Adaptação do *Rank-Based Ant System* ao problema de múltiplos caixeiros viajantes no atendimento de ordens de serviço em uma empresa de distribuição de energia elétrica

Denilson F. Barbosa[1], **Carlos N. Silla Jr.**[1], **André Y. Kashiwabara**[1]

[1]Programa de Pós-Graduação em Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Cornélio Procópio – PR – Brazil

Abstract. *This paper presents a modification of the Rank-Based Ant System to deal with the optimization of the execution of the commercial services of an electric power distribution company. This activity represents a significant portion of the operational costs of these companies. In this paper this problem is cast as a multiple travelling salesmen problem, where each team is a travelling salesman and each commercial service is a place to be visited. The methodology presented innovates to distribute orders and route the teams simultaneously; and to explore new ways to apply Ant Colony Optimization in Multiple Traveling Salesmen Problem. In order to evaluate our approach, we have used a set of real instances of the problem. The analysis of our results shows that the costs related to the longest route of each work day is reduced, on average by 44,43%. The source-code of our system and the dataset are freely available at https://github.com/denilsonfag/STRBAS.*

Resumo. *Neste artigo é proposta uma adaptação do Rank-Based Ant System para lidar com a otimização do atendimento comercial em uma empresa de distribuição de energia elétrica, atividade que representa uma parcela significativa dos custos operacionais destas empresas. Neste trabalho esse problema é caracterizado como um problema de múltiplos caixeiros viajantes, onde cada equipe de atendimento é um caixeiro e cada ordem de serviço é uma posição a ser visitada. A metodologia apresentada é inovadora ao distribuir as ordens e rotear as equipes simultaneamente; e ao explorar novas possibilidades de aplicação da Otimização por Colônia de Formigas ao Problema de Múltiplos Caixeiros Viajantes. Nos experimentos utilizando instâncias reais os resultados mostram que, na média, houve uma redução dos custos da maior rota individual dos dias de trabalho de 44,43%. O protótipo desenvolvido e os dados reais utilizados encontram-se disponíveis em https://github.com/denilsonfag/STRBAS.*

1. Introdução

As empresas de distribuição de energia elétrica são responsáveis pela manutenção do sistema de distribuição e pela gestão comercial dos consumidores localizados em sua área de atendimento. Estas atividades exigem diariamente a execução de ordens de serviço por equipes móveis de eletricistas em diferentes pontos da área de atendimento. As ordens de serviço são denominadas emergenciais quando relacionadas à manutenção emergencial do sistema, e comerciais quando ligadas à gestão dos consumidores.

BARBOSA, D. F.; SILLA JR., C. N.; KASHIWABARA, A. Y.
Adaptação do Rank-Based Ant System ao problema de múltiplos caixeiros viajantes no atendimento de ordens de serviço em uma empresa de distribuição de energia elétrica
iSys – Revista Brasileira de Sistemas de Informação, Rio de Janeiro, vol. 8, No. 4, pp. 05-43, 2015

O atendimento comercial representa a maior parte das ordens de serviço executadas pelas empresas de distribuição de energia [Volpi et al. 2008]. A Agência Nacional de Energia Elétrica (ANEEL), que regulamenta as distribuidoras brasileiras, define prazos para execução das ordens de serviço comercias que, se não cumpridos, implicam em sanções ou multas às distribuidoras [Garcia et al. 2010b]. Devido a esses fatores, é economicamente rentável que as equipes de atendimento realizem rotas eficientes, de forma a diminuir as distâncias percorridas e executar o maior número possível de ordens durante seu dia de trabalho.

Normalmente, o gerenciamento do atendimento comercial é realizado por técnicos que distribuem manualmente as ordens entre as equipes no início do dia de trabalho. Em seguida, cabe a cada equipe definir também manualmente a sequência em que realizará seus atendimentos. Tanto a distribuição das ordens quanto as sequências de atendimento admitem um grande número de combinações, exigindo a utilização de uma ferramenta computacional para melhor aproveitamento das informações disponíveis, como as posições das ordens a executar já conhecidas.

A partir de informações da agência de atendimento de Cornélio Procópio, estado do Paraná, pertencente à subsidiária de Distribuição da Companhia Paranaense de Energia (COPEL), foi desenvolvida uma metodologia para realizar a distribuição das ordens e a definição de rotas eficientes para as equipes simultaneamente. A metodologia consiste na criação de instâncias do Problema de Múltiplos Caixeiros Viajantes (*Multiple Travelling Salesmen Problem*, MTSP) a partir das ordens comerciais a executar em um dia de trabalho, e na aplicação da Otimização por Colônia de Formigas (*Ant Colony Optimization*, ACO) para criação de soluções otimizadas para as instâncias.

O MTSP foi escolhido para tratamento do problema real porque as soluções geradas podem ser utilizadas para distribuir as ordens entre as equipes e também definir suas rotas. Nos trabalhos relacionados, as metodologias empregadas realizam estas tarefas isoladamente, conforme pode ser constatado a partir dos trabalhos apresentados na Seção 1.2.1. A ACO foi escolhida para realização das otimizações porque tem conseguido resultados consistentes para outros problemas de otimização e porque, conforme pode ser verificado na Seção 1.2.3, a sua aplicação ao MTSP representa um campo de estudo com várias hipóteses a serem exploradas.

Devido à complexidade do problema, algoritmos baseados em diferentes metodologias já foram aplicados ao MTSP. Além de métodos exatos, como a programação linear inteira, que passam a exigir tempo computacional proibitivo à medida que as instâncias crescem [Bektas 2006], diversos métodos baseados em heurísticas têm sido utilizados: redes neurais [Somhom et al. 1999, Zhu and Yang 2003], algoritmos genéticos [Tang et al. 2000, Carter and Ragsdale 2006, Brown et al. 2007, Yuan et al. 2013, Arya et al. 2014], teoria dos grafos [Wang et al. 2013], *Artificial Bee Colony* (Colônia de Abelhas Artificial) [Venkatesh and Singh 2015], além da própria ACO [Vallivaara 2008, Junjie and Dingwei 2006, Wang et al. 2007, Liu et al. 2009b, Ghafurian and Javadian 2011, Costa et al. 2012, Vasagam 2012, Yousefikhoshbakht et al. 2013, Zhou and Yao 2013]. Assim como nos métodos exatos, o tamanho das instâncias é a principal limitação da eficiência dos métodos heurísticos, o que estimula a exploração de novas metodologias para tratamento do MTSP.

Este artigo é uma versão estendida e convidada do trabalho apresentado em [Barbosa et al. 2015]. As principais contribuições desta versão estendida são: a definição de um algoritmo geral que adapta a ACO para o MTSP, baseado no trabalho de [Vallivaara 2008]; um novo algoritmo ACO adaptado para MTSP, o Sistema Baseado em Rank com Equipe Única de Formigas (*Single Team Rank-Based Ant System*, STRBAS), que obteve melhores resultados que o algoritmo apresentado no artigo anterior, o Sistema de Colônia com Equipe Única de Formigas (*Single Team Ant Colony System*, STACS), quando aplicados às mesmas instâncias reais; uma seleção inicial do parâmetro β para os dois algoritmos ACO adaptados para MTSP, sendo o melhor valor inferido utilizado nos demais experimentos deste artigo; e a aplicação de custos de deslocamento previstos obtidos a partir de um sistema *web* de consulta cartográfica [MapQuest 2015] para preenchimento das matrizes de custos das instâncias, buscando uma melhor representação do ambiente real do problema.

Nos experimentos realizados a partir de dados reais, o STRBAS gerou soluções nas quais a distância percorrida pela equipe que mais se deslocou durante o dia de trabalho foi em média 44,43% melhor que a realizada nos dias de trabalho reais, quando utilizando custos representando o tempo de deslocamento com veículo entre os pontos das instâncias, confirmando a eficácia da nossa metodologia aplicada ao problema real. O restante deste trabalho está organizando da seguinte forma: na Seção 1.1 é realizada uma descrição detalhada do problema real e na Seção 1.2 são apresentados os trabalhos relacionados ao problema e à nossa metodologia. Na Seção 2, o funcionamento da nossa metodologia é detalhado e na Seção 3 são apresentados os experimentos realizados a partir dos dados reais. A Seção 4 aponta as contribuições deste trabalho e indica o rumo de sua continuação.

1.1. O problema real

As empresas de distribuição de energia elétrica são responsáveis pela manutenção do sistema elétrico de distribuição e pela gestão dos consumidores localizados na sua área de concessão. Na COPEL, a área de concessão é dividida entre as agências de atendimento, que dispõem de um conjunto de equipes para execução das ordens de serviço de sua subárea. Cada agência pode ser responsável por um grupo de municípios vizinhos, um único município, ou um grupo de bairros de uma grande cidade.

O ambiente analisado neste trabalho é o atendimento no município de Cornélio Procópio, estado do Paraná, que possui 637,95 km^2 de extensão territorial e 19.276 consumidores de energia elétrica, dos quais 3,76% estão localizados na área rural [IPARDES 2014]. As equipes de atendimento iniciam seus atendimentos a partir da garagem da agência e retornam a ela no final do dia trabalho. O horário de trabalho das equipes inicia-se às 08:00hs e termina às 18:00hs. Normalmente, as equipes disponíveis estão aptas a executar qualquer tipo de ordem de serviço, não havendo, portanto, restrições quanto à habilidade das equipes na distribuição das ordens.

As ordens de serviço são denominadas emergenciais quando relacionadas à manutenção do sistema de distribuição e necessitam de atendimento imediato; ou comercias, quando ligadas à gestão dos consumidores e podem ser programadas, pois existe um prazo máximo para atendimento, definido de acordo com o tipo da ordem comercial. No início do dia de trabalho existe um conjunto de ordens comerciais aguardando execução,

formado pelas ordens geradas automaticamente pelo sistema de gerenciamento de ordens durante a noite, como por exemplo ordens para suspensão de fornecimento devido à falta de pagamento; e pelas ordens que não puderam ser atendidas no dia de trabalho anterior devido ao término do horário de trabalho das equipes.

Na Figura 1 os pontos vermelhos representam as 31 ordens de serviço executadas no município de Cornélio Procópio em 3 de fevereiro de 2014, um dia de trabalho típico. O ponto azul representa a garagem da agência de atendimento, local onde as 3 equipes disponíveis iniciaram e concluíram suas rotas neste dia. Os pontos destacados na Figura 1 foram determinados a partir das coordenadas no sistema de projeção Universal Transverso de Mercator (UTM), que foram obtidas do histórico da agência de atendimento. O UTM é o sistema de coordenadas cartesianas bidimensional oficial brasileiro para mapeamentos [Machado Jr et al. 2004].

Figura 1. Os losangos vermelhos representam as ordens de serviço executadas no município de Cornélio Procópio em um dia de trabalho típico e o círculo azul corresponde à garagem da agência de atendimento. Fontes: mapa de *Google Maps* e coordenadas dos pontos de COPEL Distribuição.

A metodologia apresentada neste trabalho foca o atendimento comercial, otimizando a programação das rotas para as ordens de serviço já conhecidas. A partir de cenários como o da Figura 1, são construídas instâncias de MTSP que são otimizadas por algoritmos ACO, conforme explicado em detalhes na Seção 2. As melhores soluções geradas na otimização são utilizadas para definir a distribuição das ordens entre as equipes e as rotas que devem ser realizadas. O caráter dinâmico do problema, no qual as ordens de serviço surgem durante o dia de trabalho, será incluído posteriormente na nossa metodologia e apresentado em trabalhos futuros.

1.2. Referencial Teórico

Na literatura são encontradas diferentes abordagens que objetivam melhorar o atendimento comercial das empresas de distribuição sob um ou mais aspectos. As referências relacionados ao problema real e as metodologias empregadas estão citadas na Seção 1.2.1. Na Seção 1.2.2 é detalhado o MTSP e suas variações, e na Seção 1.2.3 são apresentadas as referências sobre a ACO e sobre os trabalhos que a aplicam ao MTSP.

1.2.1. Trabalhos relacionados ao problema real

[Steiner et al. 2006] objetivam otimizar o deslocamento das equipes aos locais de execução das ordens de serviço comerciais e emergenciais. Para isso, procuram determinar qual o menor trajeto a ser percorrido ao se deslocar entre duas rotas de leitura quaisquer das 48 que compõem uma das seis agências de atendimento do município de Curitiba, Paraná. Uma agência de atendimento é responsável por uma partição da área de concessão da empresa de distribuição, sendo constituída por uma determinada quantidade de equipes de atendimento comercial e emergencial e pela delimitação de uma região de atendimento. Uma rota de leitura são subdivisões da região de atendimento de uma agência que agrupam consumidores geograficamente próximos.

Em sua metodologia, [Steiner et al. 2006] consideram as posições centrais das rotas de leitura como os pontos que definem os vértices de um grafo assimétrico $G = (V, A)$. O conjunto de arestas A é definido por uma matriz preenchida pelos custos de se deslocar de uma rota a outra qualquer. A partir do grafo é utilizada uma implementação do algoritmo de Floyd, que é capaz de fornecer os custos mínimos e os caminhos de menor distância entre os pares de nós de um grafo. Os resultados da aplicação são duas matrizes que relacionam as 48 rotas de leitura entre si, uma apontando o menor custo entre elas, para ser utilizada pelo despachador no momento que um serviço deve ser designado. A outra matriz apresenta o trajeto de menor custo entre duas rotas de leitura quaisquer, e deve ser consultada pelas equipes antes de iniciarem seus deslocamentos durante a execução dos atendimentos.

[Costa et al. 2006] focam seu trabalho na distribuição entre as equipes dos serviços comerciais, cuja execução pode ser programada antecipadamente. Propõem uma solução que busca encontrar os melhores agrupamentos de rotas de leitura em setores de atendimento, a fim de homogeneizar a distribuição dos serviços entre as equipes disponíveis. Um setor de atendimento é uma subdivisão da região de atendimento da agência e é formado por um conjunto de rotas de leitura. Analisando os dados de um mês dos atendimentos da mesma agência que [Steiner et al. 2006], [Costa et al. 2006] verificam que as demandas de serviços dos setores de atendimento são significativamente diferentes, o que dificulta uma divisão homogênea dos serviços entre as equipes, já que cada equipe fica responsável por um ou mais setores. Sugerem a solução deste problema modelando-o como um Problema de p-Medianas Capacitado (PPMC), definindo os centros das rotas de leitura como os vértices do grafo, assim como [Steiner et al. 2006].

[Costa et al. 2006] criam as soluções através de duas metodologias. Na primeira, é criada uma aplicação para a geração de um modelo matemático a partir do PPMC e das características do problema real, na qual é definida uma capacidade temporal para as equipes. O modelo gerado alimenta uma ferramenta comercial de otimização que realiza

os agrupamentos. Na segunda metodologia, é implementado um algoritmo genético específico para o problema. Algoritmos genéticos são meta-heurísticas que usam técnicas baseadas na Evolução Natural dos seres vivos na criação das soluções. Heurísticas são regras e métodos incorporados aos algoritmos visando encontrar melhores soluções para problemas de otimização combinatória, enquanto que meta-heurística é uma heurística genérica que utiliza a combinação de escolhas aleatórias e conhecimento histórico de resultados anteriores na construção das soluções.

