CARLOS EDUARDO DI GIACOMO ARAÚJO

ALGORITMOS GENÉTICOS HÍBRIDOS SEM DELIMITADORES DE ROTAS PARA PROBLEMAS DE ROTEIRIZAÇÃO DE VEÍCULOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

São Paulo

2008

CARLOS EDUARDO DI GIACOMO ARAÚJO

ALGORITMOS GENÉTICOS HÍBRIDOS SEM DELIMITADORES DE ROTAS PARA PROBLEMAS DE ROTEIRIZAÇÃO DE VEÍCULOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

Área de Concentração:
Engenharia de Sistemas Logísticos

Orientador:
Prof. Dr. Cláudio Barbieri da Cunha

São Paulo
2008

Este trabalho foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 16 de janeiro de 2008.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Araújo, Carlos Eduardo Di Giacomo
Algoritmos genéticos híbridos sem delimitadores de rotas para problemas de roteirização de veículos / C.E.Di G. Araújo. -- ed.rev. -- São Paulo, 2007.
89 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Transportes. Sistemas Logísticos.

1.Heurística 2.Algoritmos genéticos 3.Roteirização de veículos I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Transportes II.t.

A minha esposa Thais
e ao colega Alexandre Góes.

AGRADECIMENTOS

Em primeiro lugar, ao professor Cláudio Barbieri da Cunha, que além de ter em muito ajudado no desenvolvimento desta dissertação com seu conhecimento, sempre que precisei me incentivou e mostrou o caminho a ser seguido.

Aos professores Marco Brinati e Miguel Cezar Santoro pelos conselhos dados no exame de qualificação.

Aos professores Nicolau Gualda e Rui Botter pelos conhecimentos transmitidos em sala de aula.

Ao professor Hugo Yoshizaki por ter despertado em mim o gosto pela logística.

Aos meus pais e meu irmão pelo apoio que sempre me deram em minhas atividades acadêmicas e ao meu amigo Guilherme Maciel pelo importante auxílio prestado.

À Thais, por sua compreensão e apoio durante todo esse tempo em que teve que dividir minha atenção com o desenvolvimento deste trabalho.

RESUMO

Apesar de serem utilizados com sucesso em problemas de roteirização clássicos como o do caixeiro-viajante e o de roteirização de veículos com janelas de tempo, os algoritmos genéticos não apresentavam bons resultados nos problemas de roteirização de veículos sem janelas de tempo.

Utilizando-se de uma tendência recente de hibridização de algoritmos genéticos, Prins (2004) elaborou um algoritmo para o problema de roteirização de veículos sem janelas de tempo, monoperíodo, e que obrigatoriamente atenda a todos os clientes cujos resultados, quando aplicado a instâncias de Christofides et al. (1979) e de Golden et al. (1998), são comparáveis aos melhores códigos elaborados com base na busca tabu.

Diferentemente da maioria dos algoritmos genéticos apresentados para solução de problemas de roteirização de veículos, no método desenvolvido por Prins (2004) o cromossomo é composto apenas pelos pontos a serem atendidos, não contendo delimitadores de rotas. Estas são definidas a partir de um método de particionamento do cromossomo.

Este trabalho implementa o algoritmo descrito por Prins (2004) e propõe a este melhorias em diversas de suas etapas, como inicialização, operação de *crossover*, operação de mutação, reinicialização e particionamento do cromossomo. As alterações implantadas são aplicadas às instâncias de Christofides et al. (1979) e comparadas com o algoritmo inicial em termos de qualidade de solução e tempo de processamento. Finalmente, é elaborado um algoritmo genético que contempla as alterações que obtiveram resultados positivos.

ABSTRACT

In the Vehicle Routing Problem (VRP) we seek for a set of minimum-cost vehicle routes for a fleet of identical vehicles, each starting and ending at a depot, such that each customer is visited exactly once and the total demand of any route does not exceed the vehicle capacity. Several families of heuristics have been proposed for the VRP. They can be broadly classified into two main classes: *classical* heuristics developed between 1960 and 1990, and, more recently, metaheuristics. Among them, tabu search plays an key role, being acknowledged by most authors as the most successful approach for the VRP. In the literature some successful implementations of metaheuristic Genetic Algorithm (GA) can be found for classic routing problems such as the traveling salesman and vehicle routing problems with time windows. However, until recently, the same did not apply for the VRP.

In this thesis we develop a genetic algorithm without trip delimiters, and hybridized with a local search procedure, for the solving the VRP, which is based on the work of Prins (2004). At any time, a chromosome can be converted into an optimal VRP solution (subject to chromosome sequence) by means of a splitting procedure, in which the chromosome sequence, representing a giant tour, is partitioned into feasible routes in terms of vehicle capacities.

Starting with the procedure originally proposed Prins (2004), we then introduce new improvements in terms of the different components of the GA, aiming to obtain improved solutions. These include how we determine the initial population, different partitioning approaches, alternative reproduction (crossover) processes, a granular tabu mechanism similar to the one proposed by Toth and Vigo (2003 and, finally, in changes in the reinitialization process, aiming to reestablish diversity.

Computational experiments are presented, based on the 14 classical Christofides instances for the VRP. The results show that the proposed improved versions of the GA allow us to obtain better solutions when compared to the original approach by Prins (2004).

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

1 INTRODUÇÃO

1.1 Apresentação

Este trabalho trata o problema de roteirização de veículos sem janelas de tempo e com frota uniforme e infinita.

A partir de uma relação de locais que devem ser atendidos, desde uma base onde está localizada a frota, a solução do problema consiste em definir o roteiro de cada veículo, tendo como objetivo obter o custo total mínimo das rotas. Os veículos têm restrições de capacidade e de duração total da rota e pode ou não haver tempo de serviço no ponto a ser atendido. Todos os pontos devem ser atendidos.

Nesta dissertação é proposta e implementada uma estratégia de solução baseada em algoritmos genéticos para a solução de problemas de roteirização de veículos, com base na implementação apresentada por Prins (2004), que tem como principal característica o cromossomo sem delimitadores de rotas.

Buscou-se também desenvolver versões aprimoradas do algoritmo de Prins (2004), resultantes de alterações nas seguintes etapas: geração de soluções iniciais, *crossover*, busca local, particionamento do cromossomo para geração das rotas e reinicialização.

Os algoritmos propostos são aplicados às instâncias de Christofides et al. (1979) e os resultados – em termos de qualidade da solução e custo computacional – são comparados com o algoritmo de Prins (2004) aqui implementado.

1.2 Relevância do problema

A elevada competição entre as empresas em um mundo globalizado exige que estas trabalhem com um alto grau de eficiência em todas as suas áreas.

A constatação de Ballou (1998), de que a logística participa de pelo menos um terço do total de despesas da maioria das empresas, mostra a impacto que uma eficiente operação logística tem no resultado de uma empresa.

Dentro das operações logísticas, o transporte rodoviário tem participação muito importante no custo de uma empresa que precise distribuir seus produtos, coletar matérias-primas ou que atue diretamente no mercado de transportes. A incerteza relacionada aos preços do petróleo torna ainda mais preocupante o futuro das despesas com transporte rodoviário.

Assim, um aumento de produtividade na roteirização de veículos é fundamental para a competitividade de empresas que dependam do transporte rodoviário. Conseqüência desta necessidade é o grande número de trabalhos publicados nessa área (Assad, 1998).

Como aponta Cunha (2000), desta maneira surgem inúmeras oportunidades de desenvolvimento e aplicação de modelos matemáticos a problemas reais em sistemas logísticos de transportes, como o problema de roteirização de veículos.

Como a grande maioria dos problemas de roteirização de veículos encontrados na prática apresenta dificuldades na resolução por métodos exatos, eles são resolvidos por processos heurísticos, que buscam respostas ótimas em tempos computacionais viáveis.

Atualmente, segundo Cunha (2006), muitos dos pacotes de roteirização de veículos disponíveis no mercado ainda têm como base heurísticas antigas, como a heurística de economias de Clarke e Wright (1964) ou de varredura de Wren e Holiday (1972) e de Gillet e Miller (1974).

Dentre as classes de modelos que vêm sendo desenvolvidos para solução de problemas de roteirização de veículos, as metaheurísticas - definidas por Corloni (1996) como heurísticas derivadas da observação de fenômenos físicos e biológicos e de inteligência artificial - têm apresentado resultados muito bons, por muitas vezes ótimos, mesmo quando aplicadas a problemas reais em que métodos otimizantes não são viáveis.

Proposto por Holland (1975), o algoritmo genético é uma metaheurística derivada da observação da evolução das espécies e vem sendo aplicado com sucesso em problemas de roteirização de veículos com janelas de tempo e em problemas do caixeiro-viajante.

No entanto, os algoritmos genéticos aplicados a problemas de roteirização de veículos sem janelas de tempo não vinham sendo competitivos quando comparados com algoritmos baseados na metaheurística busca tabu, como os de Taillard (1993) e de Xu e Kelly (1996).

Prins (2004), com sua proposta de algoritmo genético com cromossomos sem delimitadores de rotas e com uma busca local intensa, chegou a resultados (em termos de qualidade de solução e de custo computacional) muito próximos aos obtidos com a busca tabu, inserindo os algoritmos genéticos no grupo das melhores heurísticas para solução de problemas de roteirização de veículos para as instâncias de Christofides et al. (1979).

Este trabalho, portanto, tem como objetivo propor um algoritmo para a solução de um problema fundamental para a competitividade de empresas que dependam de uma roteirização de veículos eficiente.

1.3 Delineamento do trabalho

O capítulo 2 contém uma revisão da literatura, onde são apresentados modelos de solução para problemas de roteirização de veículos. É dado um detalhamento maior aos métodos baseados em algoritmos genéticos.

No capítulo 3 o problema de roteirização de veículos é caracterizado e matematicamente formulado.

O capitulo 4 descreve em detalhes o algoritmo genético publicado por Prins (2004), apresentando também os algoritmos propostos neste trabalho.

Os resultados computacionais dos algoritmos propostos aplicados nas instâncias de Christofides et al. (1979) são apresentados e analisados no capítulo 5.

