proceedings of the 27th IASTED International Conference on Modelling, Identification and Control*. [S.l.], 2008. p. 387–392.

[5] MARTELLO, S.; TOTH, P. *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990. ISBN 0-471-92420-2.

[6] RIBEIRO, S. *LogÌstica hospitalar: desafio contante*. 2005. Disponível em: <http://www.noticiashospitalares.com.br/mar2005/htms/apoio.htm>.

[7] SOUZA, A. A. de; PEREIRA, A. C. C.; XAVIER, A. G.; XAVIER, D. O.; MENDES, E. S. Logística hospitalar: um estudo de caso diagnóstico das dificuldades na gestão logística do setor de engenharia clínica. *REA-Revista Eletrônica de Administração*, v. 12, n. 1, p. 1–14, 2013.

[8] HOLLAND, J. H. Genetic algorithms. *Scientific american*, JSTOR, v. 267, n. 1, p. 66–73, 1992.

[9] DORIGO, M.; OCA, M. A. M. de; ENGELBRECHT, A. Particle swarm optimization. *Scholarpedia*, v. 3, n. 11, p. 1486, 2008.

[10] KARABOGA, D. *An idea based on honey bee swarm for numerical optimization*. [S.l.], 2005.

[11] TANG, L.; LIU, J.; RONG, A.; YANG, Z. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. *European Journal of Operational Research*, Elsevier, v. 124, n. 2, p. 267–282, 2000.

[12] LIU, W.; LI, S.; ZHAO, F.; ZHENG, A. An ant colony optimization algorithm for the multiple traveling salesmen problem. In: IEEE. *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*. [S.l.], 2009. p. 1533–1537.

[13] BARBOSA, D. F.; JR, C. N. S.; KASHIWABARA, A. Y. Aplicaçao da otimizaçao por colônia de formigas ao problema de múltiplos caixeiros viajantes no atendimento de ordens de serviço nas empresas de distribuiçao de energia elétrica. *Anais do XI Simpósio Brasileiro de Sistemas de Informaç ao*, p. 23–30, 2015.

[14] BARBOSA, D. F.; JR, C. N. S.; KASHIWABARA, A. Y. Aplicação do rank-based ant system ao problema de múltiplos caixeiros viajantes no atendimento de ordens de serviço nas empresas de distribuição de energia elétrica. *iSys-Revista Brasileira de Sistemas de Informação*, v. 8, n. 4, p. 05–43, 2016.

[15] WANG, X.; WANG, S.; BI, D.; DING, L. Hierarchical wireless multimedia sensor networks for collaborative hybrid semi-supervised classifier learning. *Sensors*, Molecular Diversity Preservation International, v. 7, n. 11, p. 2693–2722, 2007.

[16] PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. [S.l.]: Courier Corporation, 1982.

[17] BRANKE, J.; BRANKE, J.; DEB, K.; MIETTINEN, K.; SLOWIŃSKI, R. *Multiobjective optimization: Interactive and evolutionary approaches*. [S.l.]: Springer Science & Business Media, 2008. v. 5252.

[18] COLLETTE, Y.; SIARRY, P. *Multiobjective optimization: principles and case studies*. [S.l.]: Springer Science & Business Media, 2013.

[19] MURAKAMI, L. T. *Solução de Problemas de Otimização utilizando Arquitetura Híbrida*. Tese (Doutorado) — Universidade de São Paulo, 2008.

[20] FILHO, C. J. B.; NETO, F. B. de L.; LINS, A. J.; NASCIMENTO, A. I.; LIMA, M. P. A novel search algorithm based on fish school behavior. In: IEEE. *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*. [S.l.], 2008. p. 2646–2651.

[21] FILHO, C. J. B.; NETO, F. B. de L.; LINS, A. J.; NASCIMENTO, A. I.; LIMA, M. P. Fish school search. In: *Nature-inspired algorithms for optimisation*. [S.l.]: Springer, 2009. p. 261–277.

[22] LACERDA, M. G. P. de. Uma nova heurística de segregação de cardumes para otimização multi-solução de problemas multimodais. 2012.

[23] MALAQUIAS, N. G. L. et al. Uso dos algoritmos genéticos para a otimização de rotas de distribuição. Universidade Federal de Uberlândia, 2006.