Em [Garcia et al. 2010a] e [Garcia et al. 2010b] é apresentada uma metodologia especifica para designação de serviços programáveis, assim como [Costa et al. 2006]. Objetivam o aumento da produtividade das equipes, modelando o problema combinando o PPMC com o Problema de Roteamento de Veículos (PRV). Modelam o problema em estudo como um PPMC definindo os locais dos serviços como os vértices do grafo e determinando uma capacidade temporal para os agrupamentos. Para tratar o problema, é proposto um algoritmo baseado na heurística construtiva de Ahmadi e Osman, que considera uma medida de densidade ao gerar os agrupamentos [Ahmadi and Osman 2005]. Heurísticas construtivas inicialmente criam uma solução e efetuam buscas locais para melhorar sua qualidade.

A partir dos agrupamentos criados, [Garcia et al. 2010b] consideram cada grupo individualmente como uma instância de um PRV e calculam o custo da rota. Caso o custo seja maior que a capacidade das equipes, a solução é descartada e são gerados novos agrupamentos. O PRV é um dos problemas mais estudados na área de otimização combinatória, que consiste no atendimento de um conjunto de consumidores por uma frota de veículos que partem de um ou mais depósitos. Na formulação básica do PRV, existe uma capacidade máxima que é determinada para os veículos e uma quantidade a ser entregue em cada ponto visitado. O objetivo pode ser o de encontrar o conjunto de rotas com menor deslocamento total ou encontrar o menor número possível de veículos que atenda à demanda. As soluções criadas devem atender à restrição de capacidade dos veículos [Christofides 1976].

[Goel and Meisel 2013] focam as atividades das equipes de manutenção, que necessitam de desligamentos de trechos da rede de distribuição para serem executadas. Os desligamentos são programados e aproveitados para execução de todos os reparos necessários em diversos pontos do trecho desenergizado. O trabalho de [Goel and Meisel 2013] tem como objetivo designar cada tarefa para um trabalhador e determinar um escalonamento que minimize os deslocamentos entre os serviços e o tempo de desligamento da rede para realização das tarefas. Utilizam a meta-heurística Busca em Larga Vizinhança (*Large Neighborhood Search*, LNS), combinada com técnicas de programação matemática. A meta-heurística LNS procura escapar de ótimos locais realizando movimentos em uma grande vizinhança durante a criação das soluções. A metodologia foi implementada em um aplicação que gera o modelo matemático, utilizado como entrada em uma ferramenta comercial de otimização que gera as soluções.

[Zografos et al. 1998] apresentam um estudo de caso para gerenciamento do atendimento emergencial de uma empresa de distribuição de energia elétrica de grande porte do sudeste dos Estados Unidos, considerando a totalidade de sua região de atendimento. Focam seu trabalho na redução da duração das interrupções de fornecimento de energia, através da otimização do aproveitamento das equipes e da utilização da grande quantidade

de informação disponível de maneira adequada. Estes objetivos são buscados através do desenvolvimento de um sistema de informação integrado composto por três módulos. O módulo de gerenciamento de dados consiste num SIG usado para representar e gerenciar os dados espaciais relacionados à área de atendimento e ao atendimento emergencial. O segundo módulo, de monitoramento de veículos e comunicação, obtém as posições correntes das equipes e viabiliza a comunicação entre elas e os outros módulos do sistema. O terceiro módulo, de modelagem, analisa as informações fornecidas pelos outros dois e gera informações otimizadas para despacho dos serviços e roteamento das equipes.

O ambiente estudado por [Gomes et al. 2008], assim como o de [Zografos et al. 1998], contempla apenas atendimentos emergenciais, consistindo no município de Fortaleza, estado do Ceará. O objetivo de sua metodologia é propor uma heurística para realizar a previsão do horário de atendimento dos serviços emergenciais e redefinir a ordem de execução dos atendimentos quando do surgimento de novos serviços. As reordenações devem atender aos seguintes objetivos, que estão citados na ordem que são aplicados: atender primeiramente as emergências de maior prioridade; minimizar o desvio entre a previsão inicial do tempo de atendimento e a previsão final; e minimizar a distância total percorrida por todas as equipes.

O trabalho de [Amorim 2010], assim como os dois anteriores, está focado no atendimento emergencial. Ele propõe uma metodologia para aumentar a eficiência do despacho das ordens emergenciais dividida em três etapas. A primeira etapa é responsável pela preparação dos dados que serão utilizados nas fases seguintes, que correspondem ao número de equipes disponíveis e às coordenadas geográficas (longitude e latitude) de todas as ordens a executar, como também a classificação dessas ordens como individuais (afetam apenas um consumidor) ou coletivas (geralmente causadas por defeitos em circuitos de baixa tensão, que alimentam em média 150 consumidores). A segunda fase é responsável pela distribuição dos serviços entre as equipes disponíveis e, na fase três, para cada grupo gerado da fase anterior, é definida a ordem que os serviços devem ser executados através um algoritmo genético, que busca minimizar os tempos de atendimento de cada equipe, como também o número médio de horas que os clientes ficaram sem fornecimento de energia.

O trabalho de [Verboski 2010] apresenta um algoritmo para otimização do despacho dos serviços comerciais e emergenciais às equipes modelando o problema como um Problema de Designação e utilizando o método de Munkres na busca de soluções em tempo real. Um problema de designação consiste em encontrar o menor custo total ao se designar a um conjunto de equipes E um conjunto de ordens de serviço O, com cada equipe recebendo apenas um serviço. Desta forma, c_{OE} corresponde ao custo de se atribuir um serviço a uma equipe, e consiste nos valores que preenchem a matriz sobre a qual é aplicada o algoritmo de Munkres. Seu valor é calculado através de uma função que considera três fatores: a distância entre a equipe e o serviço; o número de clientes atendidos pelo serviço; e o tempo que falta para que o prazo de execução do serviço se esgote.

[Garcia et al. 2012] abordam o problema do despacho das ordens de serviço como um PRV que requer um sistema de tempo real para vincular cada ordem a uma equipe. Segundo eles, a definição dos tipos das ordens de serviço e o fato de nem todas serem conhecidas ao mesmo tempo e antes das equipes iniciarem seus turnos de trabalho são

as principais características específicas do problema real. Em sua análise, os serviços comerciais são conhecidos, *a priori*, e as ordens emergenciais podem surgir a qualquer momento. Então, existe a necessidade de reprogramação das rotas de todas as equipes quando do surgimento de ordens emergenciais, que devem ser atendidas imediatamente. Para isso, propõem um modelo matemático com três funções objetivo: 1: minimizar o tempo total das rotas de todas as equipes; 2: minimizar o tempo que os clientes aguardam a execução das ordens emergenciais; e 3: minimizar o *makespan* (diferença entre os custos finais das rotas das equipes). Foram atribuídos pesos a cada um dos objetivos, o que permite que o modelo matemático seja configurado a considerá-los com diferentes intensidades ao ser resolvido.

[Nahuis 2013] apresenta uma metodologia para automação do despacho dinâmico das equipes de atendimento mediante um modelo matemático de Programação Linear Inteira Mista (PLIM). A metodologia inclui uma ferramenta computacional para hierarquizar o conjunto de ordens de serviço a executar e atribuí-las às equipes disponíveis. A cada nova ordem gerada, o conjunto de ordens a executar é reordenado e uma nova atribuição é realizada a todas as equipes.

1.2.2. O Problema de Múltiplos Caixeiros Viajantes

O TSP (*Travelling Salesman Problem*, Problema do Caixeiro Viajante) é um dos problemas de otimização combinatória mais pesquisados, que se destaca por ser de fácil declaração mas de difícil resolução, pertencendo à classe dos problemas NP-completos [Augustine 2002]. Em sua definição clássica, em uma instância do problema existem n cidades a serem visitadas por um caixeiro, que objetiva realizar a menor rota possível que passe por todas as cidades exatamente uma vez e retorne à cidade de partida. Várias aplicações do mundo real podem ser modeladas como instâncias de um TSP, como rotas de transporte escolar, sequenciamento de DNA, manufatura de circuitos de microchips e inspeções em plataformas de petróleo [Kanda 2014].

O MTSP (*Multiple Travelling Salesmen Problem*, Problema de Múltiplos Caixeiros Viajantes) é uma generalização do TSP na qual mais de um caixeiro participa da solução. Como explica [Bektas 2006], o MTSP é mais apropriado para aplicações do mundo real que o TSP e pode ser estendido para uma grande variedade de PRVs (Problema de Roteamento de Veículos) com a adição de restrições às soluções. Na definição geral do MTSP, ou MTSP básico, existem $m > 1$ caixeiros inicialmente posicionados na mesma cidade, que representa o depósito da instância. As demais cidades da instância são chamadas de intermediárias. O objetivo é minimizar o deslocamento total dos caixeiros, sendo que todas as rotas devem começar e terminar no depósito e todas as cidades intermediárias devem ser visitadas exatamente uma vez. A rota de cada caixeiro deve conter ao menos uma cidade além do depósito. Neste artigo, o termo "rota" é empregado ao conjunto de cidades visitadas por um caixeiro ordenadas pelo instante da visita, e o termo "solução" é empregado para uma solução completa do problema.

[Wang et al. 2013] apresentam quatro variações do MTSP, que estão representadas na Figura 2 através de grafos, nos quais os vértices em azul representam os depósitos. A solução em (a) corresponde à de um MTSP básico. Na solução em (b), todos os caixeiros partem do mesmo depósito, mas não retornam a ele depois de visitarem todos

os vértices intermediários. Esta variação é denominada MTSP com rotas abertas (*open paths*). Em (c), o número de depósitos é maior que 1, e cada caixeiro retorna ao seu próprio depósito, configurando o MTSP com múltiplos depósitos e rotas fechadas. Em (d) cada caixeiro também tem seu próprio depósito, mas não retorna a ele, consistindo no MTSP com múltiplos depósitos e rotas abertas. A metodologia proposta neste trabalho aplica o MTSP com único depósito e rotas fechadas, como em (a), para modelagem do problema real, conforme detalhado na Seção 2.

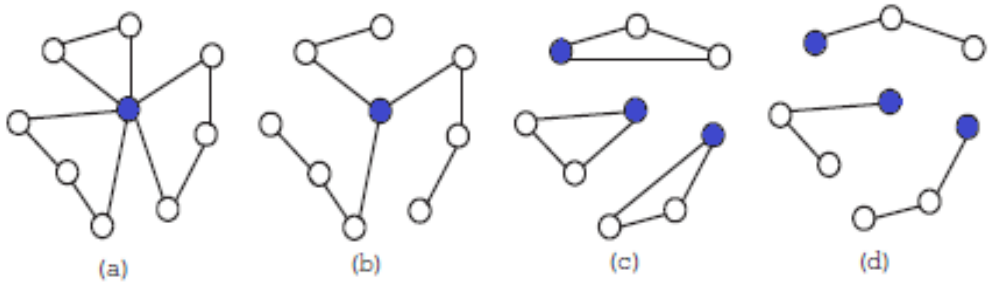

Figura 2. Variações do MTSP quanto ao formato das soluções: (a) único depósito e rotas fechadas, (b) único depósito e rotas abertas, (c) múltiplos depósitos e rotas fechadas, (d) múltiplos depósitos e rotas abertas. Fonte: [Wang et al. 2013].

Outra variação importante do MTSP está relacionada ao objetivo do problema. [Okonjo-Adigwe 1988] aponta que o MTSP sem restrições não considera importantes parâmetros de problemas reais, entre eles o equilíbrio da carga de trabalho (*workload balance*) entre os caixeiros. Para buscar o equilíbrio entre os custos das rotas em um MTSP, ao invés de minimizar o custo total da solução, o custo a ser minimizado passa a ser o da maior rota da solução, que é composta por m rotas, uma de cada caixeiro. Esta variação é também denominada MTSP com objetivo *minmax*, e corresponde ao objetivo adequado ao problema real em estudo, sendo utilizado nos experimentos apresentados na Seção 3.

Uma instância de MTSP pode ser representada em um grafo completamente conectado $G = (V, E)$, no qual V é o conjunto de n vértices v_j representando as cidades. E é o conjunto de arestas e_i que conectam todas as cidades entre si, de forma que cada aresta possui custo associado $\omega(e_i)$, que normalmente é uma medida de distância ou tempo. m equivale ao número de caixeiros e v_0 ao depósito. [Liu et al. 2009a] apresentam um modelo matemático para o MTSP objetivando a minimização do custo total das soluções e também a minimização da maior rota individual das soluções. Neste modelo, $C_1, C_2, ..., C_m$ correspondem às m rotas, ou ciclos fechados, que devem compor a solução; e $\omega(C_i)$ é o custo da rota i. As funções objetivo são:

$$Min \sum_{j=1}^{m} \sum_{e_i \in (C_j)} \omega(e_i) \tag{1}$$

$$Min \quad argmax[\omega(C_i)] \tag{2}$$

$$Sujeito \quad a \begin{cases} v_0 \in C_i(V)(i = 1, 2, ..., m) \\ \bigcup_{i=1}^{m} C_i(V) = G(V) \end{cases} \tag{3}$$

Neste modelo, a equação 1 limita o custo total das soluções MTSP e a equação 2 restringe o custo das rotas individuais de cada caixeiro. As restrições na equação 3 definem que um depósito e m rotas devem compor a solução [Liu et al. 2009a].

Dentre os trabalhos pesquisados relacionados ao MTSP, [Somhom et al. 1999] criam uma rede neural baseada em competição (*competition-based neural network*) para tratamento do *minmax* MTSP. Comparam seu algorimto com um algoritmo de rede elástica (*elastic net*) e com a heurísitica 2-opt generalizada, a partir de experimentos com instâncias modelo da TSPLIB. A TSPLIB é uma biblioteca digital disponível em *http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/* com várias instâncias de TSP e de problemas relacionados [Reinelt 1991].

[Tang et al. 2000] buscam melhorar a programação do processo de *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. Para isso, modelam o problema real como um MTSP com depósitos iniciais e finais distintos e minimizam o custo total das soluções com um algoritmo genético modificado especialmente desenvolvido para o problema. Avaliam a eficiência de seu algoritmo comparando-o com o algoritmo exato de Volgenant e Jonker, em experimentos realizados utilizando instâncias de MTSP aleatórias; e também comparando-o com o método utilizado na empresa, em experimentos com dados reais.

[Zhu and Yang 2003] minimizam o custo total das soluções para o MTSP básico considerando o equilíbrio da carga de trabalho em sua metodologia, aplicando uma abordagem baseada em um *self-organizing map* melhorado em conjunto com uma rede neural. Comparam os resultados obtidos com a utilização do mapa convencional com os do mapa melhorado em experimentos com instâncias da TSPLIB, considerando os custos totais das soluções e os tempos computacionais dos algoritmos ao realizar a comparação.