Finalmente, no capítulo 6 são apresentadas as conclusões desta pesquisa e recomendações para trabalhos futuros.

2 REVISÃO DA LITERATURA

Este capítulo inicia-se com um breve histórico do problema de roteirização de veículos. Em seguida, na seção 2.2, são apresentadas as estratégias para solução do problema, com ênfase nas estratégias metaheurísticas, que correspondem às estratégias mais recentes e com melhores resultados. Na seção 2.3 são descritos os principais trabalhos que envolvem algoritmos genéticos para solução de problemas de roteirização de veículos já publicados na literatura. As duas principais características do algoritmo genético de Prins (2004), utilização de cromossomos sem delimitadores de rotas e formação das rotas a partir do método do particionamento do roteiro gigante também estão na seção 2.3. Os principais trabalhos desenvolvidos no âmbito da Escola Politécnica da Universidade de São Paulo são relacionados na seção 2.4, seguindo-se as considerações finais.

2.1 O Problema de roteirização de veículos

O problema de roteirização de veículos (*VRP – Vehicle Routing Problem*) consiste basicamente em determinar um conjunto de rotas a serem executadas por uma frota de veículos estabelecida em dado depósito central, para atender (coletar, entregar ou visitar) um conjunto de clientes geograficamente dispersos, que têm custos e distâncias associados a suas ligações.

O objetivo é determinar o conjunto de rotas de menor custo que atenda a todos os clientes, respeitando-se as restrições operacionais definidas para cada problema, como a capacidade dos veículos (sendo que a frota pode ser homogênea ou heterogênea), duração das rotas, janelas de tempo para atendimento dos clientes, entre outras.

O primeiro problema estudado envolvendo roteirização foi o problema do caixeiro-viajante (do inglês *TSP – "traveling salesman problem"*), que segundo Cook (2005) foi introduzido pelo economista Karl Menger em Viena, na década de 20. Seu objetivo é estabelecer a melhor rota ou seqüência de cidades a serem visitadas pelo

vendedor, a fim de que este percorra a menor distância possível, visitando cada cidade exatamente uma vez.

Restrições operacionais foram sendo adicionadas ao problema do caixeiro-viajante com a finalidade de representar com mais fidelidade os diferentes tipos de problemas que envolvem a definição de roteiros para pessoas e veículos. De acordo com Cunha (2006), os problemas de roteirização podem ser vistos como problemas de múltiplos caixeiros viajantes com restrições de capacidade e outras restrições que dependem de cada aplicação.

Assim, desde sua formulação original, proposta por Dantzig e Ramser (1959), o problema de roteirização de veículos tem sido abordado de maneiras variadas e apresentado em diferentes formulações, aplicações e estratégias de solução.

As diversas características para taxonomia aplicáveis a problemas de roteirização de veículos, conforme definidas por Bodin et al. (1983), são apresentadas no Quadro 2.1.

2.2 Estratégias de solução para problemas de roteirização de veículos

Diante da grande diversidade de fatores e condicionantes que aparecem em problemas de roteirização, como ilustrado no Quadro 2.1, para que possam ser atingidas soluções de qualidade e para que sejam supridas as demandas particulares de cada problema originado de situações reais, são necessárias estratégias matemáticas eficazes e robustas o suficiente para serem aplicadas nos mais diferentes problemas.

Para Laporte et al. (1992), as estratégias de solução de problemas de roteirização e programação de veículos podem ser divididas em algoritmos exatos e soluções heurísticas.

Quadro 2.1:Taxonomia dos problemas de roteirização de veículos

Características	Opções possíveis
1. Tamanho da frota disponível	um único veículo
	mais de um veículo
2. Tipo de frota disponível	frota homogênea
	frota heterogênea
	tipos especiais de veículos (ex: compartimentalizados)
3. Localização da frota disponível	uma única garagem
	multíplas garagens
4. Natureza da demanda	demanda determinística
	demanda estocástica
	permitida satisfação parcial da demanda
5. Localização da demanda	nos nós (não necessariamente todos)
	nos arcos (não necessariamente todos)
	combinação de nós e arcos
6. Restrições temporais da demanda	não existentes
	janelas de tempo rígidas
	janelas de tempo flexíveis com penalidade
	combinação de janelas de tempo rígidas e flexíveis
7. Rede subjacente	rede direcionada
	rede não-direcionada
	combinação de rede direcionada e não-direcionada
	rede euclidiana
8. Restrições de capacidade do veículo	não existentes
	iguais para todos os veículos
	diferentes para cada veículo
9. Restrições de duração máxima da rota	não existentes
	iguais para todos os veículos
	diferentes para cada rota
10. Operação	somente coletas
	somente entregas
	combinação de coletas e entregas
	com ou sem desmembramento de carga
11. Custos	custos fixos
	custos variáveis
	custos por violação de restrições
	custos por não atendimente de demandas
12. Objetivos	minimizar total de custos variáveis
	minimizar soma de custos fixos e variáveis
	minimizar número de veículos (custo fixo)
	maximizar função objetivo baseada em nível de serviço ou conveniência
	maximizar função objetivo baseada em prioridades do usuário

Fonte: Bodin et al. (1983)

Os autores ainda enumeram algumas metaheurísticas, ou heurísticas modernas, já aplicadas a problemas de roteirização: algoritmos genéticos, colônias de formigas, busca tabu, redes neurais, GRASP, *simulated annealing* e *deterministic annealing*.

O termo *metaheurística*, introduzido pela primeira vez por Glover (1986), é a união de duas palavras gregas: *heurística*, derivada do verbo *heuriskein*, que significa "encontrar", e o sufixo *meta*, que significa "além, acima" (no sentido de superior).

De acordo com Cunha (2006), as metaheurísticas podem ser definidas como as estratégias e técnicas mais recentes e avançadas, que guiam outras heurísticas a fim de se encontrar soluções melhores, ultrapassando o ponto de parada das heurísticas tradicionais.

Segundo Osman (2002), uma metaheurística pode ser definida como um procedimento mestre iterativo que guia e modifica a seqüência de operações de heurísticas subordinadas, a fim de produzir soluções de alta qualidade de maneira eficiente. Pode combinar inteligentemente diferentes idéias para explorar o espaço de solução usando estratégias adaptativas com aprendizado durante o processo. Busca-se evitar ficar preso em um ótimo local (para problemas que apresentam múltiplos ótimos locais) e/ou reduzir o espaço de busca de maneira inteligente.

As metaheurísticas, inspiradas em processos não relacionados à otimização, tem sido foco de extensivas pesquisas, em particular na última década, e seu sucesso pode ser explicado por sua aplicabilidade genérica, flexibilidade para incorporar condicionantes reais, referência a processos observados na natureza e ao excelente *trade-off* entre qualidade da solução, tempo computacional e facilidade de implementação (Pirlot, 1996).

Ainda segundo Laporte (2000), a tendência de evolução para estes métodos de solução seria a de resolver instâncias de maior porte, com redução do tempo de processamento.

Os métodos exatos possibilitam a obtenção da solução exata e ótima para o problema, enquanto métodos heurísticos não garantem que a resposta ótima será obtida. No entanto, seu custo computacional é bem menor.

Segundo Cunha (2006), sob a ótica de otimização, os problemas de roteirização de veículos pertencem à categoria conhecida como NP-difícil (do inglês *NP-hard*), ou seja, possuem ordem de complexidade exponencial. Em outras palavras, o esforço computacional de solução cresce exponencialmente com o tamanho do problema (dado pelo número de pontos a serem atendidos). Em termos práticos, isto significa que geralmente não é possível resolver até a otimalidade, por métodos exatos, problemas de certo porte pertencentes à classe NP-difícil. Nem mesmo o mais avançado computador existente no mercado poderia resolvê-los em tempo de processamento aceitável.

Assim, ainda segundo Cunha (2006), a maioria dos artigos encontrados na literatura para solução de problemas de roteirização de veículos descreve métodos de soluções heurísticos. Essas estratégias de solução baseiam-se em abordagens intuitivas, de modo que a estrutura particular do problema possa ser considerada e explorada de forma inteligente (Cunha, 1997). Esta característica das heurísticas, no entanto, faz com que muitas vezes elas careçam de robustez quando aplicadas aos diferentes problemas de roteirização de veículos, conforme mostrado por Hall e Partyka (1997).

Laporte et al. (2000) apresentam uma revisão dos métodos heurísticos desenvolvidos para problemas de roteirização, e os classificam em *heurísticas clássicas e heurísticas modernas*.

No mesmo trabalho, heurísticas clássicas são subdivididas em heurísticas de construção, que constroem soluções otimizadas para o custo, sem qualquer fase destacada de melhoria; heurísticas de melhoria, que buscam melhorias da solução através de trocas de segmentos intra e inter rotas; e heurísticas de duas fases, que agrupam vértices em rotas possíveis e constroem rotas com trocas de informações entre si. Entre os principais métodos apontados para estes tipos de heurísticas destacam-se, respectivamente, o do vizinho mais próximo, a heurística das

economias de Clarke e Wright (1964), e a heurística de varredura de Gillet e Miller (1974).
to e quantidade de parâmetros, ainda que com discreto prejuízo na qualidade das soluções, a fim de tornar possível a implantação destas metodologias em pacotes comerciais de roteirização.

2.2.1 Metaheurísticas aplicadas a problema de roteirização de veículos

Nesta seção serão apresentadas sucintamente as principais metaheurísticas aplicadas a problemas de roteirização de veículos encontradas na literatura.

Algoritmo Genético

O algoritmo genético é uma metaheurística proposta por John Holland (1975) que se inspira na teoria de evolução darwiniana, que considera que quanto melhor um indivíduo se adapta ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes.

Em linhas gerais, os algoritmos genéticos trabalham com um grupo (ou população) de soluções. Cada indivíduo é representado por um cromossomo, que armazena as informações genéticas. Os cromossomos são compostos por genes, os quais são responsáveis pelas características dos seres e são trocados ou transmitidos durante o processo de reprodução.