[24] SILVA, F. A. V. *Um Algoritmo genético para o problema de roteamento de veículos com janela de tempo aplicado na distribuição de serviços de telecomunicação*. Tese (Doutorado), 2016.

[25] BASTOS-FILHO, C. J.; GUIMARÃES, A. C. Multi-objective fish school search. *International Journal of Swarm Intelligence Research (IJSIR)*, IGI Global, v. 6, n. 1, p. 23–40, 2015.

[26] CHAN, F. T.; TIWARI, M. K. Preface: Swarm intelligence, focus on ant and particle swarm optimization. In: *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*. [S.l.]: InTech, 2007.

[27] EBERHART, R. C.; SHI, Y.; KENNEDY, J. *Swarm intelligence*. [S.l.]: Elsevier, 2001.

[28] ENGELBRECHT, A. P. *Fundamentals of computational swarm intelligence*. [S.l.]: John Wiley & Sons, 2006.

[29] DORIGO, M. The any system optimization by a colony of cooperating agents. *IEEE Trans. System, Man & Cybernetics-Part B*, v. 26, n. 1, p. 1–13, 1996.

[30] MULATI, M. H.; CONSTANTINO, A. A.; SILVA, A. F. da. Otimização por colônia de formigas. *LOPES, HS; RODRIGUES, LCDA; STEINER, MTA Meta-Heuristicas em Pesquisa Operacional. Ominipax*, p. 53–68, 2013.

[31] KENNEDY, J.; EBERHART, R. C. The particle swarm: social adaptation in information-processing systems. In: MCGRAW-HILL LTD., UK. *New ideas in optimization*. [S.l.], 1999. p. 379–388.

[32] SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: IEEE. *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. [S.l.], 1998. p. 69–73.

[33] CLERC, M.; KENNEDY, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, IEEE, v. 6, n. 1, p. 58–73, 2002.

[34] SOMHOM, S.; MODARES, A.; ENKAWA, T. Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers & Operations Research*, Elsevier, v. 26, n. 4, p. 395–407, 1999.

[35] REINELT, G. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, INFORMS, v. 3, n. 4, p. 376–384, 1991.

[36] CARTER, A. E.; RAGSDALE, C. T. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research*, Elsevier, v. 175, n. 1, p. 246–257, 2006.

[37] JUNJIE, P.; DINGWEI, W. An ant colony optimization algorithm for multiple travelling salesman problem. In: IEEE. *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*. [S.l.], 2006. v. 1, p. 210–213.

[38] ZHU, A.; YANG, S. X. An improved self-organizing map approach to traveling salesman problem. In: IEEE. *Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on*. [S.l.], 2003. v. 1, p. 674–679.

[39] YOUSEFIKHOSHBAKHT, M.; SEDIGHPOUR, M. A combination of sweep algorithm and elite ant colony optimization for solving the multiple traveling salesman problem. *Proceedings of the Romanian academy A*, v. 13, n. 4, p. 295–302, 2012.

[40] JI, D.-g.; HUANG, D.-m. A research based on k-means clustering and artificial fish-swarm algorithm for the vehicle routing optimization. In: IEEE. *Natural Computation (ICNC), 2012 Eighth International Conference on*. [S.l.], 2012. p. 1141–1145.

[41] VERÇOSA, L.; BASTOS-FILHO, C.; MONTEIRO, R. Combining a novel feeding operator and recent advances to improve the fish school search. In: IEEE. *Computational Intelligence (LA-CCI), 2017 IEEE Latin American Conference on*. [S.l.], 2017. p. 1–6.

[42] DURILLO, J. J.; NEBRO, A. J. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, Elsevier, v. 42, n. 10, p. 760–771, 2011.

[43] NEBRO, A. J.; DURILLO, J. J.; GARCIA-NIETO, J.; COELLO, C. C.; LUNA, F.; ALBA, E. Smpso: A new pso-based metaheuristic for multi-objective optimization. In: IEEE. *Computational intelligence in miulti-criteria decision-making, 2009. mcdm'09. ieee symposium on*. [S.l.], 2009. p. 66–73.

[44] ZHANG, Q.; LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, IEEE, v. 11, n. 6, p. 712–731, 2007.

[45] ZITZLER, E.; LAUMANNS, M.; THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm. 2001.