[Carter and Ragsdale 2006] e [Brown et al. 2007] trabalham com o MTSP básico, realizando experimentos minimizando o custo total e experimentos minimizando a maior rota individual das soluções. Aplicam um algoritmo genético com um novo cromossomo e operadores relacionados para tratamento do MTSP à instâncias aleatórias com 51, 100 e 150 cidades. Nos experimentos, comparam os resultados obtidos com seu novo cromossomo com os resultado de outros dois cromossomos já existentes, fixando o tempo de execução e analisando os custos das soluções.

[Yuan et al. 2013] apresentam um algoritmo genético para o MTSP básico com um novo operador de *crossover*, propondo a utilização de seu algoritmo na locação automatizada de portos marítimos. Realizam experimentos minimizando o custo total e minimizando a maior rota individual das soluções, sobre as mesmas instâncias de [Carter and Ragsdale 2006]. Comparam seu novo operador com outros três operadores já existentes e com os resultados de [Carter and Ragsdale 2006]. Também realizam experimentos com 5 instâncias pequenas, de 11 a 16 nós.

[Wang et al. 2013] abordam uma variação específica do problema, o MTSP com múltiplos depósitos e rotas abertas. Utilizando conceitos da Teoria dos Grafos, criam um modelo para simplicação de grafos que representam as instâncias de MTSP, que são aplicadas para criação das soluções otimizadas para as instâncias. Para validação do método, realizam experimentos sobre 37 instâncias da TSPLIB, com 2 a 10 caixeiros para cada instância, comparando os seus resultados com o custo das soluções ótimas para TSP das instâncias, já conhecidas.

[Arya et al. 2014] apresentam um algoritmo genético para o MTSP básico, minimizando o custo total das soluções. Executam seu algoritmo sobre uma instância com 50 nós e 3 caixeiros, mostrando em detalhes a melhora das soluções de acordo com as iterações. [Arya et al. 2014] dão uma importante contribuição ao citar o grande número de aplicações possíveis para sua metodologia: roteamento de veículos, cabeamento de computadores, perfuração de placas de circuito impresso, revisão de motores de turbina a gás, cristalografia de raios X, problema de separação de encomendas em depósitos, problema de programação de entrevistas, problema de planejamento de missões, projeto de redes de levantamento a partir de sistemas de navegação global por satélite, fabricação de microchips e sequenciamento de DNA.

[Venkatesh and Singh 2015] apresentam duas versões de um algoritmo baseado em *Artificial Bee Colony* (Colônia de Abelhas Artificial) para o MTSP básico. A *Artificial Bee Colony* é uma das mais recentes metodologias de otimização definidas, sendo baseada no comportamento inteligente de colônias de abelhas reais. [Venkatesh and Singh 2015] propõem também outro algoritmo para o MTSP, baseado em *Invasive Weed Optimization* (Otimização Invasiva por Ervas Daninhas, tradução literal), que é uma nova meta-heurística inspirada no processo biológico robusto de colonização e distribuição das ervas daninhas em um ecossistema. Os três algoritmos desenvolvidos são aplicados às mesmas instâncias de [Yuan et al. 2013], [Brown et al. 2007] e [Carter and Ragsdale 2006], tanto para o *minsum* quanto para o *minmax* MTSP. Os resultados obtidos são comparados com os dos outros autores, sendo também realizada uma comparação da eficiência dos três algoritmos com ou sem a aplicação da busca local 2-opt integrada a eles.

1.2.3. A Otimização por Colônia de Formigas

A Otimização por Colônia de Formigas (*Ant Colony Optimization*, ACO) está entre as principais meta-heurísticas usadas para o TSP, que também tem alcançado resultados consistentes para outros problemas de otimização combinatória [Kanda 2014]. A ACO recebe esta denominação por ser baseada na maneira como colônias de formigas reais encontram rotas mais curtas até suas fontes de alimento se comunicando através de trilhas de feromônio. O feromônio é uma substância odorífera e volátil depositada pelas formigas por onde passam e que tende a se acumular naturalmente em caminhos mais curtos.

Na ACO aplicada ao TSP, as soluções são geradas num processo iterativo no qual agentes inteligentes, as formigas, percorrem todas as cidades da instância escolhendo à cada iteração a próxima cidade de sua rota a partir de probabilidades calculadas considerando a distância entre as cidades e também a quantidade de feromônio entre elas. Essas probabilidades são calculadas a partir da Regra de Transição de Estado (RTE). Depois que constroem suas rotas, que equivale ao final de um ciclo, ocorre o processo de evaporação

do feromônio e, em seguida, as formigas depositam feromônio por onde passaram de acordo com a qualidade da solução criada: quanto melhor a solução, mais feromônio é deposito nas arestas que a compõem. A evaporação e o depósito são controlados pela Regra de Atualização de Feromônio (RAF), que determina como o feromônio influenciará as escolhas das formigas nos próximos ciclos.

A partir do primeiro algoritmo ACO, denominado *Ant System* (AS) [Dorigo et al. 1996], que obteve resultados encorajadores para o TSP, a ACO tem sido aplicada com sucesso para vários problemas NP-difíceis de otimização combinatória. Dentre os algoritmos que surgiram a partir de aperfeiçoamentos no *Ant System*, destacam-se: o *Elitist Ant System*, o *Ant-Q*, o *MAX-MIN Ant System*, o *Rank-Based Ant System* (RBAS) e o *Ant Colony System* (ACS) [Dorigo and Stutzle 2004]. As duas variações que servem de base para a metodologia apresentada nesta dissertação são o RBAS e o ACS.

O RBAS, também abreviado como AS_{rank}, foi apresentado em [Bullnheimer et al. 1999] e altera a forma como o feromônio é depositado no AS. Ao contrário de todas as formigas depositarem feromônio ao final do ciclo, como ocorre no AS, apenas as $w-1$ formigas que geraram as melhores soluções depositam feromônio por onde passaram, em quantidade proporcional à posição no *rank* das melhores soluções geradas no ciclo, sendo w um parâmetro. A formiga que criou a melhor solução até o momento atual da execução do algoritmo, a *best so far solution*, também deposita feromônio ao final do ciclo, em maior quantidade que as outras formigas.

O ACS foi outro algoritmo que surgiu a partir de aperfeiçoamentos no AS, tendo sido apresentado em [Dorigo and Gambardella 1997], e difere dele em três pontos principais: utiliza uma RTE mais agressiva; a RAF determina que a evaporação e o depósito de feromônio, ao final de cada ciclo, acontecem apenas nas arestas pertencentes à *best-so-far solution*; 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 diferença importante é em relação ao número N de formigas que percorrem o grafo e constroem suas próprias soluções simultaneamente: enquanto no *Ant System* o melhor valor encontrado experimentalmente por [Dorigo and Stutzle 2004] foi $N = n$, sendo n o número de cidades da instância, para o ACS o melhor valor encontrado foi $N = 10$.

O TACO (*Team Ant Colony Optimization*, Otimização por Colônia com Equipes de Formigas) é o algoritmo desenvolvido por [Vallivaara 2008] para o planejamento de rotas para múltiplos robôs em um ambiente hospitalar. O TACO adapta o ACS para instâncias de MTSP substituindo cada formiga do ACS, que gera soluções para o TSP, por um time de formigas formado por m elementos, onde cada formiga de um time corresponde a um caixeiro na construção de uma solução para MTSP.

No início da construção de uma solução pelo TACO, todas as formigas de todos os times são posicionadas no depósito. Para realizar a distribuição da carga de trabalho, a formiga que tem a menor rota parcial, em qualquer momento da construção do solução, escolhe sua próxima cidade de acordo com a regra de transição de estado do ACS. Segundo [Vallivaara 2008], este método é eficiente para a distribuição da carga de trabalho, mas em muitos casos as escolhas das formigas tendem para soluções não ótimas. Para tratar este problema, é verificado a cada escolha se, caso a cidade escolhida fosse cedida

a outra formiga do time, não haveria um tamanho total menor da solução, considerando também o retorno ao depósito. Se houver, é a formiga com menor rota parcial que se movimenta, podendo realizar uma nova escolha do seu próximo vértice a partir da RTE.

Matematicamente, 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; $CP(k)$ o custo parcial da rota já percorrida pela formiga k; e tendo sido escolhido o vértice v_j como seu próximo vértice de acordo com a RTE; o método consiste em verificar, antes da formiga de mover, se existe qualquer outra formiga l da instância em que

$$c(v_l, v_j) + c(v_j, v_0) + CP(l) < c(v_k, v_j) + c(v_j, v_0) + CP(k) \quad (4)$$

Se houver outra formiga l da instância que atenda ao primeiro termo da inequação 4, é permitido que esta nova formiga l escolha seu próximo vértice novamente de acordo com a RTE, e se movimente para o novo vértice escolhido. Outra técnica utilizada por [Vallivaara 2008] é a aplicação de algoritmos de buscas locais para melhoramento das soluções geradas. A busca local 2-opt, que tenta trocar duas arestas quaisquer da solução, é aplicada a todas as soluções geradas. A busca local 3-opt, que tenta comutar quaisquer três arestas de uma solução ao mesmo tempo, é aplicada apenas à melhor solução dentre as N geradas simultaneamente pelos times em um ciclo.

Além do TACO de [Vallivaara 2008], outros trabalhos aplicando a ACO ao MTSP foram encontrados na literatura. [Junjie and Dingwei 2006] definem seu trabalho como o primeiro que aplica um algoritmo ACO ao MTSP com restrição de habilidade (MTSP *with ability constraint*). Segundo eles, o objetivo geral do MTSP, de minimizar a distância total, chamado de critério mínimo (objetivo *minsum*), gera soluções desequilibradas. Como em problemas do mundo real todos os caixeiros têm habilidade semelhante, os autores definem um número máximo de cidades que cada caixeiro pode visitar durante a construção de uma solução para tentar equilibrar as rotas. O algoritmo utiliza as mesmas RTE e RAF do *Ant System*.

Ao iniciar a construção de uma solução, no algoritmo de [Junjie and Dingwei 2006] cada um dos m caixeiros recebe um número aleatório de nós a visitar, de forma que a soma de todos esses números seja igual ao número de nós da instância menos um, para desconsiderar o depósito. Em seguida, N formigas são posicionadas no depósito, com cada formiga possuindo sua própria memória, que armazena os nós já visitados pela formiga durante a construção de uma solução. Cada formiga escolhe então sua primeira cidade não visitada, independentemente das outras formigas, e se desloca para ela, de forma que todas as formigas se desloquem para a primeira cidade do primeiro caixeiro. O processo de escolha e movimento se repete para todas as formigas até que o primeiro caixeiro atinja seu número pré-calculado de cidades. Neste momento todas as formigas retornam ao depósito e iniciam as visitas para o segundo caixeiro. O processo se repete até o último caixeiro visite sua última cidade, instante em que todas as N formigas criaram, cada uma, uma solução completa para o MTSP. Em resumo, [Junjie and Dingwei 2006] buscam soluções mais equilibradas do que as do MTSP básico limitando o número de cidades de cada caixeiro pode visitar, chamando-o de MTSP com restrição de habilidade. O objetivo continua sendo o critério

mínimo de encontrar a solução com a menor distância total (objetivo *minsum*), mas não há garantia, porém, que os comprimentos das rotas individuais dos caixeiros estejam equilibrados nas soluções geradas.

Assim como [Junjie and Dingwei 2006], [Wang et al. 2007] determinam em sua abordagem que o número máximo de cidades que um caixeiro pode visitar fique limitado a um intervalo predefinido, utilizando as regras do *Ant System* na construção de soluções para o MTSP básico com objetivo *minsum*. Utilizam seu algoritmo para o agrupamento e roteamento de nós sensores em redes de sensores multimedia sem fio, visando o consumo eficiente de energia. Comparam seus resultados com um algoritmo de *simulated annealing* e com um algoritmo genético, aplicando-os a uma instância de 200 sensores e 3 grupos, analisando o consumo de energia resultante das três abordagens.

[Liu et al. 2009a] desenvolve um algoritmo para o MTSP básico aplicando a RTE do *Ant Colony System* (ACS) e a RAF do *MAX-MIM Ant System* (MMAS). Assim como no MMAS, o mecanismo de reinicialização de feromônio é implementado para fugir do comportamento de estagnação. Para melhorar a qualidade do algoritmo, [Liu et al. 2009a] submete todas as soluções criadas a 4 heurísticas de busca local. Em [Liu et al. 2009b], o algoritmo desenvolvido é aplicado na distribuição de cigarros em uma empresa chinesa de grande porte. Para isso, foi realizado um experimento com dados reais com 2153 pontos de entrega, que resultou na diminuição de 8 para 7 o número de veículos necessários para realização da distribuição.

[Ghafurian and Javadian 2011] trabalham com uma variação específica do MTSP: com múltiplos depósitos e rotas fechadas, de forma que cada caixeiro retorna para o seu próprio depósito. Para criação de soluções, desenvolvem um algoritmo que implementa as regras do *Ant System*, de forma que cada formiga constrói uma solução completa e o número máximo de cidades que cada uma delas pode visitar fica limitado a um intervalo predefinido. Utilizando instâncias próprias criadas aleatoriamente com 30, 40, 50 e 60 nós, [Ghafurian and Javadian 2011] realizam uma ajuste de parâmetros em seu algoritmo para: o número de formigas, o coeficiente de persistência, a quantidade inicial de feromônio e o critério de parada; e comparam os resultados com as soluções ótimas obtidas pelo software de otimização Lingo 8.0, em relação ao tempo computacional necessário para criação das soluções.

[Costa et al. 2012] trabalham com o MTSP básico com objetivo *minsum*. Desenvolvem um algoritmo ACO no qual várias colônias produzem soluções paralelamente e cada formiga constrói uma solução completa. Mais um fator é incluido na RTE do ACS, para considerar o quanto de feromônio contém as arestas das outras colônias, sendo implementada a mesma RAF do ACS. Realizam experimentos com 6 instâncias da TSPLIB, de 124 a 789 nós, sendo três delas assimétricas. Nas conclusões, [Costa et al. 2012] informam que o seu algoritmo auxilia no processo de decisão relacionado ao planejamento de rotas nos reparos em caso de apagões elétricos.

[Vasagam 2012], assim como [Liu et al. 2009a], apresenta um algoritmo para o MTSP básico aplicando a RTE do *Ant Colony System* (ACS) e a RAF do *MAX-MIM Ant System* (MMAS). O objetivo é o de minimizar o custo total das soluções. Para realização dos experimentos, implementa o algoritmo ACO e um algoritmo genético e compara os custos totais das soluções obtidas com os dois algoritmos.