A cada iteração, indivíduos da população são selecionados para reprodução, sendo os mais aptos com maior probabilidade de serem transmitidos para a geração seguinte, e os demais menos aptos com maior probabilidade de serem eliminados de acordo com o princípio darwiniano de seleção natural e sobrevivência do mais forte.

A Figura 2.1 ilustra o esquema geral de funcionamento do algoritmo genético:

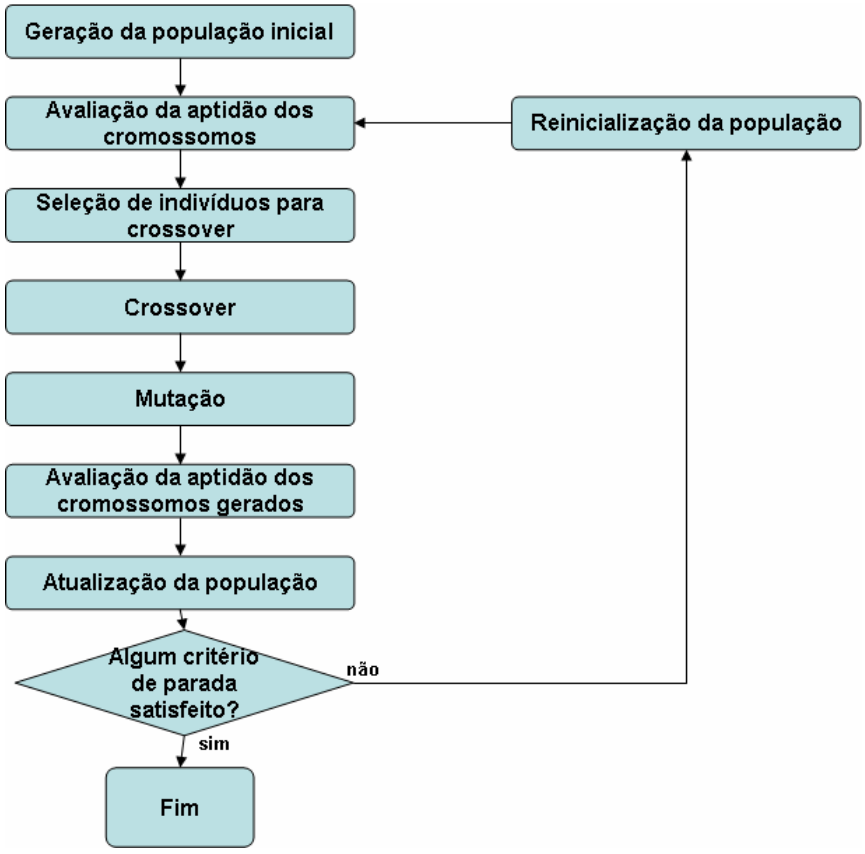


Figura 2.1: Esquema de funcionamento do algoritmo genético

Entre os algoritmos genéticos para solução de problemas de roteirização de veículos propostos, temos o de Shmitt (1995) e o de Van Breedam (1996), com resultados pouco animadores. Berger e Barkaoui (2003) e Prins (2004), apresentaram algoritmos genéticos híbridos com resultados promissores. A seção 2.3 detalha estes algoritmos.

Busca Tabu

Proposta por Glover (1986) – para um trabalho mais recente, ver Glover e Laguna (1997) – a metaheurística busca tabu é uma busca em vizinhança que procura não percorrer regiões do espaço de soluções já visitadas e que não sejam promissoras.

Para evitar que a busca vá para tais regiões, é criada implícita ou explicitamente uma lista de movimentos "tabu", ou seja, movimentos proibidos que, se realizados, poderiam levar a busca a essas regiões.

A mais importante associação com o uso tradicional da palavra "tabu" vem do fato de que proibições são transmitidas por meio de uma memória social, sujeita a modificações no tempo; ou seja, o que é proibido no momento corrente pode não ser proibido em um momento futuro, e vice-versa. Isso cria a ligação fundamental do sentido do termo "tabu" na busca tabu. O conjunto de elementos proibidos da busca tabu faz parte de uma memória evolutiva, o que possibilita sua alteração de acordo com o tempo e a circunstância (Cunha, 2006).

Partindo-se de uma solução inicial, essa heurística de busca local baseia-se na exploração da vizinhança de uma solução, vizinhança essa definida pelos movimentos considerados em cada implementação específica. Em problemas de roteirização, por exemplo, pode-se imaginar um movimento de troca de dois clientes que pertencem a rotas diferentes, ou ainda realocação de um cliente, que troca sua posição dentro da própria rota, ou é movimentado para uma determinada posição em outra rota. Cada movimento transforma uma solução intermediária em outra, e a cada iteração um movimento da vizinhança é selecionado, sendo o processo repetido até se atingir um critério de parada (Cunha, 2006).

Pode-se citar os trabalhos de Gendreau et al. (1994), Taillard (1993), Taillard (1999), Gendreau et al. (1999) e Wassan e Osman (2002) como exemplos bem sucedidos de aplicações de busca tabu a problemas de roteirização de veículos. Das quatorze instâncias de teste apresentadas em Christofides et al. (1979), o trabalho de Taillard (1993) atingiu cinco das melhores soluções conhecidas e apresenta em média um desvio de 0,86% das melhores soluções.

GRASP

Proposta por Feo e Resende (1989, 1995), GRASP (*Greedy Randomized Adaptative Search Procedure,* ou procedimento de busca adaptativo randomizado guloso) é uma metaheurística iterativa em que cada iteração é composta por duas etapas: construção e busca local. Na etapa de construção, determina-se uma solução viável, enquanto na busca local, a vizinhança desta solução é explorada até que se encontre um ponto de ótimo local.

Na fase de construção, a solução viável é formada por seguidas iterações que adicionam um novo elemento por vez à solução em construção. Em um problema de roteirização de veículos, esse novo elemento é um cliente a ser atendido.

O elemento que será adicionado à solução em construção é selecionado a partir de uma lista restrita de candidatos (LRC), composta pelos elementos que menos incrementem o custo da solução em construção, segundo uma função de avaliação gulosa (*greedy*). Definida a lista restrita de candidatos, o novo elemento é escolhido aleatoriamente (*randomized*). Antes da adição do próximo elemento à solução em construção, os custos incrementais de todos os elementos são recalculados, o que dá a característica adaptativa ao GRASP.

Construída a solução viável, o GRASP entra na fase de busca local, em que a solução corrente é substituída por uma solução melhor de sua vizinhança, até o ótimo local ser obtido. Nesta fase, diversos métodos de busca em vizinhança (inclusive metaheurísticas) podem ser aplicados.

As aplicações de GRASP a problemas de roteirização de veículos têm ocorrido principalmente na forma de algoritmos híbridos. Como exemplo, temos o GRASP aplicado em conjunto com busca tabu (Tortelly e Occhi 2006, Ho e Haugland, 2004) e em conjunto com métodos de aprendizagem (Atkinson, 1998).

Simulated Annealing

Simulated Annealing, também conhecida como têmpera (ou recozimento) simulada, é uma metaheurística proposta por Kirkpatrick et al. (1983), inspirada no processo térmico de resfriamento de um material, com a virtude de permitir escapar de um ótimo local através da aceitação de movimentos que pioram a solução corrente. O termo *simulated annealing* deve-se à analogia com a termodinâmica, mais especificamente um processo físico, denominado recozimento (*annealing*), no qual um material é aquecido até atingir o estado líquido e em seguida, resfriado. Em altas temperaturas, os átomos têm liberdade de se moverem livremente. A medida que a temperatura diminui, os átomos tendem a se cristalizar num sólido. Se a temperatura é reduzida muito rapidamente, o sólido cristalizado não apresenta forma. Se o resfriamento ocorre vagarosamente, maiores são as chances de se formar um cristal perfeito, ou seja, numa estrutura cristalina de energia mínima global. Para apresentar bons resultados, um resfriamento depende de uma temperatura inicial alta e de um resfriamento lento, a fim de evitar uma estrutura irregular e fraca, com alta energia em decorrência do esforço interno despendido (Cunha, 2000).

O algoritmo inicia-se com a definição de uma solução inicial viável S_0, de uma temperatura T_0 e de uma função de redução de temperatura α.

A partir de S_0, o algoritmo gera aleatoriamente uma solução vizinha *S'*. Se *S'* é melhor do que S_0, ela é aceita como solução corrente. Caso contrário, a solução tem uma probabilidade $p(T)$ de ser aceita, e quanto maior é a temperatura *T*, maior a chance de ela ser aceita.

A temperatura T é resfriada à razão α, reduzindo-se, assim, a probabilidade de se aceitar uma solução pior.

Usualmente a temperatura deve decrescer até zero, sendo esse o critério de parada usual do SA. Entretanto, tal critério pode levar a tempos de processamento muito elevados; na prática, basta reduzir a temperatura até um nível que a seja virtualmente impossível aceitar um movimento de piora (Cunha, 2006).

Bent e Hentenryck (2004) propuseram uma heurística híbrida, baseada em *simulated annealing* e busca local em vizinhança de grande porte para o problema de roteirização de veículos com janelas de tempo em que se busca primeiramente minimizar o número de veículos necessários e posteriormente a distância total percorrida. Nesse trabalho foram obtidos os melhores resultados da literatura para o problema de roteirização de veículos com janelas de tempo.

2.3 Algoritmos genéticos para o problema de roteirização de veículos

Esta seção inicia-se com uma breve descrição de algoritmos genéticos para problemas de roteirização de veículos publicados, abordando com maior profundidade, nas seções 2.3.2 e 2.3.3, dois fatores fundamentais para o sucesso do algoritmo de Prins (2004): codificação do cromossomo sem delimitadores de rotas, o que permite a utilização do operador de *crossover* do tipo OX, e o método de particionamento do roteiro gigante.

2.3.1 Principais trabalhos na literatura envolvendo algoritmos genéticos para o problema de roteirização de veículos

Segundo Gendreau et al. (1998) e Golden et al. (1998), a metaheurística que apresenta os melhores resultados para solução de problemas de roteirização de veículos sem restrições temporais é a busca tabu, com resultados bem superiores àqueles obtidos por outras metaheurísticas, como *simulated annealing* e algoritmos genéticos.