[YousefiKhoshbakht and Sedighpour 2012] utilizam um algoritmo construtivo denominado *Sweep Algorithm* para construir uma solução inicial, e uma variação do *Elitist Ant System* combinada com a busca local 3-opt para melhorar a solução inicial. Durante a construção desta primeira solução, cada caixeiro visita um número de cidades n/m e então retorna ao depósito. Isto garante uma distribuição equilibrada de cidades entre os caixeiros, mas não há garantia que os custos das rotas individuais sejam semelhantes. A distribuição da carga de trabalho ocorre antes do início da construção da solução e os caixeiros retornam obrigatoriamente ao depósito quando seu número de cidades é alcançado. Em [Yousefikhoshbakht et al. 2013], outro algoritmo ACO é apresentado, no qual mais um fator é incluído na RTE do *Ant System*. Este algoritmo aplica a lista de nós candidatos com 30% do total de nós da instânica limitada a 20 nós. Uma nova RAF é definida e as buscas locais *insert*, *swap* e 2-opt são utilizadas para melhoramento das soluções obtidas.

[Zhou and Yao 2013] criam uma versão melhorada do *MAX-MIM Ant System* para tratamento do MTSP básico com objetivo *minsum*. No algoritmo desenvolvido, a visibilidade é substituída por um valor variável na RTE e são realizadas modificações na RAF do MMAS. A fim de obter um melhor desempenho, executam experimentos para inferência de parâmetros com instâncias com 30, 50 e 100 nós criadas aleatoriamente. A eficiência da metodologia é comprovada comparando os resultados obtidos pelo MMAS melhorado com sua versão original adaptada para o MTSP.

2. Metodologia

A nossa metodologia consiste na criação de instâncias de MTSP a partir do ambiente real do atendimento comercial das empresas de distribuição de energia elétrica e na aplicação de algoritmos baseados na ACO para construção de soluções otimizadas para distribuição das ordens de serviço e roteamento das equipes. As posições das ordens de serviço a executar representam as cidades da instância de MTSP e as equipes de atendimento representam os caixeiros.

Devido à grande variação do tempo necessário para atendimento das ordens de serviço, é difícil estabelecer uma capacidade para as equipes, como ocorre na descrição geral do PRV, a fim de limitar os custos individuais das rotas geradas. Além disso, normalmente todas as ordens conhecidas são executadas durante o dia de trabalho, independente do tempo gasto pelas equipes. Estes foram os motivos que levaram à escolha do MTSP para modelagem do problema real, o qual em sua definição geral não determina um custo máximo para as rotas individuais das equipes.

Neste artigo, dois algoritmos ACO adaptados para o MTSP, denominados STACS (*Single Team Ant Colony System*) e STRBAS (*Single Team Rank-Based Ant System*), são aplicados à instâncias reais. A ACO foi escolhida devido ao sucesso obtido com outros problemas de otimização combinatória; e a fim de explorar novas formas de aplicá-la ao MTSP. Os resultados obtidos pelo STACS e pelo STRBAS são comparados na Seção 3.

As variações de algoritmos ACO aplicadas ao TSP neste artigo diferem em dois pontos: a forma como uma formiga escolhe a próxima cidade de sua rota, que é controlado pela Regra de Transição de Estado (RTE); e a forma que o feromônio é inicializado e atualizado durante a execução do algoritmo, que é controlado pela Regra de Atualização de Feromônio (RAF). Dessa forma, um algoritmo ACO para MTSP geral é apresentado na Seção 2.1 e, a partir deste algoritmo, são implementadas duas adaptações para MTSP:

o STACS, baseado no ACS, detalhado na Seção 2.2; e o STRBAS, baseado no RBAS, detalhado na Seção 2.3. A principal referência para o algoritmo ACO para MTSP geral da Seção 2.1 é o trabalho de [Vallivaara 2008].

2.1. Algoritmo ACO para MTSP

O pseudocódigo do algoritmo ACO para MTSP geral desenvolvido neste trabalho está representado no Algoritmo 1.

Algoritmo 1: Algoritmo ACO para MTSP

Entrada: Instância MTSP e parâmetros ACO
Saída: Melhor solução MTSP criada

```
1  Início
2      n ← número de cidades da instância, incluindo o depósito;
3      m ← número de caixeiros da instância;
4      v_0 ← cidade depósito da instância;
5      N ← número de soluções geradas em cada ciclo;
6      Defina o objetivo do MTSP;
7      Construa a matriz de custos da instância;
8      Inicialize a matriz de feromônio;
9      Enquanto o critério de parada não for atingido faça
10         Para contador_de_soluções ← 1 até N faça
11             Crie um time com m formigas;
12             Esvazie a lista de cidades visitadas da colônia;
13             Posicione as m formigas do time em v_0;
14             Enquanto houver cidades não visitadas faça
15                 Selecione uma formiga k do time;
16                 Crie a lista de candidatos para k;
17                 Escolha a próxima cidade j da rota de k;
18                 Verifique o movimento selecionado;
19                 se houver uma formiga l que resulta num melhor movimento então
20                     Selecione a formiga l;
21                     Atualize a lista de candidatos para l;
22                     Escolha uma nova próxima cidade j para l;
23                 fim
24                 Movimente a formiga selecionada para a cidade escolhida;
25                 Insira a cidade visitada na lista de cidades visitadas;
26                 Aplique a atualização local de feromônio à aresta percorrida;
27             Fim
28             Retorne todas as m formigas para v_0;
29             Aplique a busca local à solução MTSP criada;
30             Atualize a melhor solução do ciclo S_m_ciclo;
31         Fim
32         Atualize a melhor solução da execução S_m_exec;
33         Aplique a atualização global de feromônio;
34     Fim
35     Retorne S_m_exec;
36 Fim
```

As entradas do Algoritmo 1 são a instância MTSP e os parâmetros ACO. Nesta fase do trabalho, as soluções criadas são para o MTSP com único depósito e rotas fechadas. Dessa forma, a instância é composta: pela matriz de custos relacionando todas as cidades; pelo número m de caixeiros; e pela definição de qual das cidades corresponde ao depósito, que será o ponto inicial e final de todas as rotas.

Os parâmetros ACO são descritos à medida que são necessários durante a apresentação das etapas do algoritmo, e os valores utilizados nos experimentos estão relacionados na Seção 3.1.2. A saída do algoritmo é a melhor solução MTSP encontrada durante toda a execução. As linhas em negrito correspondem às regras específicas de cada um dos dois algoritmos adaptados, que são detalhadas nas Seções 2.2 e 2.3.

2.1.1. Inicialização

O algoritmo ACO para MTSP desenvolvido inicia com a declaração das variáveis principais. Nas linhas 2 a 4 são declarados: o valor inteiro n, que equivale ao total de cidades da instância, incluindo o depósito; o número m de caixeiros; e a cidade v_0 correspondente ao depósito. Esses dados são fornecidos pela instância de MTSP. Na linha 5 é definido o valor de N, que corresponde ao número de soluções que serão criadas em cada ciclo do algoritmo. Nos algoritmos ACO originais aplicados ao TSP, N equivale ao número de formigas que geram soluções simultaneamente durante um ciclo. No ACS aplicado ao TSP, o melhor valor encontrado experimentalmente foi $N = 10$ ([Dorigo and Stutzle 2004]), o que significa que 10 formigas são posicionadas inicialmente em cidades distintas da instância e se movimentam alternadamente de forma que, ao final do ciclo, 10 soluções para TSP foram criadas. Para o RBAS aplicado ao TSP, o melhor valor encontrado foi $N = n$, o que significa que n soluções foram criadas ao final de um ciclo, sendo n o número de cidades da instância.

[Vallivaara 2008] adapta o ACS para o MTSP transformando cada formiga da ACO para MTSP em um time composto por m formigas, cada formiga representando um caixeiro. Dessa forma, cada time gera uma solução completa para MTSP. Em cada ciclo, N times se movimentam alternadamente, de forma que N soluções foram criadas no seu término. Em nosso algoritmo ACO para MTSP, cada formiga de um time também corresponde a um caixeiro, mas apenas um time percorre a instância por vez. As soluções são geradas uma após a outra e, quando N foram geradas, o ciclo é finalizado.

Na linha 6 é definido o objetivo do MTSP, que pode ser o de minimizar o custo total da solução ou o de minimizar o custo da maior rota individual da solução. Na literatura, esses objetivos são chamados de *minsum* e *minmax*, respectivamente [Somhom et al. 1999]. Em todos os experimentos apresentados neste artigo, o protótipo foi configurado com o objetivo *minmax*, que produz soluções com custos equilibrados entre as rotas dos caixeiros, como desejado no ambiente real das equipes de atendimento.

A matriz de custos da instância é construída na linha 7. Caso estejam sendo aplicadas as distâncias euclidianas, a matriz de custos é simétrica e é calculada a partir das coordenadas planas das cidades. Caso estejam sendo utilizados custos previstos, a fim de uma melhor representação do ambiente real, a matriz é assimétrica e todos os custos terão que ser fornecidos para confecção da matriz. A matriz é assimétrica porque deslocamen-

tos entre dois pontos reais podem ser diferentes de acordo com o sentido (ir de A para B tem um custo diferente que ir de B para A, devido a vias de mão única, por exemplo).

Na inicialização da matriz de feromônio, que ocorre na linha 8, todas as arestas, ou combinações duas a duas entre todas as cidades da instância, recebem a mesma quantidade de feromônio, indicando que nenhuma informação foi ainda adquirida sobre a instância. A quantidade inicial de feromônio é definida pelo parâmetro τ_0, sendo os melhores valores obtidos experimentalmente para este parâmetro específicos para cada variação de algoritmo ACO. Os valores de τ_0 utilizados pelo STACS e pelo STRBAS são detalhados nas Seções 2.2 e 2.3, respectivamente.

Na linha 9 iniciam-se os ciclos do algoritmo, que são limitados pelo critério de parada. No protótipo desenvolvido três critérios diferentes podem ser definidos: um número máximo de ciclos; um número máximo de ciclos sem que a melhor solução da execução seja atualizada; ou um limite temporal, em segundos. Nos experimentos deste artigo, o critério de parada foi fixado em 1.000 ciclos para todas as execuções. Na linha 10 inicia-se a construção das soluções, num total de N soluções criadas por ciclo.

2.1.2. Construção das soluções

A construção de uma solução começa na linha 11 do Algoritmo 1 com a criação de um time com m formigas, cada uma representando um caixeiro da instância. Apenas um time constrói uma solução de cada vez. O time possui uma lista de cidades visitadas comum, que é esvaziada na linha 12, indicando que todas as cidades ainda são permitidas nos movimentos. Em seguida, na linha 13, todas as formigas do time são posicionadas no depósito, de onde iniciarão suas rotas.

Na linha 14 iniciam-se os movimentos do time, que consistem no deslocamento de uma das formigas de sua cidade atual para a próxima cidade de sua rota. A formiga escolhida para se movimentar é aquela que possuir a rota parcial com menor custo, como aplicado por [Vallivaara 2008], o que ocorre na linha 15. Empates são resolvidos aleatoriamente (no protótipo, é selecionada a formiga com menor número de identificação). Esta estratégia é adequada ao *minmax* MTSP ao tentar equilibrar os custos das rotas já na seleção da formiga a se movimentar. Após selecionada uma formiga, na linha 16 é criada a lista de candidatos, que é formada pelas cl_{size} cidades mais próximas da cidade onde se encontra a formiga selecionada. O objetivo da lista de candidatos é restringir o espaço de busca, incluindo apenas as arestas mais prováveis de comporem uma boa solução, reduzindo o tempo computacional. Em todos os experimentos realizados, cl_{size} é fixado em 20.

A linha 17 corresponde à Regra de Transição de Estado (RTE) do algoritmo ACO, que é quando uma formiga escolhe a próxima cidade de sua rota. Os dois algoritmos adaptados utilizam diferentes RTEs, que estão detalhadas nas Seções 2.2 e 2.3.

Após escolhida a próxima cidade da formiga selecionada, é realizada uma verificação antes da execução do movimento, que começa a partir da linha 18. Esta verificação é necessária porque, como explica [Vallivaara 2008], as escolhas das formigas para se movimentar de acordo com o menor custo parcial de suas rotas tendem para soluções não ótimas. Para tratar este problema, é verificado a cada movimento se, caso

a cidade escolhida fosse cedida a qualquer outra formiga do time, não resultaria em um deslocamento menor que o da formiga selecionada, considerando também o retorno ao depósito. Isto significa que se houver qualquer outra formiga do time que: partindo de sua cidade atual, visite a cidade escolhida, retorne ao depósito, e este trajeto resulte num deslocamento menor que o da formiga selecionada para realizar o mesmo trajeto, é permitido que a formiga com menor deslocamento se movimente. Na linha 20, a formiga que resulta no menor deslocamento é selecionada, não se movendo obrigatoriamente para a cidade escolhida pela formiga anterior, sendo a lista de candidatos refeita para a posição da nova formiga e a RTE executada novamente, nas linhas 21 e 22, onde uma próxima cidade diferente pode ser escolhida.

Na linha 24, a formiga selecionada inclui a cidade escolhida em sua rota e atualiza a sua posição. A cidade que acabou se ser visitada é incluída na lista de cidades vistadas na linha 25, para que não volte a ser selecionada durante a construção da solução atual. Na linha 26 ocorre a Atualização Local de Feromônio (ALF), na qual uma quantidade de feromônio é removida da aresta que acabou de ser percorrida, para aumentar a exploração de caminhos alternativos. A ALF é implementada apenas no STACS, e está detalhada na Seção 2.2. Os movimentos do time, da linha 15 à linha 26, se repetem até que não haja mais cidades a serem visitadas. Quando isso ocorre, todas as formigas retornam ao depósito, na linha 28, concluindo a solução para MTSP com único depósito e rotas fechadas.

2.1.3. Busca local, avaliação das soluções e finalizações dos ciclos e da execução

Nesta adaptação de algoritmos ACO para MTSP, a busca local 2-opt é aplicada a todas as soluções geradas, o que ocorre na linha 29 do Algoritmo 1. A busca local 2-opt tenta melhorar uma solução substituindo as arestas que a compõe duas a duas. Se uma troca resulta em solução melhor, a nova solução substitui a anterior. A busca 2-opt implementada realiza todas as substituições possíveis nas rotas individuais das soluções (*intra routes*) e também entre rotas diferentes (*inter routes*). No Experimento 5, apresentado na Seção 3.4, a busca local 3-opt foi aplicada à cada atualização da melhor solução da execução dos algoritmos.

Após a solução construída ser submetida à busca local, ocorre a atualização da melhor solução gerada no ciclo, na linha 30. Se a nova solução gerada for melhor do que a que se encontra armazenada em S_{m_ciclo}, esta substitui a anterior. Quando ocorre a finalização de um ciclo, é verificado na linha 32 se a melhor solução do último ciclo é melhor que a melhor solução encontrada até o momento em toda execução, que fica armazenada em S_{m_exec}. Se for, a melhor solução da execução é atualizada.