No que tange aos algoritmos genéticos, esta constatação não é óbvia, uma vez que, para outros problemas de roteirização de veículos, eles têm alcançado resultados muito bons, como os obtidos por Potvin (1996) para o problema do caixeiro-viajante e por Potvin e Bengio (1996) e Tangiah (1993) para problemas de roteirização de veículos com janela de tempo.

Van Breedam (1996), por exemplo, construiu um algoritmo com delimitadores de rotas, ou seja, os cromossomos eram formados pelos pontos a serem atendidos e entre o final de uma rota e o início de outra havia um gene ocupado pelo depósito. Assim, as operações de *crossover* acabavam produzindo soluções inviáveis (ver seção 2.3.2) que eram rejeitadas, reduzindo a eficiência do algoritmo.

O objetivo de Van Breedam (1996) era o de estudar o impacto de diversos parâmetros em métodos para solução de problemas de roteirização de veículos baseadas em algoritmos genéticos e em *simulated annealing*, comparando seus resultados com os obtidos por um método baseado em busca tabu. Assim, apesar de resultados semelhantes para as três metaheurísticas, não é possível medir a qualidade de seus algoritmos, já que ele não utilizou instâncias da literatura nem comparou os resultados com aqueles disponíveis na literatura.

O que talvez explique o fato de modelos baseados em algoritmos genéticos não terem obtido bons resultados nas tentativas iniciais seja o modo de construção dos cromossomos. As primeiras tentativas foram baseadas em cromossomos com delimitadores de rotas, o que obrigava o algoritmo a executar procedimentos de reparo para transformar as soluções inviáveis obtidas nos *crossover*s em soluções viáveis.

Para corrigir tais distorções dos cromossomos com delimitadores de rotas, é preciso utilizar algoritmos que corrigem os cromossomos gerados, o que eleva o custo computacional do *crossover* e prejudica a transmissão de bons trechos de rotas para as gerações seguintes (Prins 2004).

Tais procedimentos são conhecidos por enfraquecer a transmissão da informação genética dos pais para os filhos, apesar de haver alguns algoritmos reparadores promissores, como o apresentado por Mitchell et al. (2003), que segundo os autores tais algoritmos geram boas soluções quando aplicado a problemas com menos de 1000 pontos de atendimento.

Para não utilizar procedimentos de correção de cromossomos, Schmitt (1995) elaborou um algoritmo genético com cromossomos sem delimitadores de rotas, em que as rotas são criadas pela ordem dos genes do cromossomo. Uma nova rota é iniciada no gene que não pôde ser incluído na rota anterior por ter afetado restrições de carga ou de distância percorrida.

Schmitt (1995) aplicou seu algoritmo nas 14 instâncias propostas por Christofides et al. (1979) com resultados pouco animadores. Apesar de seus resultados terem superado aqueles de métodos heurísticos mais simples, como o de Clarke e Wright (1964), eles são bem inferiores àqueles atingidos com heurísticas um pouco mais elaboradas.

Para atenuar os pontos fracos dos algoritmos genéticos, como longos tempos de processamento e convergência prematura para um ótimo local, recentemente vêm sendo desenvolvidos algoritmos genéticos híbridos.

Nos algoritmos genéticos híbridos, o algoritmo genético em si é utilizado para uma exploração global do espaço de soluções, enquanto outras heurísticas são utilizadas para realizar uma exploração local.

Berger e Barkaoui (2003), por exemplo, apresentaram um algoritmo genético hibrido em que são criadas duas populações que periodicamente trocam os seus indivíduos. Os resultados para as instâncias de Christofides et al. (1979) são muito bons, mas inferiores aos obtidos por Prins (2004) tanto em termos de custo computacional quanto de qualidade das soluções.

Prins (2004) apresentou um algoritmo genético híbrido que se mostrou superior a todos os outros algoritmos genéticos publicados, quando aplicados às instâncias de Christofides et al. (1979) e Golden et al. (1998). Segundo ele, são dois os fatores responsáveis pelo sucesso de seu método:

a) codificação de seu cromossomo sem delimitadores de rotas, o que permite a utilização do operador de *crossover* OX (*order crossover*), que é o método de

cruzamento utilizado com mais sucesso nos algoritmos genéticos aplicados a problemas do tipo caixeiro-viajante e de roteirização de veículos (Prins, 2004);

b) formação das rotas pelo método de particionamento do roteiro gigante (Prins (2004) adota o nome de *split procedure*) o que garante que, para a seqüência de clientes determinada pelo cromossomo, seja obtido a roteiro ótimo.

Assim, o algoritmo genético de Prins (2004) pode ser classificado com um algoritmo do tipo *route-first, cluster-second*, em que os cromossomos são melhorados a cada geração pelo paralelismo do algoritmo genético, com a utilização de um *crossover* de sucesso (OX). Em seguida, o método de particionamento simples do roteiro gigante, produz rotas ótimas, respeitando a ordem dos pontos no cromossomo.

2.3.2 Crossover OX

Entre as principais maneiras de se representar um cromossomo, estão as descritas abaixo (Michalewicz 1996):

Representação ordinal

Nesta forma de codificação, cada solução é representada como uma lista de *n* cidades, onde o *i-ésimo* elemento da lista é um número entre *1* e *n – i +1*. Existe uma lista ordenada de cidades que serve como referencia para construir a representação. A lista ordenada *l*, varia de 1 a *n* cidades. O cromossomo é um vetor de posicionamento da cidade na lista. Por exemplo, seja o cromossomo *p* = (1, 1, 2, 3, 1, 1) e a lista *l* = (1, 2, 3, 4, 5, 6).

O primeiro elemento do cromossomo *p* é 1. Ele corresponde ao primeiro elemento da lista *l* que é o 1 (Rota: 1). O próximo elemento do cromossomo *p* é 1. Ele corresponde ao primeiro elemento da lista *l* atualizada (após remoção do 1) que é o 2 (Rota: 1 2). O próximo número do cromossomo *p* é 2. Tal número corresponde ao segundo elemento da lista *l* atualizada (sem o 1 e o 2) que é o 4 (Rota: 1 2 4). O próximo elemento do cromossomo é 3, correspondendo ao 6 na lista *l* atualizada

(Rota: 1 2 4 6). Seguindo este raciocínio, chega-se à rota final: 1 2 4 6 3 5. Essa é a solução representada por este cromossomo.

Representação por adjacência

Cada cromossomo é representado como uma lista de *n* cidades. A cidade *j* da rota é listada na posição *i* se, e somente se, a rota vai da cidade *i* a cidade *j*. Por exemplo, um cromossomo *p* = (3, 1, 5, 6, 4, 2) apresenta os seguintes arcos em uma rota associado a *p*: 1→ 3; 2 → 1; 3 → 5; 4 → 6; 5 → 4; 6 → 2. Assim, o cromossomo representa a seguinte solução: 1 3 5 4 6 2.

Representação por permutação (ou caminho)

O cromossomo é formado pela seqüência dos nós na solução. Por exemplo, o cromossomo *p* = (1, 4, 2, 3, 6, 5) representa diretamente a solução 1 4 2 3 6 5.

A representação de cromossomos mais utilizada em problemas de roteirização de veículos e do caixeiro-viajante é a representação por permutação (Michalewicz 1996). No entanto, os operadores de *crossover* usualmente utilizados nos algoritmos genéticos não são aplicáveis a problemas cujo cromossomo seja representado por permutação. Isso decorre do fato de que tais operadores, nesses casos, podem gerar indivíduos inconsistentes (Cunha 2006).

Para ilustrar o fato como os operadores de *crossover* comuns geram soluções inviáveis, tome-se, por exemplo, o problema do caixeiro-viajante, em que o cromossomo de solução é representado por uma seqüência ordenada de cidades a serem visitadas. Sejam A e B dois pais selecionados para cruzamento, tal que:

A= 3 5 7 1 2 4 8 6 9
B= 1 9 2 3 4 6 8 7 5

Em outras palavras, o cromossomo A corresponde à seguinte seqüência de cidades: 3-5-7-1-2 4-8-6-9-3.

Suponha que seja aplicado um operador de cruzamento em dois pontos, após a 2ª e 6ª posições, gerando os seguintes filhos:

3 5 | 2 3 4 6 | 8 6 9
1 9 | 7 1 2 4 | 8 7 5

Ambas as soluções acima não representam roteiros válidos para o problema do caixeiro-viajante, uma vez que algumas cidades aparecem duas vezes, enquanto outras cidades não são visitadas (por exemplo, as cidades 3 e 6 aparecem duas vezes no primeiro filho, enquanto as cidades 1 e 7 não fazem parte do roteiro).

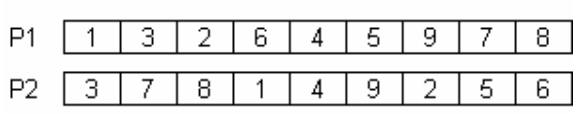
A fim de evitar esse tipo de problema, existem operadores específicos para tratar cromossomos que representam uma permutação.

O operador OX é um exemplo de operador que permite tal tratamento. Para tanto, escolhem-se aleatoriamente dois pontos de corte que irão determinar uma sub-rota; um pré-filho é então criado, copiando-se a sub-rota na posição correspondente do pai 1; os elementos da sub-rota são excluídos do pai 2; a partir do segundo ponto de corte, são preenchidas as posições vazias com os elementos não excluídos do pai 2; quando o fim do cromossomo foi atingido, continua-se a partir da primeira posição deste; finalmente, o segundo filho é construído, seguindo-se o mesmo procedimento.

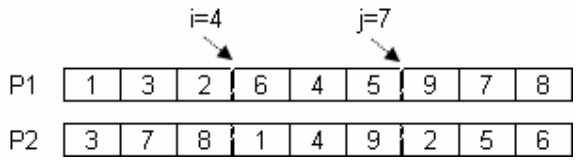
Além do operador especial OX, existem também os operadores CX e PMX (Cunha 2006).

O exemplo da Figura 2.2 mostra como o *crossover* OX gera dois filhos F1 e F2 a partir dos pais P1 e P2:

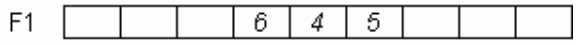
Passo 1) Sejam os cromossomos pais P1 e P2:



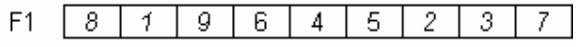
Passo 2) São escolhidos dois pontos de corte *i* e *j*. No exemplo, *i =4* e *j=7* :



Passo 3) Os pontos de P1 entre *i* e *j* são passados para o filho F1 :



Passo 4) O filho F1 é completado com os pontos de P2 a partir de *j*, tomando-se o cuidado de não inserir duas vezes o mesmo ponto :



Passo 5) O mesmo processo é realizado para obtenção de F2 :

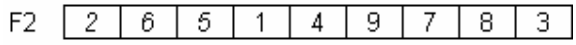


Figura 2.2: *Crossover* OX em cromossomo sem delimitadores de rotas

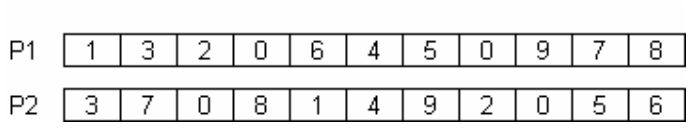
Os cromossomos da Figura 2.3, que são do mesmo tipo proposto por Prins (2004), são compostos apenas pelos pontos a serem atendidos, sendo as rotas obtidas pelo método de particionamento do roteiro gigante (que no caso é o próprio cromossomo), apresentado ainda nesta seção.

Faz-se necessária, neste ponto, a apresentação do conceito de delimitadores de rotas no contexto da roteirização de veículos. Trata-se de um gene (neste trabalho, é

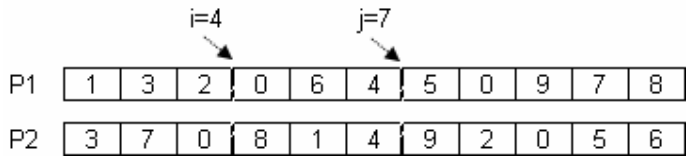
o gene “0”) a ser inserido no cromossomo com o objetivo de sinalizar o final de uma rota e início de outra.

Diferentemente do cromossomo da Figura 2.2, o exemplo da Figura 2.3 mostra as dificuldades de se aplicar o *crossover* OX em cromossomos com delimitadores de rotas:

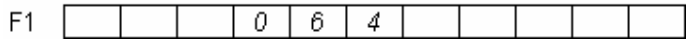
Passo 1) Sejam os cromossomos pais P1 e P2:



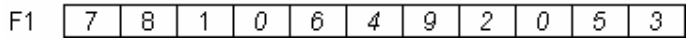
Passo 2) São escolhidos dois pontos de corte *i* e *j*. No exemplo, *i =4* e *j=7* :



Passo 3) Os pontos de P1 entre *i* e *j* são passados para o filho F1 :



Passo 4) O filho F1 é completado com os pontos de P2 a partir de *j*, tomando-se o cuidado de não inserir duas vezes o mesmo ponto :



Passo 5) O mesmo processo é realizado para obtenção de F2 :

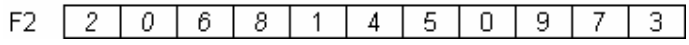


Figura 2.3: *Crossover* OX em cromossomo com delimitadores de rotas

Analisando-se os filhos F1 e F2 gerados no exemplo da Figura 2.3, percebe-se que os cromossomos com delimitadores de rotas:

- podem gerar soluções inviáveis (não há certeza que as restrições de custo e capacidade das rotas criadas são respeitadas)

- ou podem gerar soluções com uma qualidade ruim (a primeira rota do filho F2, por exemplo, é uma rota que provavelmente prejudica a qualidade da solução, já que atende apenas um ponto).

2.3.3 Particionamento do roteiro gigante

É justamente o particionamento eficiente do cromossomo sem delimitadores de rotas aplicado por Prins (2004), o que diferencia seu algoritmo genético dos anteriores desenvolvidos para problemas de roteirização de veículos.

Para obter as rotas a partir do de um cromossomo, Prins (2004) utiliza o método de particionamento simples do roteiro gigante proposto por Golden et al. (1982).

Este método é executado sobre um grafo orientado em que:
- cada nó representa um cliente a ser atendido;
- os custos dos arcos que conectam cada nó *i*, *j* é o custo da rota que atende todos os clientes entre *i* e *j* (inclusive *i* e *j*).

Como exemplo, a Tabela 2.1 apresenta as coordenadas dos clientes A, B, C, D, E, enquanto a Figura 2.4 mostra o grafo associado ao roteiro gigante A->B->C->D->E. Os valores apontados nos arcos que ligam os nós *i* , *j* correspondem ao custo da rota que atende todos os clientes entre *i* e *j* (inclusive *i* e *j*).

Por exemplo, o valor do arco (A,D)=5 é o custo das viagens entre A->B (2), B->C (1) e C->D (2).

Tabela 2.1: Coordenadas dos nós do roteiro gigante

Cliente	Coodernada x	Coordenada y
A	0	0
B	0	2
C	1	2
D	1	4
E	2	6

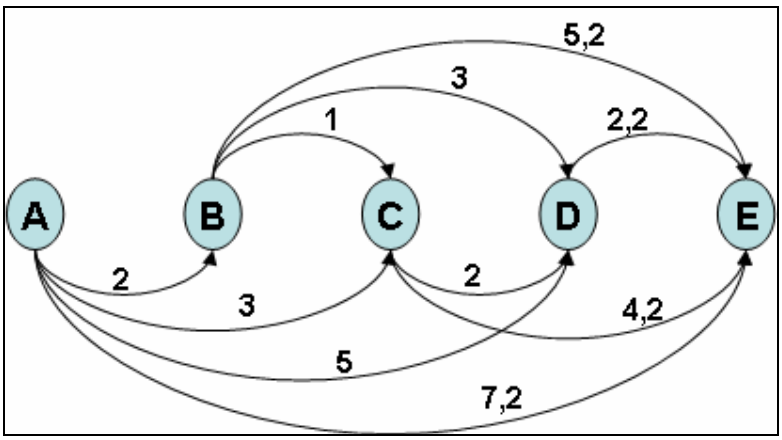


Figura 2.4: Grafo que representa o roteiro gigante

Caso os veículos disponíveis não tenham capacidade para atender todos os clientes entre *i* e *j*, o custo do arco entre *i* e *j* é infinito.

Definidos os custos dos arcos, deve-se obter o caminho de menor custo entre o primeiro e o último nó do grafo. Para isso, diversos algoritmos podem ser utilizados, como os de Dijkstra (Ahuja et al., 1993), Bellman (Ahuja et al., 1993) e Ulusoy (1985). Cada arco que compõe o caminho de menor custo é uma rota do roteirização final, que é a ótima para a seqüência de pontos do cromossomo.

2.3.4 Algoritmo de Dijkstra

O algoritmo de Dijkstra (Ahuja et al., 1993) é um dos algoritmos que calcula o caminho de custo mínimo entre vértices de um grafo e será detalhado nesta seção. É muito semelhante ao algoritmo de Ulusoy (1985) e é menos genérico do que o de

Bellman (Ahuja et al., 1993), já que este pode ser aplicado a problemas com custos negativos, o que não é o caso do problema tratado nesta dissertação. O algoritmo de Dijkstra é bastante simples e com um bom nível de desempenho.

No algoritmo de Dijkstra, escolhido um nó como raiz da busca, é calculado o custo mínimo deste nó para todos os demais nós do grafo.

Este algoritmo parte de uma estimativa inicial para o custo mínimo e vai sucessivamente ajustando esta estimativa. Ele considera que um nó estará fechado quando já tiver sido obtido um caminho de custo mínimo do nó tomado como raiz da busca até ele. Caso contrário ele é dito estar aberto.

Seja *G(V,A)* um grafo orientado e *s* um nó de *G*:

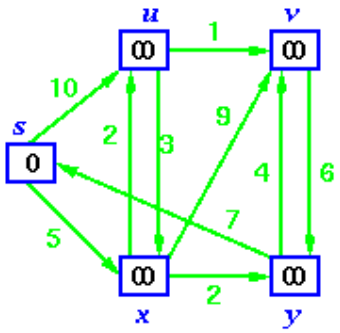
1. Atribua valor zero à estimativa do custo mínimo do nó *s* (a raiz da busca) e infinito às demais estimativas;
2. Atribua um valor qualquer aos precedentes (o precedente de um nó *t* é o nó que precede *t* no caminho de custo mínimo de *s* para *t*);
3. Enquanto houver nó aberto:
    - seja *k* um nó ainda aberto cuja estimativa seja a menor dentre todos os nós abertos;
    - feche o nó *k;*
    - Para todo nó *j* ainda aberto que seja sucessor de *k* faça:
        - some a estimativa de custos do nó *k* com o custo do arco que une *k* a *j*;
        - caso esta soma seja melhor que a estimativa anterior para o nó *j*, substitua-a e anote *k* como precedente de *j*.

Quando todos os nós tiverem sido fechados, os valores obtidos serão os custos mínimos dos caminhos que partem do nó tomado como raiz da busca até os demais nós do grafo. O caminho propriamente dito é obtido a partir dos nós chamados acima de precedentes.