A Atualização Global de Feromônio (AGF) acontece ao final de cada ciclo, e ocorre de forma diferente no STACS e no STRBAS. As AGFs dos dois algoritmos adaptados estão detalhadas em 2.2 e 2.3, respectivamente. Quando o número de ciclos realizados atinge o critério de parada, o algoritmo finaliza e retorna S_{m_exec}, que armazena a melhor solução gerada em toda a execução.

2.2. STACS

Os algoritmos STACS e STRBAS foram implementados a partir do algoritmo ACO geral mostrado no Algoritmo 1, baseados no ACS e no RBAS, respectivamente. As duas adaptações, replicando os algoritmos originais, são diferentes nas seguintes linhas do algoritmo geral: na linha 5, porque cada algoritmo adaptado cria uma quantidade diferente de soluções por ciclo; na linha 8, na qual cada algoritmo inicializa a matriz de feromônio com um valor diferente; nas linhas 17 e 22, onde são aplicadas as RTEs específicas de cada algoritmo; e nas linhas 26 e 33, onde ocorre a atualização da matriz de feromônio, que também ocorre a partir de RAFs específicas de cada algoritmo. Nesta Seção é detalhado o STACS, e na Seção 2.3 o STRBAS.

O STACS foi o primeiro algoritmo ACO adaptado para MTSP aplicado à instâncias do problema real nesta pesquisa [Barbosa et al. 2015]. O melhor valor para o parâmetro N, inferido experimentalmente por [Dorigo and Stutzle 2004] a partir da aplicação do ACS ao TSP, foi $N = 10$. Para τ_0, o melhor valor inferido por [Dorigo and Stutzle 2004] é obtido a partir da Equação 5:

$$\tau_0 = 1/nC^{nn} \tag{5}$$

Na Equação 5, n corresponde ao número de cidades da instância e C^{nn} ao custo da solução do algoritmo do vizinho mais próximo aplicado à instância. [Dorigo and Stutzle 2004] informam que, de fato, qualquer algoritmo razoável para o cálculo de C^{nn} pode ser utilizado. Nesta metodologia, a solução do vizinho mais próximo para MTSP foi criada da seguinte forma: todos os caixeiros partem do depósito; o caixeiro com a menor rota parcial se move para a cidade mais próxima de sua posição atual; o movimento se repete até que não haja mais cidades a serem visitadas; todos os caixeiros retornam ao depósito. À C^{nn} foi atribuído o custo total da solução calculada conforme descrito.

Tanto o ACS quanto o RBAS utilizam RTEs baseadas na regra do *Ant System*, o primeiro algoritmo da ACO. A RTE do *Ant System* está representada na Equação 6:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum [\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta} & \text{se } j \text{ é uma aresta permitida à } k \\ 0 & \text{caso contrário} \end{cases} \tag{6}$$

A Equação 6 determina a probabilidade de uma cidade ser escolhida por uma formiga em um movimento. i representa a cidade na qual a formiga k se encontra, e j uma das possíveis cidades para a qual ela pode se deslocar. As cidades permitidas à formiga são aquelas ainda não visitadas até a fase atual da construção da solução. Cada formiga armazena as cidades já visitadas em sua memória, que é denominada como lista tabu. $\tau_{ij}(t)$ corresponde à quantidade de feromônio presente na aresta entre as cidades 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 da próxima cidade pela formiga.

A RTE do ACS e do STACS está mostrada na Equação 7, na qual j representa a cidade escolhida por uma formiga k que se encontra no vértice i ao se movimentar.

$$j = \begin{cases} \text{argmax}_{l \in S_i^k}\{\tau_{il}[\eta_{il}]^\beta\}, & \text{se } q \leq q_0; \\ J, & \text{caso contrário.} \end{cases} \quad (7)$$

A Equação 7 é chamada de regra proporcional pseudoaleatória, pois o parâmetro q_0 define a porcentagem das escolhas que serão feitas de forma determinística pelas formigas, onde $0 \leq q_0 \leq 1$. 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 $l \in S_i^k$ 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 RTE do *Ant System* (Equação 6), com a única diferença que o parâmetro α foi excluído, ou seja, seu valor foi fixado em 1. O melhor valor definido experimentalmente por [Dorigo and Stutzle 2004] foi $q_0 = 0.9$, o que significa que 90% das escolhas serão determinísticas.

A RAF de um algoritmo ACO inclui uma regra que determina a evaporação do feromônio. A regra de evaporação do *Ant System* está representada na Equação 8.

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \forall (i, j) \in A, \quad (8)$$

Na Equação 8, τ_{ij} corresponde à quantidade atual de feromônio na aresta. ρ é um parâmetro de define a taxa de evaporação de forma que $0 < \rho \leq 1$, e A é o conjunto formado por todas as arestas da instância.

No ACS, a evaporação de feromônio ocorre como no AS (Equação 8), só que apenas nas arestas que fazem parte da *best so far solution*, e não em todas as arestas da instância como no AS. A RAF do ACS define que a atualização de feromônio ocorre em dois momentos, além da evaporação ao final do ciclo: uma atualização global (AGF), ao final de cada ciclo do algoritmo; e uma atualização local (ALF), logo depois que uma formiga se move de uma cidade a outra, ou seja, inclui mais uma aresta na sua rota. A regra de AGF do ACS e do STACS está apresentada na Equação 9, onde ρ é o coeficiente de evaporação de feromônio, cujo melhor valor encontrado por [Dorigo and Stutzle 2004] para o ACS aplicado ao TSP foi $\rho = 0.1$.

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall (i, j) \in T^{bs}, \quad (9)$$

$\Delta\tau_{ij}^{bs} = 1/C^{bs}$, onde T^{bs} corresponde ao conjunto de arestas que fazem parte da melhor solução encontrada até o momento atual da execução do algoritmo, a *best so far solution*, e C^{bs} corresponde ao custo de T^{bs}.

A ALF do ACS e do STACS está apresentada na Equação 10, na qual ξ é um parâmetro cujo melhor valor experimentalmente calculado por [Dorigo and Stutzle 2004] aplicando o ACS ao TSP foi $\xi = \rho = 0.1$:

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0, \quad (10)$$

2.3. STRBAS

O STRBAS é a adaptação do RBAS para o MTSP utilizando a estrutura mostrada no Algoritmo 1. O melhor valor para o parâmetro N, que corresponde ao número de soluções que serão criadas em cada ciclo do algoritmo, foi calculado experimentalmente como sendo igual a n, quando o RBAS foi aplicado à instâncias de TSP [Dorigo and Stutzle 2004]. n corresponde ao número de cidades da instância. Este valor foi mantido para o STRBAS nos experimentos computacionais. Isto significa que, para uma instância com 65 cidades incluindo o depósito, por exemplo, foram criadas 65 soluções em cada ciclo do STRBAS.

Na inicialização da matriz de feromônio dos algoritmos ACO, todas as arestas recebem a mesma quantidade de feromônio, que é definida pelo parâmetro τ_0. Para o RBAS aplicado ao TSP, o melhor valor obtido experimentalmente por [Dorigo and Stutzle 2004] para τ_0 é obtido pela Equação 11.

$$\tau_0 = 0.5w(w-1)/\rho C^{nn} \tag{11}$$

Na Equação 11, w é um parâmetro que também será utilizado para definir o número de soluções que serão consideradas para atualização do feromônio; e ρ corresponde à taxa de evaporação de feromônio. C^{nn} corresponde ao custo da solução criada quando aplicando o algoritmo do vizinho mais próximo à instância, calculada da mesma forma como descrito na Seção 2.2.

O RBAS e o STRBAS compartilham a mesma RTE do *Ant System*, apresentada na Equação 6. O RBAS não aplica a Atualização Local de Feromônio (ALF), de forma que a linha 26 do Algoritmo 1 não é executada para o STRBAS. Em sua RAF, o RBAS implementa a evaporação em todas as arestas da matriz de feromônio, aplicando a mesma regra do AS, que está representada na equação 8.

Na AGF do RBAS, é permitido que as $w-1$ melhores formigas depositem feromônio, em quantidade proporcional à sua posição no *rank* das melhores soluções geradas no ciclo. w é um parâmetro cujo melhor valor inferido experimentalmente por [Dorigo and Stutzle 2004] para o TSP foi 6. A formiga que gerou a melhor solução até o momento atual da execução também deposita feromônio, em maior quantidade que as anteriores. No STRBAS, são os times que geraram as $w-1$ melhores soluções e a *best so far solution* que realizam o depósito (linha 33 do Algoritmo 1). A regra da AGF do RBAS está representada na Equação 12.

$$\tau_{ij} = \tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs} \tag{12}$$

Na Equação 12, r equivale à posição no *rank* formado pelas melhores soluções do ciclo. $\Delta\tau_{ij}^{r}$ é igual a $1/C^{r}$, onde C^{r} é o custo da solução criada pelo time de *rank* r. A quantidade de feromônio depositada pelo time que gerou a *best so far solution*, $\Delta\tau_{ij}^{bs}$, é igual a $1/C^{bs}$, onde C^{bs} é o custo da *best so far solution*.

3. Experimentos Computacionais

Os experimentos apresentados em [Barbosa et al. 2015] consistiram na criação de 17 instâncias de MTSP a partir de dados reais de uma agência de atendimento e na otimização de soluções para essas instâncias utilizando o STACS, que é uma adaptação do ACS para MTSP desenvolvida nesta pesquisa, que se encontra detalhada na Seção 2.2. Este artigo estendido inclui também os resultados do STRBAS, uma nova adaptação ACO para o MTSP desenvolvida nesta pesquisa e detalhada na Seção 2.3. O STRBAS foi aplicado sobre as mesmas instâncias reais, possibilitando a comparação entre os resultados obtidos pelos dois algoritmos adaptados. Para possibilitar a replicação destes experimentos, em *https://github.com/denilsonfag/STRBAS* encontram-se disponíveis o código fonte do protótipo desenvolvido e os dados reais utilizados.

Neste artigo são apresentados cinco experimentos. No experimento 1 é realizada uma seleção do parâmetro β, presente nos dois algoritmos adaptados. No experimento 2 os dois algoritmos são comparados utilizando instâncias representadas por matrizes simétricas compostas pelas distâncias euclidianas entre os pontos a serem percorridos pelas equipes. No experimento 3 são utilizadas matrizes de custos assimétricas, compostas por distâncias previstas de deslocamento com veículo extraídas da ferramenta de consulta cartográfica [MapQuest 2015]. O experimento 4 utiliza custos temporais previstos de deslocamento para composição das matrizes, também obtidas de [MapQuest 2015]. O objetivo dos experimentos 3 e 4 é comprovar a aplicabilidade da nossa metodologia à dados que representem de forma mais fiel as rotas a serem percorridas pelas equipes. No experimento 5 o TACO, o STACS e o STRBAS são comparados a partir da otimização de instâncias da TSPLIB.

3.1. Detalhes experimentais

Para realizar a seleção inicial de β no experimento 1 foi selecionada a maior instância dentre as 17 obtidas a partir dos dados reais, que foi submetida aos dois algoritmos adaptados com β assumindo valores entre 1 e 5. Nos experimentos 2, 3 e 4 os dois algoritmos são aplicados às 17 instancias reais com o melhor valor inferido para β no experimento 1. As demais configurações, apresentadas na Seção 3.1.2, são iguais para todos os experimentos, com exceção do experimento 5, no qual são aplicados valores específicos para β e ρ, mostrados na descrição do experimento na Seção 3.4.

3.1.1. Base de dados reais

A base de dados reais utilizada nos experimentos consiste no histórico de atendimento do dia 1 de fevereiro a 23 de fevereiro de 2014 do município de Cornélio Procópio, Paraná. Os dados foram obtidos a partir do sistema de gerenciamento das ordens de serviço da COPEL Distribuição que, dentre outras funções, armazena as informações sobre as ordens desde a sua geração até a sua conclusão.

Para realização desta experimentação, as ordens foram separadas quanto ao seu dia de execução, que representa o dia em que uma equipe de atendimento realizou as atividades solicitadas na ordem. Em seguida, a partir das coordenadas UTM que indicam as posições de execução das ordens, foi montada uma instância de MTSP para cada dia de trabalho. Além das posições das ordens executadas em seu respectivo dia, em

cada instância também foram incluídos: o número de equipes em serviço no dia, que corresponde ao número m de caixeiros da instância de MTSP; e a posição da garagem da agência de atendimento, que representa o ponto inicial e final das rotas das equipes, correspondendo ao depósito da instância.

Tabela 1. Ordens de serviço executadas no município de Cornélio Procópio de 1º de fevereiro a 23 de fevereiro de 2014. Fonte: COPEL Distribuição.

Dia de trabalho	Equipes	Ordens de serviço
1	1	7
2	1	1
3	3	31
4	4	38
5	4	40
6	4	33
7	4	40
8	1	4
9	1	1
10	3	37
11	4	27
12	4	53
13	4	35
14	4	33
15	2	12
16	0	0
17	2	26
18	4	64
19	4	35
20	4	30
21	4	36
22	2	9
23	1	5

Os atendimentos realizados pelas equipes nos 23 dias está resumido na Tabela 1, que traz a divisão das ordens de acordo com seu dia de execução e o número de equipes em serviço em cada dia. Nos dias 1, 2, 8, 9 e 23 havia apenas uma equipe em serviço ($m = 1$), o que gera instâncias para o TSP e não para o MTSP. Como o foco da nossa metodologia é o MTSP, estes dias de trabalho não foram incluídos nos experimentos. No dia 16 não houve nenhum atendimento, sendo também excluído da base de dados. Dessa forma, os experimentos foram realizados sobre os 17 dias de trabalho restantes.

Para a confecção das matrizes de custos das instâncias dos experimentos 1 e 2 foram utilizadas as distâncias euclidianas, as quais foram calculadas a partir das coordenadas UTM dos pontos das instâncias e após divididas por 1.000, o que gerou valores em quilômetros, já que cada unidade no sistema de projeção UTM corresponde a um metro na superfície da Terra [Machado Jr et al. 2004]. Os valores gerados foram utilizados com a precisão decimal de 3 dígitos. No experimento 3, as matrizes de custos foram preenchidas com as distâncias previstas de deslocamento com veículo, em quilômetros, obtidas de [MapQuest 2015]. Estes custos consideram os percursos das vias reais e foram obtidos

a partir de consultas para todos os pares de pontos que formam as instâncias. Os custos temporais do experimento 4, em segundos, foram obtidas a partir das mesmas consultas e, de acordo com [MapQuest 2015], são calculadas considerando a distância, os limites de velocidade e os sentidos das vias reais.

3.1.2. Protocolo experimental

Nesta experimentação foi desenvolvido um protótipo que implementa os dois algoritmos adaptados para o MTSP. O protótipo foi escrito em C++ e compilado com o pacote de desenvolvimento *Cygwin 1.7.32*. Foi utilizado um computador executando o sistema operacional *Windows 8.1* em um processador de 2,40 GHz e 64 bits com dois núcleos e quatro *threads*; e 8 GB de memória RAM. Os resultados analisados são os valores médios de 100 execuções independentes dos algoritmos para cada configuração. O critério de parada foi fixado em 1.000 ciclos e o tamanho da lista de candidatos é igual a 20 em todas as execuções.

O formato das soluções geradas nesta experimentação é como na descrição geral do MTSP, com único depósito e rotas fechadas. O objetivo foi configurado para minimização da maior rota individual das soluções, conforme a definição do MTSP com objetivo *minmax*. Isto significa que nas comparações entre duas soluções MTSP durante a execução dos algoritmos, a que é classificada como sendo a melhor é aquela cuja maior rota individual possuir um custo menor que a maior rota individual da outra solução.

Em [Dorigo and Stutzle 2004], além da descrição das duas variações dos algoritmos ACO originais que foram adaptadas para o MTSP, também são encontrados os conjuntos de parâmetros que alcançaram os melhores resultados quando da aplicação desses algoritmos sobre um conjunto significativo de instâncias de TSP. Os valores destes parâmetros estão apresentados na Tabela 2 e foram os mesmos utilizados nos experimentos deste trabalho, com exceção do parâmetro β, cujo melhor valor é inferido no experimento 1; e dos parâmetros β e ρ no Experimento 5. Células sem valores na Tabela 2 indicam que o parâmetro não é aplicado ao algoritmo.

3.2. Experimento 1: seleção do parâmetro β

O parâmetro β está presente nas duas variações de algoritmos ACO adaptados para MTSP em estudo, e determina o peso que a visibilidade de uma aresta terá ao serem calculadas as probabilidades que os nós candidatos terão de serem escolhidos por uma formiga em todos os movimentos da colônia durante a execução do algoritmo. A visibilidade corresponde ao inverso do custo da aresta ($1/c_{ij}$). Na Tabela 2 pode ser observado que não é definido um valor único para β, sendo informado que os melhores resultados foram obtidos empiricamente com β assumindo um valor inteiro entre 2 e 5 para instâncias de TSP. Com base nisto, este experimento busca realizar uma seleção inicial do melhor valor para β para os dois algoritmos adaptados para o MTSP.

Neste experimento foi selecionada a maior instância dentre as 17 obtidas a partir dos dados reais: a do dia de trabalho 18, com 64 ordens de serviço e 4 equipes de atendimento. Neste caso, a instância MTSP possui 65 nós (com a inclusão da garagem) e 4 caixeiros. A instância selecionada foi submetida aos dois algoritmos adaptados, com β assumindo valores entre 1 e 5. Os resultados estão representados na Tabela 3. Os valores

Tabela 2. Melhores conjuntos de parâmetros obtidos experimentalmente para os algoritmos ACO originais aplicados a instâncias de TSP. Fonte: [Dorigo and Stutzle 2004]

Parâmetro	RBAS	ACS
τ_0	$0.5w(w-1)/\rho C^{nn}$	$1/nC^{nn}$
N	n	10
α	1	-
β	2 a 5	2 a 5
ρ	0,1	0,1
w	6	-
ξ	-	0,1
q_0	-	0,9

analisados, de acordo com o objetivo do problema, correspondem ao custo da maior rota individual das melhores soluções MTSP construídas ao final das execuções. Os demais parâmetros foram mantidos como os apresentados na Seção 3.1.2.

Tabela 3. Experimento 1: resultados da variação de β no STACS e no STRBAS. Médias de 100 execuções do custo da maior rota individual das melhores soluções de cada execução para uma instância MTSP com 65 nós e 4 caixeiros.

β	STACS				STRBAS				Melhora pelo STRBAS
	melhor	pior	média(m_1)	d. p.	melhor	pior	média(m_2)	d. p.	(m_1-m_2) /m_1x100
1	7,76	8,54	8,04	0,13	7,70	8,15	7,97	0,08	0,84%
2	7,92	8,41	8,14	0,09	7,82	8,16	8,02	0,07	1,47%
3	7,84	8,50	8,15	0,10	7,82	8,20	8,05	0,07	1,22%
4	7,94	8,51	8,18	0,13	7,93	8,19	8,09	0,05	1,13%
5	8,00	8,60	8,27	0,12	8,01	8,22	8,12	0,05	1,76%

Na Tabela 3 pode ser observado que as melhores médias foram obtidas com $\beta = 1$ para os dois algoritmos adaptados. Verifica-se também que, à medida que β cresce, ambos passam a apresentar piores médias. Dessa forma, o valor de β utilizado nos Experimentos 2, 3 e 4 é igual a 1. Os resultados do STACS com $\beta = 2$ foram apresentados em [Barbosa et al. 2015].

3.3. Aplicação do STACS e do STRBAS ao problema real

Nos experimentos 2, 3 e 4 as 17 instâncias de MTSP construídas a partir dos dados reais foram submetidas aos dois algoritmos adaptados, com diferentes matrizes de custos representando as instâncias em cada um dos experimentos. O parâmetro β foi fixado em 1, de acordo com os resultados obtidos pelo Experimento 1, sendo os demais parâmetros mantidos como os apresentados na Seção 3.1.2. Os valores analisados correspondem ao custo da maior rota individual das soluções.

3.3.1. Experimento 2: matrizes de custo com distâncias euclidianas

Neste experimento as instâncias foram representadas por matrizes de custos com as distâncias euclidianas, que foram calculadas a partir das coordenadas reais das ordens de serviço. Os valores analisados na Tabela 4 correspondem ao custo da maior rota individual das melhores soluções MTSP. O parâmetro β é igual a 1 e os demais parâmetros foram mantidos como os apresentados na Seção 3.1.2.

Tabela 4. Experimento 2: resultados do STACS e do STRBAS para as 17 instâncias reais representadas por matrizes de custo com distâncias euclidianas. Médias de 100 execuções do custo da maior rota individual das melhores soluções encontradas em cada execução.

Instância	STACS				STRBAS				Melhora pelo STRBAS
	melhor	pior	média(m_1)	d. p.	melhor	pior	média(m_2)	d. p.	$(m_1\text{-}m_2)/m_1$x100
3	8,22	8,51	8,31	0,03	8,26	8,32	8,30	0,01	0,12%
4	33,89	33,89	33,89	0,00	33,89	33,89	33,89	0,00	0,00%
5	73,29	73,29	73,29	0,00	73,29	73,29	73,29	0,00	0,00%
6	39,33	39,62	39,60	0,05	39,14	39,58	39,47	0,13	0,33%
7	39,81	39,81	39,81	0,00	39,81	39,81	39,81	0,00	0,00%
10	8,26	8,79	8,45	0,17	8,26	8,37	8,32	0,02	1,54%
11	7,65	7,73	7,72	0,02	7,65	7,69	7,65	0,00	0,91%
12	60,80	60,80	60,80	0,00	60,80	60,80	60,80	0,00	0,00%
13	9,24	9,30	9,26	0,02	9,24	9,28	9,24	0,01	0,22%
14	7,86	7,96	7,88	0,01	7,61	7,88	7,86	0,04	0,25%
15	89,94	89,94	89,94	0,00	89,94	89,94	89,94	0,00	0,00%
17	8,66	8,90	8,70	0,06	8,66	8,83	8,72	0,05	-0,23%
18	7,74	8,67	8,05	0,15	7,75	8,13	7,96	0,08	1,12%
19	11,64	11,68	11,64	0,00	11,64	11,64	11,64	0,00	0,00%
20	7,38	7,68	7,40	0,04	7,38	7,40	7,40	0,01	0,00%
21	7,81	8,36	8,09	0,12	7,81	8,15	7,91	0,10	2,22%
22	83,70	83,70	83,70	0,00	83,70	83,70	83,70	0,00	0,00%

3.3.2. Experimento 3: matrizes de custo com distâncias previstas de deslocamento

Neste experimento as instâncias foram representadas por matrizes de custos com as distâncias previstas de deslocamento com veículo, em quilômetros, obtidas da ferramenta de consulta cartográfica [MapQuest 2015]. Os valores analisados na Tabela 5 correspondem ao custo da maior rota individual das melhores soluções MTSP. O parâmetro β é igual a 1 e os demais parâmetros foram mantidos como os apresentados na Seção 3.1.2.

3.3.3. Experimento 4: matrizes de custo com tempos previstos de deslocamento

Neste experimento as instâncias foram representadas por matrizes de custos com os tempos previstos de deslocamento com veículo, em segundos, obtidas da ferramenta de consulta cartográfica [MapQuest 2015]. Os valores analisados na Tabela 6 correspondem ao custo da maior rota individual das melhores soluções MTSP. O parâmetro β é igual a 1 e os demais parâmetros foram mantidos como os apresentados na Seção 3.1.2.

Tabela 5. Experimento 3: resultados do STACS e do STRBAS para as 17 instâncias reais representadas por matrizes de custo com distâncias previstas de deslocamento. Médias de 100 execuções do custo da maior rota individual das melhores soluções encontradas em cada execução.

Instância	STACS				STRBAS				Melhora pelo STRBAS
	melhor	pior	média(m_1)	d. p.	melhor	pior	média(m_2)	d. p.	$(m_1\text{-}m_2)/m_1$x100
3	11,46	12,68	11,91	0,35	11,46	12,53	11,67	0,15	2,02%
4	50,80	53,34	50,83	0,25	50,80	50,96	50,80	0,02	0,06%
5	96,81	99,97	98,87	0,47	96,81	99,06	98,14	0,67	0,74%
6	42,75	45,49	44,23	0,49	42,75	44,50	43,82	0,44	0,93%
7	50,32	53,50	52,49	1,09	50,32	53,20	51,11	0,94	2,63%
10	11,79	12,76	12,21	0,20	11,77	12,29	12,04	0,14	1,39%
11	10,20	11,21	10,46	0,32	10,20	10,50	10,26	0,06	1,91%
12	77,51	79,45	78,35	0,42	77,32	77,69	77,64	0,08	0,91%
13	10,83	11,43	10,95	0,11	10,74	11,16	10,86	0,07	0,82%
14	11,18	12,88	11,68	0,39	11,16	11,87	11,40	0,21	2,40%
15	118,52	118,52	118,52	0,00	118,52	118,52	118,52	0,00	0,00%
17	14,69	15,50	14,95	0,17	14,69	15,39	14,91	0,16	0,27%
18	11,13	12,63	11,88	0,37	11,14	12,69	11,78	0,34	0,84%
19	13,68	15,06	14,20	0,42	13,49	14,67	13,85	0,23	2,46%
20	10,73	11,68	10,84	0,14	10,73	10,96	10,79	0,02	0,46%
21	11,27	12,56	11,70	0,23	11,46	11,54	11,49	0,04	1,79%
22	201,85	202,32	201,92	0,17	201,85	201,85	201,85	0,00	0,03%

Tabela 6. Experimento 4: resultados do STACS e do STRBAS para as 17 instâncias reais representadas por matrizes de custo com tempos previstos de deslocamento. Médias de 100 execuções do custo da maior rota individual das melhores soluções encontradas em cada execução.

Instância	STACS				STRBAS				Melhora pelo STRBAS
	melhor	pior	média(m_1)	d. p.	melhor	pior	média(m_2)	d. p.	$(m_1\text{-}m_2)/m_1$x100
3	1520	1584	1537,76	14,34	1524	1566	1527,09	8,34	0,69%
4	2984	3261	3046,33	41,05	2984	3028	3022,32	10,30	0,79%
5	5569	5981	5717,10	84,72	5569	5726	5656,16	46,86	1,07%
6	2196	2507	2341,84	78,14	2218	2282	2277,95	8,68	2,73%
7	2936	3217	3079,99	56,70	2936	3077	3018,48	31,40	2,00%
10	1504	1594	1549,72	19,92	1503	1591	1535,47	16,75	0,92%
11	1289	1515	1338,65	48,27	1289	1301	1289,66	2,58	3,66%
12	4503	4938	4669,11	106,65	4441	4580	4512,79	35,04	3,35%
13	1316	1437	1361,84	32,84	1318	1384	1339,97	7,11	1,61%
14	1374	1525	1430,54	40,11	1374	1454	1390,51	17,81	2,80%
15	6935	6946	6935,22	1,55	6934	6946	6934,90	1,19	0,00%
17	1706	1907	1749,65	60,71	1706	1864	1723,63	38,27	1,49%
18	1474	1667	1545,91	37,11	1434	1530	1496,39	23,59	3,20%
19	1452	1576	1494,72	26,13	1456	1542	1498,61	23,83	-0,26%
20	1227	1387	1287,71	37,86	1222	1251	1240,90	10,29	3,64%
21	1457	1620	1537,67	45,57	1431	1560	1484,53	12,90	3,46%
22	13034	13034	13034,00	0,00	13034	13034	13034,00	0,00	0,00%

3.3.4. Análise estatística dos resultados do Experimento 4

Para a confirmação da superioridade do STRBAS sobre o STACS é necessário verificar se as soluções criadas pelos algoritmos com uma determinada configuração experimental pertencem à uma mesma distribuição estatística, o que significaria que não há vantagem de um em relação ao outro. Essa verificação pode ser realizada pelo teste de Wilcoxon *Signed-Rank* (WSR) pareado, que é um teste estatístico de hipótese não paramétrico que procura responder se duas amostras distintas representam a mesma população [Derrac et al. 2011]. O teste WSR pareado foi realizado sobre os resultados médios obtidos com a matriz assimétrica de tempos previstos no Experimento 4, apresentados na Tabela 6, cujo experimento resultou nas diferenças mais significativas entre as soluções geradas pelos dois algoritmos para as instâncias reais.

Tabela 7. Cálculo dos postos para realização do teste de Wilcoxon pareado comparando os dois algoritmos adaptados para as 17 instâncias reais representadas por matrizes de custo com tempos previstos de deslocamento.

Instância	STACS (X)	STRBAS (Y)	$X - Y$	$\|X - Y\|$	Postos	Negativos	Positivos
3	1537,76	1527,09	10,67	10,67	3		3
4	3046,33	3022,32	24,01	24,01	6		6
5	5717,10	5656,16	60,94	60,94	13		13
6	2341,84	2277,95	63,89	63,89	15		15
7	3079,99	3018,48	61,51	61,51	14		14
10	1549,72	1535,47	14,25	14,25	4		4
11	1338,65	1289,66	48,99	48,99	10		10
12	4669,11	4512,79	156,32	156,32	16		16
13	1361,84	1339,97	21,87	21,87	5		5
14	1430,54	1390,51	40,03	40,03	8		8
15	6935,22	6934,90	0,32	0,32	1		1
17	1749,65	1723,63	26,02	26,02	7		7
18	1545,91	1496,39	49,52	49,52	11		11
19	1494,72	1498,61	-3,89	3,89	2	-2	
20	1287,71	1240,90	46,81	46,81	9		9
21	1537,67	1484,53	53,14	53,14	12		12
22	13034,00	13034,00	0,00	0,00	-		

Como pode ser observado na Tabela 7, os resultados médios de 100 execuções do STACS e do STRBAS foram transcritos e denominados como X e Y, respectivamente. Na próxima coluna da Tabela 7 é calculada a diferença das médias para as 17 instâncias. Em seguida é apresentado o valor absoluto das diferenças. Na coluna **Posto** é definida uma ordem para as diferenças absolutas, iniciando a numeração em 1 a partir da menor diferença. Não são atribuídos postos para diferenças nulas, como a da instância 22, de forma que as amostras passam a conter 16 elementos. Nas duas colunas restantes da Tabela 7 os postos são separados de acordo com os sinais originais das diferenças.