A seqüência da figura 2.5 ilustra o funcionamento do Algoritmo de Dijkstra:

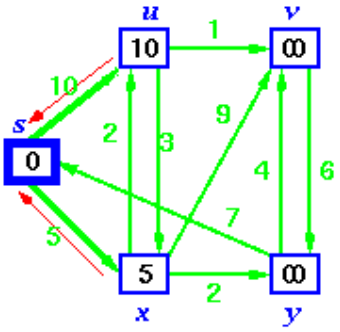
- Inicialmente todos os nós têm um custo infinito, exceto ***s*** (a raiz da busca) que tem valor 0:

Nós	***s***	***u***	***v***	***x***	***y***
estimativas	0	∞	∞	∞	∞
precedentes	-	-	-	-	-



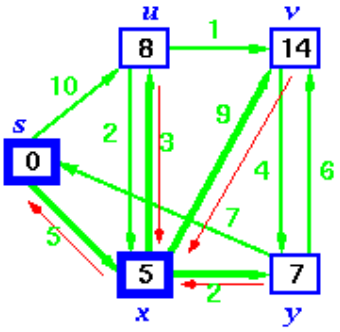
- selecione ***s*** (nó aberto de estimativa mínima)
- feche ***s***
- recalcule as estimativas de *u* e *x*

Nós	***s***	***u***	***v***	***x***	***y***
estimativas	**0**	10	∞	5	∞
precedentes	***s***	*s*	-	*s*	-



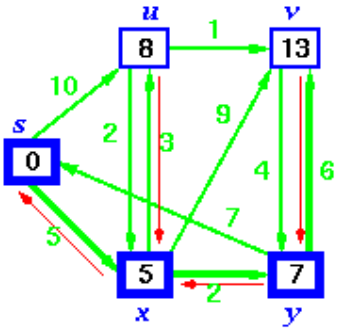
- selecione ***x*** (nó aberto de estimativa mínima)
- feche ***x***
- recalcule as estimativas de *u,v* e *y*

Nós	***s***	***u***	***v***	***x***	***y***
estimativas	**0**	8	14	*5*	7
precedentes	***s***	*x*	*x*	***s***	*x*



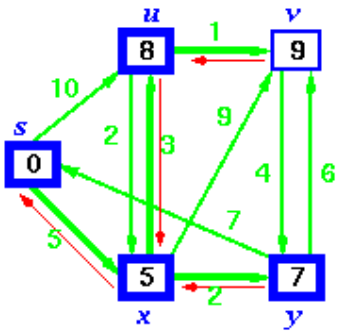
- selecione y (nó aberto de estimativa mínima)
- feche y
- recalcule a estimativa de v

Nós	**s**	**u**	***v***	**x**	**y**
estimativas	**0**	8	13	*5*	**7**
precedentes	***s***	*x*	*y*	***s***	***x***



- selecione u (nó aberto de estimativa mínima)
- feche u
- recalcule a estimativa de v

nós	S	u	v	x	y
estimativas	0	8	9	5	7
precedentes	s	x	u	s	x



- selecione v (nó aberto de estimativa mínima)
- feche v

nós	s	u	v	x	y
estimativas	0	8	9	5	7
precedentes	s	x	u	s	x

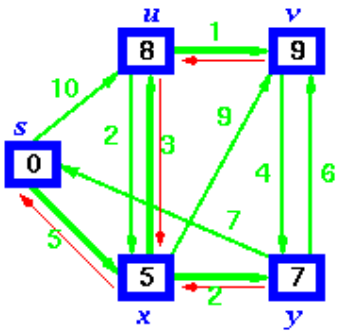


Figura 2.5: Algoritmo de Dijkstra: passo-a-passo

Quando todos os nós tiverem sido fechados, os valores obtidos serão os custos mínimos dos caminhos que partem do nó tomado como raiz da busca até os demais nós do grafo. O caminho propriamente dito é obtido a partir dos nós chamados acima de precedentes.

Para exemplificar, considere o caminho de custo mínimo que vai de *s* até *v*, cujo custo mínimo é 9. O nó precedente de *v* na última das tabelas acima é *u*. Sendo assim, o caminho é:

$s \rightarrow ... \rightarrow u \rightarrow v$

Por sua vez, o precedente de *u* é *x*. Portanto, o caminho é:

$s \rightarrow ... \rightarrow x \rightarrow u \rightarrow v$

Por último, o precedente de *x* é o próprio nó *s*. Logo, o caminho de custo mínimo é:

$s \rightarrow x \rightarrow u \rightarrow v$

2.3.5 Particionamento múltiplo do roteiro gigante

O caminho mínimo de um roteiro gigante obtido pelo algoritmo de Dijkstra depende da definição do primeiro nó do roteiro gigante, que é aquele em que se iniciará a primeira viagem da primeira rota.

Para melhorar a qualidade da solução obtida, Golden et al. (1982) propôs o método de particionamento múltiplo do roteiro gigante, cuja idéia é variar o primeiro nó do roteiro gigante visando à obtenção de soluções diferentes para o mesmo roteiro gigante.

Após a obtenção de um particionamento do roteiro gigante, o processo de particionamento deverá ser iniciado pelo nó seguinte àquele em que o processo anterior foi iniciado. O procedimento deverá ser repetido M vezes, onde M é calculado segundo a equação abaixo:

$$\sum_{1}^{M-1} di < Cv \leq \sum_{1}^{M} di$$

onde:

d_i é demanda no ponto i

C_v é a capacidade do maior veículo.

A melhor solução para o roteiro gigante é aquela com menor custo entre todas as M soluções obtidas.

2.3.6 Diferença entre particionamento de Prins (2004) e Schmitt (1995)

Conforme explicado, no algoritmo de particionamento simples do roteiro gigante, a ordem dos genes (ou dos pontos de entrega) no cromossomo é respeitada, mas um gene, mesmo que possa ser inserido na rota do gene anterior, pode iniciar uma nova rota, caso isto reduza o custo total.

O exemplo a seguir mostra a diferença no funcionamento do método de particionamento proposto por Prins (2004) com o de Schmitt (1995) (ver seção 2.3.1) para o algoritmo genético aplicado ao problema de roteirização de veículos, diferença esta que contribua para explicar a melhor qualidade dos resultados obtidos por Prins (2004).

Dado um depósito (retângulo em verde) e cinco pontos a serem atendidos, suponhamos que:

- os veículos que estão no depósito possam atender no máximo três pontos em um dia
- o cromossomo tem os genes na ordem 1-2-3-4-5

O algoritmo de Schmitt (1995) formaria uma rota com os pontos 1,2 e 3 e uma segunda rota com os pontos 4 e 5, conforme mostra a Figura 2.4.

O algoritmo de particionamento simples do roteiro gigante não incluiria o ponto 3 na primeira rota, já que este está localizado mais próximo dos pontos 4 e 5. Assim, teria como resultado a solução ótima para o cromossomo 1-2-3-4-5, como mostra a Figura 2.5.

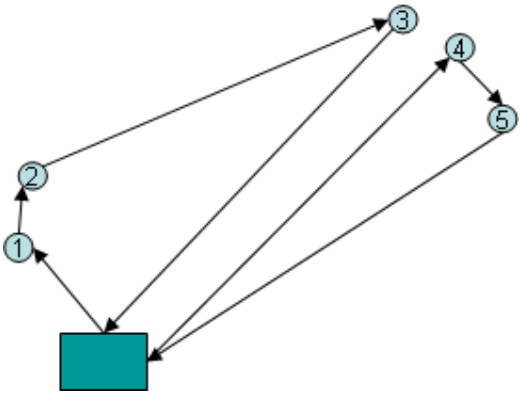


Figura 2.6: Rotas formadas por Schmitt

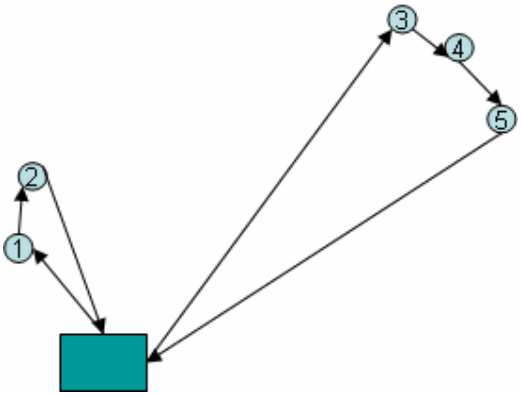


Figura 2.7: Rotas formadas pelo método de particionamento simples do roteiro gigante

Nota-se que o método de particionamento simples do roteiro gigante construiu roteiros mais econômicos do que Schmitt (1995).

Vale dizer que um processo similar ao método do particionamento simples do roteiro gigante já havia sido utilizado por Beasley (1983) como a segunda fase de uma heurística *route-first cluster-second*. A primeira fase consistia na solução de um problema do caixeiro-viajante gigante passando por todos os pontos de entrega. No entanto, os resultados de Beasley (1983) nunca superaram os das mais tradicionais heurísticas para o problema de roteirização de veículos.

Analisando a solução de Beasley (1983), observa-se que o método do particionamento simples do roteiro gigante certamente teria resultados melhores se, ao invés de ser executado apenas uma vez como fase seguinte a um problema do caixeiro-viajante (como fez Beasley (1993), ele fosse aplicado em um algoritmo genético. Isto porque é certo que existe um cromossomo cuja seqüência de genes resulta na solução ótima para o problema se aplicado o particionamento simples do roteiro gigante. Fica a cargo do paralelismo do algoritmo genético encontrar tal cromossomo ótimo.

2.4 Trabalhos desenvolvidos no âmbito da Escola Politécnica da USP

O presente trabalho pretende integrar a produção científica das áreas de Logística e Engenharia de Transportes da Escola Politécnica da USP para o desenvolvimento e implementação computacional de novos algoritmos e heurísticas de solução de problemas de roteirização e programação de veículos, juntamente com o que será relacionado a seguir.

Cunha (1997) utilizou-se da relaxação lagrangeana para um modelo de problema de roteirização de veículos com restrições operacionais de janela de tempo e duração máxima da jornada, desenvolvendo duas heurísticas para frota homogênea e uma outra para frota heterogênea.

Brejon (1998) aborda um problema de programação de transporte de suprimentos para unidades marítimas de exploração de petróleo, visando a garantia de que os suprimentos necessários estejam na unidade marítima solicitante na quantia correta e dentro dos horários solicitados. É analisado como um problema de roteirização e programação de veículos com restrição de janelas de tempo. A estratégia de solução utilizada se baseia na heurística de inserção I1, proposta por Solomon (1987), modificada para se adequar às restrições do problema estudado. A alocação dos clientes às rotas segue um critério de primeiro inserir em uma rota os clientes mais distantes e depois os clientes não alocados que têm menor valor de fim de janela de tempo, obviamente a inserção é feita juntamente com um estudo de viabilidade de inserção, que está relacionado a limites de capacidade, distância e tempo.