Sendo W^- definido como a soma dos postos negativos e W^+ como a soma dos postos positivos, então $W^- = -2$ e $W^+ = 134$. No teste WSR pareado, caso o menor valor absoluto W dentre estes dois valores for menor ou igual a um valor critico W_{crit}, deve-se rejeitar a hipótese nula H_0, que considera que as duas amostras são iguais [Derrac et al. 2011]. Dessa forma, nesta comparação, $W = 2$. Os valores críticos de W para amostras com 16 elementos estão representados na Tabela 8.

Tabela 8. Valores críticos para W para amostras com 16 elementos de acordo com o nível de significância α . Fonte: [Zar 2007]

α	0,5	0,2	0,1	0,05	0,02	0,01	0,005	0,001
W_{crit}	54	42	35	29	23	19	15	8

Tendo sido obtido $W = 2$ a partir da Tabela 7, e sendo $W_{crit} = 8$ numa comparação com nível de significância α igual a $0,001$ (Tabela 8) pode-se afirmar que as amostras representam diferentes distribuições estatísticas até este nível de significância, pois $W < W_{crit}$. Dessa forma, como os resultados obtidos pelo STRBAS considerando os valores médios foram melhores que o STACS, pode-se concluir que STRBAS foi mais eficiente que o STACS neste experimento.

Uma forma mais eficiente da realização do teste WSR pareado para comparação entre dois algoritmos é a utilização de softwares de análise estatística, como o R [R Core Team 2015]. O comando *wilcox.test* do pacote *Stats* do R retorna o valor W do teste WSR pareado a partir de duas amostras de entrada. Além de W, este comando também retorna o *p-value*, que corresponde à probabilidade de H_0 ser verdadeira. O teste foi realizado com o auxilio do R para as duas amostras da Tabela 7 configurando H_0 como "o STACS foi melhor que o STRBAS", que é melhor definida como: "a amostra com os resultados obtidos pelo STRBAS pertencem a uma distribuição diferente da amostra dos resultados do STACS e são valores maiores (soluções piores)". Rejeitando-se H_0, pode-se concluir que o segundo algoritmo gerou melhores soluções que o primeiro. Os valores retornados foram $W = 2$ (de acordo com os valores obtidos a partir da Tabela 7) e *p-value* = $0,0003534$, confirmando que H_0 pode ser rejeitada até um nível de significância α inferior a $0,001$, conforme constatado no parágrafo anterior.

3.3.5. Tempos computacionais

Na Tabela 9 estão apresentados os tempos de execução, em segundos, calculados a partir da média das 100 execuções do STACS e do STRBAS nos Experimentos 2, 3 e 4.

3.3.6. Comparação com as rotas reais

A partir do histórico da agência de atendimento foram obtidas as coordenadas de todas as ordens executadas, que possibilitaram a construção de instâncias de MTSP representando os 17 dias de trabalho. Analisando o horário de execução das ordens disponível no histórico, também foi possível definir a sequência real dos atendimentos nos dias de trabalho, cujos custos das rotas calculados pelo protótipo com a utilização das matrizes de custos aplicadas nos experimentos servem de referência para a avaliação dos resultados obtidos pela nossa metodologia. Para realizar esta comparação, na Tabela 10 os custos das maiores rotas das equipes reais são comparados com a média das maiores rotas individuais obtidas pelo STRBAS, algoritmo que obteve melhor desempenho nos experimentos realizados.

Tabela 9. Tempos de execução do STACS e do STRBAS para as 17 instâncias reais. Médias dos tempos das execuções dos Experimentos 2, 3 e 4.

Instância	Equipes	Ordens	Tempo médio de execução (segundos)	
			STACS	**STRBAS**
3	3	31	1,68	5,91
4	4	38	3,36	13,60
5	4	40	3,08	13,29
6	4	33	2,47	7,82
7	4	40	3,22	13,57
10	3	37	2,43	12,95
11	4	27	1,54	4,23
12	4	53	7,28	33,62
13	4	35	2,62	8,53
14	4	33	2,51	7,85
15	2	12	0,08	0,08
17	2	26	1,67	3,82
18	4	64	13,47	81,76
19	4	35	2,69	9,33
20	4	30	1,67	5,41
21	4	36	2,67	9,25
22	2	9	0,08	0,17

Tabela 10. Comparação entre as rotas reais e a média das soluções de 100 execuções do STRBAS. Os valores correspondem ao custo da maior rota individual das soluções.

Dia	Solução Real			STRBAS			Melhoramento		
	euc.(e_r)	dist.(d_r)	tempo(t_r)	euc.(e_s)	dist.(d_s)	tempo(t_s)	(e_r-e_s)/e_rx100	(d_r-d_s)/d_rx100	(t_r-t_s)/t_rx100
3	22,37	32,14	4088	8,30	11,67	1527	62,89%	63,69%	62,64%
4	41,39	62,48	4188	33,89	50,80	3022	18,12%	18,69%	27,83%
5	84,36	112,55	7631	73,29	98,14	5656	13,12%	12,81%	25,88%
6	54,24	65,37	4859	39,47	43,82	2278	27,23%	32,96%	53,12%
7	51,15	70,80	5162	39,81	51,11	3018	22,17%	27,81%	41,52%
10	23,47	35,66	3913	8,32	12,04	1535	64,55%	66,24%	60,76%
11	18,61	25,97	3635	7,65	10,26	1290	58,90%	60,50%	64,52%
12	74,20	97,01	7420	60,80	77,64	4513	18,06%	19,97%	39,18%
13	23,05	35,31	4085	9,24	10,86	1340	59,91%	69,25%	67,20%
14	23,78	35,19	4023	7,86	11,40	1391	66,95%	67,61%	65,44%
15	133,77	169,97	10237	89,94	118,52	6935	32,76%	30,27%	32,26%
17	20,22	30,71	3743	8,72	14,91	1724	56,86%	51,45%	53,95%
18	24,05	35,43	4404	7,96	11,78	1496	66,91%	66,75%	66,02%
19	21,65	32,60	3213	11,64	13,85	1499	46,24%	57,51%	53,36%
20	23,83	40,97	3939	7,40	10,79	1241	68,94%	73,66%	68,50%
21	28,70	43,28	5009	7,91	11,49	1485	72,44%	73,45%	70,36%
22	108,06	251,60	15804	83,70	201,85	13034	22,54%	19,77%	17,53%
				Melhoramento médio:			**34,88%**	**35,35%**	**44,43%**

3.4. Experimento 5: comparação entre o TACO, o STACS e o STRBAS utilizando instâncias da TSPLIB

A TSPLIB é uma biblioteca digital utilizada por pesquisadores que disponibiliza várias instâncias de TSP e de problemas relacionados [Reinelt 1991]. Para verificar a eficiência de seu algoritmo TACO, que foi a base utilizada para desenvolvimento da algoritmo ACO para MTSP apresentado neste artigo (Algoritmo 1), [Vallivaara 2008] aplicou-o sobre instâncias disponíveis na TSPLIB. Neste experimento, o STACS e o STRBAS foram aplicados sobre três destas instâncias, Eil51, Eil76 e Eil101, em configurações com 2, 3 e 4 caixeiros.

Os resultados do STACS e do STRBAS foram obtidos obedecendo o protocolo descrito na Seção 3.1.2, exceto: os resultados são referentes a 10 execuções, como os provenientes do TACO; para as instâncias Eil51 e Eil76: $\rho = 0,01$; para a Eil101: $\rho = 0,02$ e $\beta = 3$. Estes novos valores para os parâmetros foram os que obtiveram os melhores resultados para as respectivas instâncias numa seleção inicial de parâmetros. Além da busca local 2-opt, que é realizada sobre todas as soluções criadas, neste experimento também foi aplicada a busca local 3-opt, mas apenas quando a melhor solução da execução é atualizada, devido ao tempo computacional exigido.

Na Tabela 11 os resultados obtidos pelo STACS obedecendo ao protocolo experimental descrito são comparados aos obtidos pelo TACO, que foram extraídos de [Vallivaara 2008]. Para a instância Eil51 o STACS conseguiu encontrar as melhores soluções equivalentes aos do TACO com 2 e 4 caixeiros. Porém, quando considerando a média das 10 execuções, o TACO foi superior em todas as configurações. Aplicando o teste estatístico WSR pareado detalhado na Seção 3.3.4 aos valores médios obtidos, e definindo H_0 como "o TACO foi melhor que o STACS", obteve-se $W = 45$ e *p-value* = $0,998047$, indicando que H_0 deve ser aceita e confirmando a superioridade do TACO em relação ao STACS nesta configuração experimental.

Tabela 11. Resultados do TACO e do STACS utilizando instâncias da TSPLIB. Médias de 10 execuções do custo da maior rota individual das melhores soluções encontradas em cada execução.

Instância	n	m	TACO		STACS		Melhora pelo STACS
			melhor	média(m_1)	melhor	média(m_2)	(m_1-m_2)/m_1x100
Eil51	51	2	224	224,70	224	231,80	-3,16%
		3	159	163,00	162	169,10	-3,74%
		4	130	131,60	130	134,70	-2,36%
Eil76	76	2	278	281,00	290	297,00	-5,69%
		3	194	199,10	201	209,50	-5,22%
		4	161	163,60	166	168,20	-2,81%
Eil101	101	2	327	330,30	332	342,40	-3,66%
		3	226	227,80	238	244,40	-7,29%
		4	178	181,00	185	192,10	-6,13%

Na Tabela 12 os resultados obtidos pelos STRBAS são comparados aos do TACO, onde pode ser notado que o STRBAS produziu resultados equivalentes ao TACO para as duas instâncias menores, quando analisando as médias produzidas. Inclusive para duas configurações o STRBAS foi capaz de encontrar melhores soluções, que estão destacadas

em negrito na Tabela 12. Para a instância com 101 nós os resultados do TACO são melhores, mas uma inferência detalhada de parâmetros ainda é necessária para verificar se as configurações utilizadas são as que resultam na melhor performance do STRBAS.

Aplicando o WSR pareado aos valores médios da Tabela 12, e definindo H_0 como "o TACO foi melhor que o STRBAS", obteve-se $W = 35$ e *p-value* = $0,9384$, indicando que H_0 deve ser aceita, o que significa que o TACO foi realmente superior ao STRBAS nas amostras analisadas.

Tabela 12. Resultados do TACO e do STRBAS utilizando instâncias da TSPLIB. Médias de 10 execuções do custo da maior rota individual das melhores soluções encontradas em cada execução.

Instância	n	m	TACO		STRBAS		Melhora pelo STRBAS
			melhor	média(m_1)	melhor	média(m_2)	(m_1-m_2)/m_1x100
Eil51	51	2	224	224,70	**223**	225,40	-0,31%
		3	159	163,00	159	161,10	1,17%
		4	130	131,60	130	132,30	-0,53%
Eil76	76	2	278	281,00	280	285,50	-1,60%
		3	194	199,10	194	198,50	0,30%
		4	161	163,60	**160**	162,70	0,55%
Eil101	101	2	327	330,30	333	336,90	-2,00%
		3	226	227,80	231	237,30	-4,17%
		4	178	181,00	183	185,20	-2,32%

Na Tabela 13 os resultados obtidos pelo STACS são comparados aos obtidos pelo STRBAS, onde percebe-se que que o segundo se saiu melhor que o primeiro considerando os valores médios das soluções. Configurando H_0 como "o STACS foi melhor que o STRBAS" no teste de WSR pareado, os valores retornados foram $W = 0$ (o STRBAS foi superior em todas as comparações das amostras) e *p-value* = $0,004545$, confirmando que H_0 pode ser rejeitada até um nível de significância α inferior a $0,005$, confirmando a superioridade do STRBAS nesta configuração experimental.

Tabela 13. Resultados do STACS e do STRBAS utilizando instâncias da TSPLIB. Médias de 10 execuções do custo da maior rota individual das melhores soluções encontradas em cada execução.

Instância	n	m	STACS		STRBAS		Melhora pelo STRBAS
			melhor	média(m_1)	melhor	média(m_2)	(m_1-m_2)/m_1x100
Eil51	51	2	224	231,80	223	225,40	2,76%
		3	162	169,10	159	161,10	4,73%
		4	130	134,70	130	132,30	1,78%
Eil76	76	2	290	297,00	280	285,50	3,87%
		3	201	209,50	194	198,50	5,25%
		4	166	168,20	160	162,70	3,27%
Eil101	101	2	332	342,40	333	336,90	1,61%
		3	238	244,40	231	237,30	2,91%
		4	185	192,10	183	185,20	3,59%

3.5. Análise dos Resultados

Os resultados do Experimento 1, na Seção 3.2, apontam que o STACS e o STRBAS são mais eficientes, para a instância utilizada, com o parâmetro β igual a 1. Esta conclusão sugere que a seleção deve se estender aos demais parâmetros, a fim da determinação da melhor configuração experimental. O protótipo desenvolvido se apresenta como uma ferramenta útil para realização desta seleção na continuação deste trabalho.

A partir dos resultados do Experimento 2, apresentado na Seção 3.3.1, é possível verificar que o novo algoritmo adaptado, o STRBAS, foi mais eficiente, em média, que o STACS na construção de soluções otimizadas para 16 das 17 instâncias reais. A superioridade do STRBAS também pode ser observada nos resultados do Experimento 3, no qual foi melhor em todas as instâncias; e no Experimento 4, no qual novamente foi inferior em apenas uma das instâncias. O teste estatístico WSR pareado realizado sobre os resultados do Experimento 4 também confirmou a superioridade do STRBAS, mas para a generalização desta constatação, além de uma seleção de parâmetros detalhada, é necessária a aplicação dos algoritmos sobre um conjunto maior de instâncias de MTSP.

Os Experimentos 3 e 4 também comprovam que a nossa metodologia pode ser aplicada a custos que representem de forma mais fiel as instâncias do problema, inclusive com matrizes de custos assimétricas. Como toda a informação que quantifica a instância fica armazenada na matriz de custos, bastou a substituição dos custos dessa matriz para gerar uma nova representação das mesmas instâncias do problema, que foram corretamente otimizadas pelos algoritmos adaptados, como apresentado nas Tabelas 5 e 6.