Souza (1999) tratou do problema de transporte do tipo “Carga Única Coleta e Entrega” (*Full Truck Load and Delivery*), com duas etapas obedecendo a janelas de tempo. A primeira seria para geração dos roteiros viáveis que atendessem às requisições dentro do período de programação, e a segunda para seleção, dentre estes, dos melhores roteiros, com custo total mínimo, a partir de um problema de programação linear do tipo particionamento de conjunto (*set partitioning*).

Znamensky (2000) propôs uma estratégia de solução para o problema de roteirização e programação de veículos para transporte porta-a-porta de idosos e portadores de necessidades especiais, por meio de veículos de pequena capacidade, do tipo peruas ou vans – um problema de porte real. A formulação utilizada foi de múltiplas bases e utilizou a heurística de inserção paralela, obtendo soluções de excelente qualidade, com redução de frota e custos operacionais, com agilidade computacional.

Teixeira (2001) desenvolveu heurísticas baseadas no algoritmo *out-of-kilter* para solucionar o problema de dimensionamento e roteirização de frota heterogênea, elaborando e combinando rotas para circulação com custo mínimo.

Fonseca (2002) utilizou-se dos conceitos da heurística de inserção de Solomon (1987) para propor uma estratégia de solução, com minimização de tempos e

custos, para rotas de coleta e entrega de mercadorias porta-a-porta por empresas de remessa expressa em grandes centros urbanos, associando, porém, alocação seqüencial dos atendimentos e inserção paralela em cada rota do conjunto.

Feriancic (2005) estudou o problema de distribuição de combustíveis com caminhões tanque para postos de abastecimento. O caminhão tanque é compartimentado, sendo que cada compartimento tem que estar sempre completamente cheio ou vazio. A estratégia de solução proposta baseia-se em definir a alocação ótima dos pedidos aos veículos e a seqüência de entrega de cada veículo. A heurística proposta baseia-se no GRASP, a fim de promover vários reinícios da heurística de alocação seqüencial, realocando apenas alguns clientes escolhidos aleatoriamente.

Mourad (2005) tratou o problema de roteirização de carga completa com janelas de tempo, incorporando algumas características do mercado de transportes brasileiro, como a possibilidade de utilização de uma frota *spot* como complemento de uma frota dedicada. Foram propostas quatro diferentes heurísticas, todas baseadas em busca tabu, que se diferenciam na estrutura de controle da busca, como tipos de proibição e período tabu. O trabalho demonstrou que a utilização de busca tabu para solucionar problemas de roteirização com janelas de tempo permite obter bons resultados.

Bonasser (2005) estudou o problema de roteirização de veículos com múltiplos depósitos e frota heterogênea fixa. Sua estratégia de solução se baseia em heurísticas de economias, busca tabu, colônia de formigas e uma metaheurística híbrida (método *Routing AnTS*), desenvolvida especialmente para a solução do problema em questão. Alguns resultados obtidos por Bonasser (2005) superam os encontrados na literatura.

Abrahão (2006) atacou o problema de programação de manutenção preventiva de veículos, um tipo de problema considerado bastante complexo, em que poucos métodos de solução são estudados para otimizar este tipo de programação. O autor propõe uma estratégia de solução baseada na metaheurística de colônia de formigas, combinada com mecanismos de intensificação e busca local.

Belfiore (2006) estudou a implementação da metaheurística *scatter search* (ou procura em dispersão) em um problema real de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas. No problema de roteirização de veículos com entregas fracionadas, cada cliente pode ser abastecido por mais de um veículo. Os modelos foram aplicados em um dos maiores grupos varejistas brasileiros, que abastece 519 clientes distribuídos em 12 estados brasileiros. Os resultados mostraram melhorias para a empresa, reduzindo em até 8% o custo total da operação.

Wu (2007) abordou o problema de roteirização periódica de veículos, que pode ser considerado como uma generalização do problema clássico de roteirização devido a duas características próprias: um período de planejamento maior que um dia, em que os veículos fazem diversas rotas, e freqüências de visitas associadas a pontos a serem servidos. Foram propostos três procedimentos diferentes para a alocação dos clientes aos dias de visitas: uma heurística de inserção seqüencial que visa equilibrar os esforços dos diferentes dias do período de planejamento, uma heurística de inserção através de um sorteio aleatório que tem seus fundamentos no GRASP e uma heurística baseada em algoritmos genéticos.

2.5 Considerações finais do capítulo

A publicação do trabalho de Prins (2004) com algoritmos genéticos sem delimitadores de rotas e com particionamento eficiente do cromossomo inseriu esta heurística entra as que produzem os melhores resultados para problemas de roteirização de veículos sem restrições temporais

O fato de o algoritmo genético Berger e Barkaoui (2003) também ter apresentado bons resultados indica a importância, nesta metaheurística, da aplicação uma intensa busca local em problemas de roteirização de veículos sem janelas de tempo.

3 CARACTERIZAÇÃO DO PROBLEMA

Este capítulo descreve e formula matematicamente o problema de roteirização de veículos tratado por este trabalho.

3.1 Descrição do problema

Os algoritmos gerados neste trabalho têm as características listadas a seguir, segundo a classificação de Bodin et al. (1993). Tais características são requeridas para a utilização das instâncias de Christofides et al. (1979), nas quais eles serão aplicados:

- Tamanho da frota disponível: infinita
- Tipo de frota disponível: homogênea
- Localização da frota disponível: uma única garagem
- Natureza da demanda: demanda determinística, sem permissão de falta
- Localização das demandas: nos nós
- Restrições temporais da demanda: não existentes
- Rede subjacente: rede euclidiana
- Restrições de capacidade do veículo: iguais para todos os veículos
- Restrições de duração máxima da rota: iguais para todas as rotas
- Operação: somente entregas (ou somente coletas)
- Custos: somente custos variáveis em função do tempo de rota e de carga/descarga
- Objetivo: minimizar total de custos variáveis

As hipóteses relacionadas acima caracterizam um problema de roteirização básico. No entanto, os algoritmos elaborados neste trabalho são flexíveis o suficiente para serem aplicados em problemas em que a função objetivo seja reduzir o número de veículos ou em situações em que os veículos possuam custos fixos e variáveis.

Dentre as características listadas, é importante dizer que o custo das rotas varia em função do tempo em que os veículos estão em rota, tempo este que é a soma do tempo consumido entre os pontos servidos e o tempo de carga, descarga ou espera.

O tempo consumido nas rotas é proporcional à distância entre os pontos, cujas localizações são dadas em coordenadas euclidianas. Esta característica do problema pode gerar dificuldades em sua aplicação em problemas reais, já que a distância entre dois pontos depende da malha rodoviária, e não simplesmente de suas coordenadas euclidianas.

No entanto, obter a distância real entre dois pontos em problemas de roteirização nos quais os pontos de entrega variam diariamente (por exemplo, os clientes de uma loja on-line) é uma tarefa muito difícil. Assim, como afirma Cunha (2006), alguns softwares de roteirização permitem o cadastramento de barreiras geográficas, através de linhas ou polígonos, representando rios, lagos, parques e montanhas por onde os veículos não possam transitar.

3.2 Formulação matemática

A partir das hipóteses e características descritas na seção anterior, o problema é aqui formulado matematicamente.

Seja o grafo $G = \{V, A\}$, onde $V = (v_0, v_1, ..., v_n)$ é um conjunto de *n* pontos e $A = ((v_i, v_j) : i.j)$ é um conjunto de arcos entre os vértices em V.

O ponto v_0 é o depósito onde estão os veículos. Os pontos v_i, $1 \leq i \leq n$ são os pontos a serem atendidos, cada um com uma demanda q_i.

A cada arco (v_i, v_j) $(i, j \in V)$, está associado um valor não-negativo c_{ij}, que representa o tempo de viagem (ou custo) entre os pontos de entrega.

As restrições dos veículos da frota (homogênea) são as seguintes:

- capacidade máxima é dada por Q
- tempo máximo de rota é dado por D

O tempo consumido na viagem entre os pontos v_i e v_j é dado por t_{ij} e o tempo de parada do veículo *k* no ponto v_i é t_{ki}.

A função objetivo a ser minimizada neste problema de roteirização de veículos é o custo total das rotas:

Minimizar $\sum_i \sum_j \sum_k c_{ij} x_{ijk}$

onde x_{ijk} é uma variável binária que, se igual a 1 indica que o veículo *k* viaja do ponto *i* ao ponto *j*.

As restrições do problema são as seguintes:

$\sum_i \sum_k x_{ijk} = 1$, para todo j (1)

$\sum_j \sum_k x_{ijk} = 1$, para todo i (2)

$\sum_i x_{ipk} - \sum_j x_{pjk} = 0$, para todo p, k (3)

$\sum_i q_i \left(\sum_j x_{ijk} \right) \leq Q$, para todo k (4)

$\sum_i t_{ik} \left(\sum_j x_{ijk} \right) + \sum_i \sum_j t_{ij} x_{ijk} \leq D$, para todo j (5)

$\sum_{j=1}^{n} x_{0jk} \leq 1$, para todo i (6)

$\sum_{i=1}^{n} x_{i0k} \leq 1$, para todo j (7)

$x_{ijk} \in (0,1)$ (8)

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \ \forall \ S \subseteq \{2,....,n\}, \text{ para todo k} \qquad (9)$$

As restrições 1 e 2 garantem que um, e apenas um veículo, fará a viagem entre dois pontos.

A restrição 3 assegura que o veículo que atende um ponto sairá deste para fazer outra viagem.

As restrições 4 e 5 não permitem que a capacidade do veículo e o tempo de rota sejam excedidos.

As restrições 6 e 7 garantem que cada veículo só sairá do depósito e retornará a ele uma única vez.

A restrição 8 garante que as variáveis x_{ijk} sejam binárias.

As condições de continuidade e de realização de uma única rota não garantem por si só que o conjunto de soluções represente rotas válidas.