Os tempos computacionais da Tabela 9 apresentam uma desvantagem do STRBAS em relação ao STACS. Como no STRBAS foram geradas n solução por ciclo, enquanto no STACS foram geradas 10 soluções por ciclo, o tempo gasto pelo STRBAS é significativamente maior para execução de um ciclo, para instâncias com $n > 10$. A atualização de feromônio do STRBAS também consome mais tempo, pois utiliza as $w = 6$ soluções na atualização, enquanto o STACS utiliza apenas a melhor solução encontrada até o momento na execução. Para a comparação dos algoritmos adaptados considerando prioritariamente o tempo computacional, uma nova experimentação deve ser realizada, definindo um critério de parada temporal para as execuções.

Quanto à eficácia da nossa metodologia quando aplicada ao problema real, pode ser verificado na Tabela 10 que o STRBAS gerou soluções com rotas individuais de 34,88% a 44,43% melhores que as rotas reais das equipes. Apesar de existirem fatores relevantes do problema real que ainda devem ser tratados pela metodologia, como por exemplo o surgimento de novas ordens de serviço durante o dia de trabalho das equipes, estes resultados iniciais comprovam que a ACO diminuiu consideravelmente os custos das rotas reais percorridas pelas equipes de atendimento nas instâncias reais experimentadas.

O objetivo do Experimento 5 foi verificar o desempenho do STACS e do STRBAS quando aplicados à instâncias da TSPLIB. Utilizando os resultados obtidos pelo TACO como referência, apresentados em [Vallivaara 2008], foi possível constatar que tanto o STACS quanto o STRBAS tiveram desempenho inferior ao do TACO para as instâncias e protocolo experimental aplicados. Entretanto, estes resultados são preliminares, pois o refinamento dos valores dos parâmetros dos algoritmos pode aumentar sua eficiência, sendo esta uma das hipóteses a serem testadas na continuação desta pesquisa.

4. Conclusões

Este trabalho comprova que a programação do atendimento comercial nas empresas de distribuição de energia elétrica pode ser realizada de forma mais eficiente com o auxílio de uma ferramenta computacional. A nossa metodologia, que trata o problema como instâncias de MTSP e aplica a ACO para otimizá-las, mostrou-se eficaz na criação de soluções para distribuição das ordens de serviço e roteamento das equipes, ao reduzir significativamente o custo da maior rota individual nos experimentos realizados e, consequentemente, os gastos operacionais da empresa.

Na primeira versão deste artigo ([Barbosa et al. 2015]), os experimentos foram realizados sobre as instâncias reais representadas por custos euclidianos submetidas ao STACS, que foi o primeiro algoritmo ACO adaptado para o MTSP desenvolvido na nossa metodologia. Nestes experimentos foram aplicados os mesmos valores dos parâmetros sugeridos na literatura para o TSP, e dois objetivos foram utilizados: a minimização do custo total e a minimização do custo da maior rota individual das soluções. Analisando os resultados obtidos, foi constatado que a minimização do custo total é menos adequada ao problema em estudo, devido à criação de rotas desiquilibradas para as equipes, não sendo, portanto, aplicada nos novos experimentos.

Nesta versão estendida verificou-se que o novo algoritmo adaptado, o STRBAS, é ainda mais eficiente que o STACS para otimização das instâncias reais, quando definindo um critério de parada em número de ciclos na comparação. O resultado obtido com a seleção do parâmetro β mostra que a adaptação da ACO para MTSP, da forma que foi feita na nossa metodologia, se mostra mais eficiente à medida que o valor de β diminui. Esta constatação sugere que resultados ainda melhores podem ser obtidos com a inferência dos demais parâmetros para os algoritmos ACO adaptados.

A partir dos experimentos desta versão estendida, também foi possível comprovar a versatilidade da nossa metodologia, através da utilização de custos mais representativos do ambiente real para otimização das instâncias: as previsões de distância e tempo de deslocamento obtidas em [MapQuest 2015]. Esta característica expande a aplicabilidade da nossa metodologia à problemas que possam ser modelados como instâncias assimétricas de MTSP, mais representativas do mundo real. Dessa forma, em relação às outras soluções aplicadas ao problema real na literatura, a nossa metodologia se destaca em dois aspectos principais: a distribuição das ordens e as rotas das equipes são obtidas a partir de uma única solução de MTSP; e a representação do ambiente pode ser alterada apenas com a substituição da matriz de custos, onde fica concentrada a representação das instâncias.

O teste WSR Pareado realizado nos Experimentos 4 e 5 mostrou-se como uma ferramenta eficiente para a comparação entre os resultados obtidos. Foi possível constatar através deste teste que tanto o STACS quanto o STRBAS produziram resultados médios inferiores que os do TACO para as instâncias modelo; e que o STRBAS se mostrou melhor que STACS tanto com as instâncias reais quanto com as instâncias modelo.

A continuação deste trabalho prevê a evolução da metodologia desenvolvida para tratamento do caráter dinâmico do problema em estudo, onde novas ordens de serviço surgem aleatoriamente, tanto geográfica quanto temporalmente, durante o dia de trabalho das equipes. Esta evolução também será capaz de lidar com as ordens emergenciais, que devem ser atendidas o mais rápido possível.

Referências

Ahmadi, S. and Osman, I. H. (2005). Greedy random adaptive memory programming search for the capacitated clustering problem. *European Journal of Operational Research*, 162(1):30–44.

Amorim, M. L. F. (2010). Otimização de atendimentos de emergência em redes de distribuição de energia elétrica. Mestrado em Computação Científica e Sistemas de Potência, Universidade Federal Fluminense, Niterói.

Arya, V., Goyal, A., and Jaiswal, V. (2014). An optimal solution to multiple travelling salesperson problem using modified genetic algorithm. *International Journal of Application or Innovation in Engineering & Management*, 3(1):425–430.

Augustine, J. (2002). *Offline and online variants of the traveling salesman problem*. PhD thesis, Louisiana State University, Department of Electrical and Computer Engineering.

Barbosa, D. F., Silla Jr, C. N., and Kashiwabara, A. Y. (2015). Aplicação da otimização por colônia de formigas ao problema de múltiplos caixeiros viajantes no atendimento de ordens de serviço nas empresas de distribuição de energia elétrica. In *Anais do XI Simpósio Brasileiro de Sistemas de Informação*, pages 23–30, Goiânia. Instituto de Informática da Universidade Federal de Goiás.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219.

Brown, E. C., Ragsdale, C. T., and Carter, A. E. (2007). A grouping genetic algorithm for the multiple traveling salesperson problem. *International Journal of Information Technology & Decision Making*, 06(02):333–347.

Bullnheimer, B., Hart, R. F., and Straub, C. (1999). A New Rank-Based Version of the Ant System: A Computational Study. *Central European Journal of Operations Research and Economics*, 7(1):25 – 38.

Carter, A. E. and Ragsdale, C. T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research*, 175(1):246–257.

Christofides, N. (1976). The vehicle routing problem. *Revue française d automatique, d informatique et de recherche opérationnelle*, 10(2):55–70.

Costa, C. E. S., Costa, D. M. B., and Goes, A. R. T. (2006). Determinação de setores de atendimento em uma concessionária de energia. In *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, pages 951–962, Goiânia. Universidade Católica de Goiás.

Costa, Y. J., Abreu, R., Coello, N. I., and Nowé, A. (2012). Multi-type ant colony system for solving the multiple traveling salesman problem. *Revista Técnica de la Facultad de Ingeniería Universidad del Zulia*, 35(3):311–320.

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Dorigo, M. and Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics*, 26(1):29–41.

Dorigo, M. and Stutzle, T. (2004). *Ant Colony Optimization*. The MIT Press, Cambridge.

Garcia, V. J., Bassi, O. A., Dhein, G., Bernardon, D. P., and Cardoso Jr, G. C. (2012). Problema de roteamento de veículos para atendimento de ordens emergenciais em concessionária de distribuição de energia elétrica. In *Anais do Simpósio Brasileiro de Pesquisa Operacional*, pages 1222–1231, Rio de Janeiro. Congresso Latino-Iberoamericano de Investigación Operativa.

Garcia, V. J., Bernardon, D., Sperandio, M., do Vale, C., and Fernandes, J. (2010a). Service order dispatching in electric utilities. In *Proceedings of 45th International Universities Power Engineering Conference*, pages 1 – 7, Cardiff. Institute of Electrical and Electronics Engineers (IEEE).

Garcia, V. J., Bernardon, D. P., Sperandio, M., Loser, G., Vale, C., and Fernandes, J. (2010b). Gestão estratégica das ordens de serviço: uma abordagem para despacho centralizado. In *Anais do XIX Seminário Nacional de Distribuição de Energia Elétrica*, pages 1–8, São Paulo. Associação Brasileira de Engenheiros Eletricistas.

Ghafurian, S. and Javadian, N. (2011). An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. *Applied Soft Computing*, 11(1):1256–1262.

Goel, A. and Meisel, F. (2013). Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research*, 231(1):210–228.

Gomes, F. R. A., Prata, B. A., Barroso, G. C., and Arruda, J. a. B. F. (2008). Uma heurística para o problema de despacho dinâmico de equipes aplicada na logística de resposta emergencial de uma concessionária de energia elétrica. In *XXVIII Encontro Nacional de Engenharia de Produção*, Rio de Janeiro. Associação Brasileira de Engenharia de Produção.

IPARDES (2014). Caderno estatístico do município de Cornélio Procópio. Disponível em: http://www.ipardes.gov.br/cadernos/montacadpdf1.php?municipio=86300. Acesso em: 28 de dezembro de 2014.

Junjie, P. and Dingwei, W. (2006). An ant colony optimization algorithm for multiple travelling salesman problem. In *Proceedings of First International Conference on Information and Control*, Beijing. Institute of Electrical and Electronics Engineers (IEEE).

Kanda, J. Y. (2014). Sistema de meta-aprendizado para a seleção de meta-heurísticas para o problema do caixeiro viajante. In *Anais do X Simpósio Brasileiro de Sistemas de Informação*, pages 651–662, Londrina. Universidade Estadual de Londrina.

Liu, W., Li, S., and Zhao, F. (2009a). An ant colony optimization algorithm for the multiple traveling salesmen problem. In *Proceedings of 4th IEEE Conference on In-*

dustrial Electronics and Applications, pages 1533–1537, Xi'an. Institute of Electrical and Electronics Engineers (IEEE).

Liu, W., Li, S., Zheng, A., and Zhao, F. (2009b). Tobacco distribution vehicle routing program and the resolving method. In *Proceedings of 2009 Chinese Control and Decision Conference*, pages 5172–5175, Guilin. Institute of Electrical and Electronics Engineers (IEEE).

Machado Jr, D. M., Dal Santo, M. A., and Loch, C. (2004). Considerações acerca de trabalhos em áreas de divisa de fusos UTM. In *Anais do Congresso Brasileiro de Cadastro Técnico Multifinalitário*, pages 1–14, Florianópolis. Universidade Federal de Santa Catarina.

MapQuest (2015). Mapquest maps - driving directions. Disponível em: http://www.mapquest.com. Acesso em: 25 de julho de 2015.

Nahuis, F. V. C. (2013). Automação do despacho dinâmico de viaturas para o atendimento das ordens de serviço nas redes de distribuição de energia elétrica. Mestrado em engenharia elétrica, Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira.

Okonjo-Adigwe, C. (1988). An effective method of balancing the workload amongst salesmen. *Omega*, 16(2):159–163.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Reinelt, G. (1991). Tsplib - a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384.

Somhom, S., Modares, A., and Enkawa, T. (1999). Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers & Operations Research*, 26(4):395–407.

Steiner, M. T. A., Costa, C. E. S., Costa, D. M. B., Filho, E. A., and Zambenedetti, V. C. (2006). Técnicas da pesquisa operacional aplicadas à logística de atendimento aos usuários de uma rede de distribuição de energia elétrica. *Revista Eletrônica Sistemas & Gestão*, 1(3):229–243.

Tang, L., Liu, J., Rong, A., and Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*, 124(2):267–282.

Vallivaara, I. (2008). A team ant colony optimization algorithm for the multiple travelling salesmen problem with minmax objective. In *Proceedings of the 27th International Conference on Modelling, Identification and Control*, pages 387–392, Anaheim. ACTA Press.

Vasagam, B. M. (2012). Ant colony algorithm and genetic algorithm for multiple travelling salesmen problem. Master's thesis, University of Salford, Manchester.

Venkatesh, P. and Singh, A. (2015). Two metaheuristic approaches for the multiple traveling salesperson problem. *Applied Soft Computing*, 26:74–89.

Verboski, T. A. (2010). Proposta e simulação de um algoritmo de designação otimizado para despacho de equipes de atendimento de uma empresa de energia elétrica. Mestrado em métodos numéricos de engenharia, Universidade Federal do Paraná, Curitiba.

Volpi, N. M. P., Wilhelm, V. E., Steiner, M. T. A., Yuan, J. Y., and Matioli, L. C. (2008). Definição do Número, do Turno e da Localização de Equipes de Atendimento aos Usuários de uma Rede de Distribuição de Energia Elétrica via Simulação. In *Anais do XL Simposio Brasileiro de Pesquisa Operacional*, pages 344–354, João Pessoa. Universidade Federal da Paraíba.

Wang, X., Liu, D., and Hou, M. (2013). A novel method for multiple depot and open paths, multiple traveling salesmen problem. In *Proceedings of 11th IEEE International Symposium on Applied Machine Intelligence and Informatics*, pages 187–192, Herl'any. Institute of Electrical and Electronics Engineers (IEEE).

Wang, X., Wang, S., Bi, D., and Ding, L. (2007). Hierarchical wireless multimedia sensor networks for collaborative hybrid semi-supervised classifier learning. *Sensors*, 7(11):2693–2722.

Yousefikhoshbakht, M., Didehvar, F., and Rahmati, F. (2013). Modification of the ant colony optimization for solving the multiple traveling salesman problem. *Romanian Journal of Information Science And Technology*, 16(1):65–80.

YousefiKhoshbakht, M. and Sedighpour, M. (2012). A combination of sweep algorithm and elite ant colony optimization for solving the multiple traveling salesman problem. In *Proceedings of the Romanian Academy*, volume 13, pages 295–301, Bucareste. The Publishing House of the Romanian Academy.

Yuan, S., Skinner, B., Huang, S., and Liu, D. (2013). A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research*, (1):72–82.

Zar, J. H. (2007). *Biostatistical Analysis (5th Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Zhou, W. and Yao, P. (2013). An improved ant colony optimization algorithm for multiple traveling salesman problem. *Advances in information Sciences and Service Sciences*, 5(May):637–644.

Zhu, A. and Yang, S. (2003). An improved self-organizing map approach to traveling salesman problem. In *Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, number October, Changsha. Institute of Electrical and Electronics Engineers (IEEE).

Zografos, K. G., Douligeris, C., and Tsoumpas, P. (1998). An integrated framework for managing emergency-response logistics: The case of the electric utility companies. *IEEE Transactions on Engineering Management*, 45(2):115–126.