Soluções inválidas são aquelas que foram as chamadas sub-rotas (*subtours*) e a impossibilidade de ocorrência de sub-rotas não contendo o depósito é assegurada pela restrição 9, sendo *S* qualquer subconjunto de pontos alocados a um veículo, excluindo-se o depósito, que não se repetem e que fazem parte de um mesmo roteiro. O número máximo de arcos que podem existir nesse roteiro não pode ser maior que o número de pontos menos uma unidade, evitando, assim, fechar o ciclo entre os pontos.

Outras opções para formação do conjunto *S* podem ser pesquisadas em Bodin et al. (1983).

4 ESTRATÉGIA DE SOLUÇÃO

Neste capítulo será detalhada a estratégia de solução proposta neste trabalho para o problema de roteirização de veículos, que é baseada no algoritmo genético proposto por Prins (2004).

Inicialmente, buscou-se implementar o algoritmo genético exatamente como proposto por Prins (2004), conforme descrito no artigo que o descreve. A partir desta versão, denominada APRINS, foram propostas algumas alterações no sentido de aprimorar seu desempenho.

O algoritmo APRINS está descrito na seção 4.1, enquanto que na seção 4.2 estão detalhados as variantes do APRINS propostas neste trabalho com a finalidade de alcançar melhores resultados.

Tanto o algoritmo APRINS como os propostos foram testados com as instâncias de Christofides et al. (1979), também utilizada por Prins (2004). A avaliação dos algoritmos é dada pelos critérios de qualidade da solução e custo computacional e está detalhada no capítulo 5.

4.1 Algoritmo APRINS

O algoritmo APRINS segue o esquema básico de funcionamento do algoritmo genético conforme ilustrado na Figura 4.1.

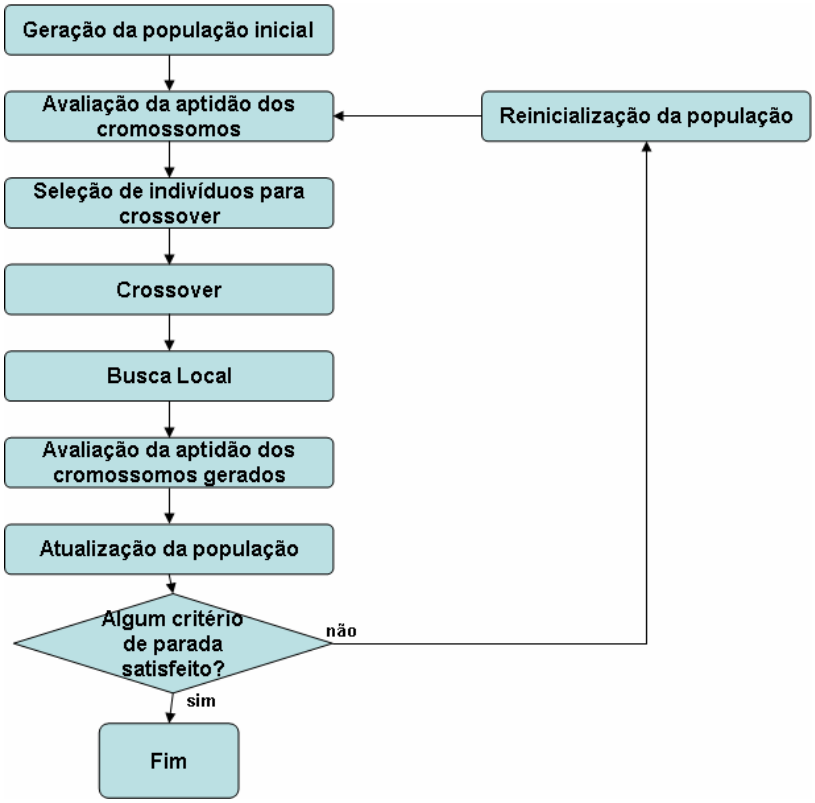


Figura 4.1: Esquema de funcionamento do algoritmo genético híbrido de Prins (2004)

Cada uma das etapas mostradas acima é detalhada nas subseções a seguir.

4.1.1 Geração da população inicial

A escolha de cromossomos de qualidade na solução inicial é importante para permitir que o algoritmo genético normalmente leva o algoritmo a convergir mais rapidamente para uma boa solução (Grefenstette, 1987).

Uma boa população inicial deve ter cromossomos com boas aptidões e ao mesmo tempo bastante diversificados. Isto porque cromossomos ruins podem levar o algoritmo a um elevado custo computacional sem encontrar uma boa solução, enquanto que cromossomos muito semelhantes tendem a fazer com que o algoritmo genético convirja prematuramente para uma solução ótima local.

O algoritmo APRINS, conforme proposto por Prins (2004) é constituído de uma população inicial com 30 indivíduos, sendo que:
- uma é gerada pela heurística de varredura de Gillet e Miller (1974), ou GM;
- uma é gerada pelo algoritmo de Mole e Jameson (1976), ou MJ;
- uma é gerada pela heurística de economias de Clarke e Wright (1964), seguido da heurística 3- opt, ou CW.
- 27 são geradas aleatoriamente

Tanto nas soluções iniciais quando no processamento do algoritmo, caso seja gerado um cromossomo clone de outro que já esteja na solução, ele é descartado e é gerado um outro cromossomo aleatoriamente.

Uma maneira de se identificar se dois cromossomos são considerados clones (ou idênticos) é verificar se a diferença, em termos absolutos,entre seus custos é menor do que uma constante Δ=0,5. Segundo Prins (2004), essa maneira simplificada de detecção de clones, que não é precisa, pois pode haver duas soluções distintas com o mesmo custo, justifica-se pelo fato de que uma detecção através de uma método de comparação mais preciso acarreta um custo computacional muito alto para pouco benefício.

4.1.2 Avaliação da aptidão dos cromossomos

No algoritmo APRINS, a aptidão do cromossomo corresponde ao custo total das rotas que ele representa. O custo total é composto pelo tempo de viagem dos veículos (função da distância percorrida) e, em algumas instâncias-teste, pelo tempo total de serviço (ou atendimento) do veículo, dado pela soma de todos os tempos de atendimento.

O cálculo da aptidão do cromossomo sem delimitadores de rotas é feito a partir do método de particionamento do cromossomo, que é o mesmo processo que gera o roteiro ótimo do cromossomo respeitando, nas rotas formadas, a seqüência de pontos do cromossomo. Conforme explicado anteriormente na seção 2.3.3, para particionar o cromossomo, Prins (2004) aplica o algoritmo de particionamento simples do roteiro gigante.

4.1.3 Seleção dos indivíduos (cromossomos) para o *crossover*

Em APRINS, os pais submetidos ao *crossover* são selecionados através do método do torneio binário. Neste método, são selecionados aleatoriamente dois cromossomos da solução vigente. Aquele que apresentar melhor aptidão é designado pai. Para gerar os dois pais do *crossover*, o processo de seleção é aplicado duas vezes.

4.1.4 *Crossover*

O operador de *crossover* aplicado é o *crossover* OX, conforme explicado na seção 2.3.2.

4.1.5 Busca local

Uma busca local com a finalidade de melhorar cada um dos novos indivíduos gerados de uma população não faz parte dos componentes do de algoritmo genético, conforme proposto por Holland (1975). Existe um operador de mutação, cuja finalidade é manter a variedade genética das populações geradas (Cunha, 2006). A idéia é que cada gene dos descendentes gerados pode sofrer mutação com uma pequena probabilidade, de modo a evitar que o processo de busca torne-se puramente aleatório. Busca-se, dessa forma, recuperar boas características eventualmente perdidas nos processos de seleção e cruzamento, e também evitar

ficar presos em determinadas regiões do espaço de busca, ou em pontos de ótimo locais.

Prins (2004), no entanto, relata ter obtido bons resultados substituindo o operador de mutação por uma busca local composta de trocas do tipo 2-*opt* e 3-*opt*,. Devido a esta substituição, o algoritmo de Prins (2004) pode ser classificado com algoritmo genético híbrido ou algoritmo memético (Moscato, 1989), embora nem todos os autores façam essa distinção de nomenclatura (Beasley e Chu, 1996; Chu e Beasley, 1997).

Dado que o trabalho de Berger e Barkaoui (2003) - um algoritmo genético em que o operador de mutação também foi substituído por uma intensa busca local - obteve excelentes resultados, parece realmente ser produtiva essa troca.

No algoritmo APRINS, para decidir se haverá busca local na iteração em andamento, é gerado um número aleatório entre 0 e 1. Se o número gerado for menor do que uma probabilidade p_m é realizado o processo de busca local. Na fase inicial, $p_m = 0{,}05$. Nas reinicializações, $p_m = 0{,}10$. Identifica-se o filho resultado da busca local com a letra M.
Na busca local, para cada par de pontos (*u,v*) selecionado, são testados os seguintes movimentos, em que *x* e *y* são sucessores de *u* e *v* em suas respectivas rotas e *T(u)* é a rota que visita *u*:

> Movimento M1. Se *u* é um nó ponto de demanda, remover *u* de *T(u)* e inseri-lo após *v* em *T(v).*
>
> Movimento M2. Se *u* e *x* forem pontos de entrega, remover ambos e inserir *(u,x)* após *v* em *T(v)*.
>
> Movimento M3. Se *u* e *x* forem pontos de entrega, remover ambos e inserir *(x,u)* após *v* em *T(v).*
>
> Movimento M4. Se *u* e *v* forem pontos de entrega, trocar as posições de *u* e *v*.

Movimento M5. Se *u*, *x* e *v* forem pontos de entrega, trocar as posições de (*u*, *x*) e *v*.

Movimento M6. Se *u*, x, *v* e *y* forem pontos de entrega, trocar as posições de (*u, x*) e (*v,y*).

Movimento M7. Se *T(u) = T(v),* substituir *(u,x)* e (*v,y)* por *(u,v)* e *(x,y).*

Movimento M8. Se *T(u) ≠ T(v)*, substituir (*u,x)* e *(v,y)* por *(u,v)* e *(x,y).*

Movimento M9. Se *T(u) ≠ T(v)*, substituir (*u,x*) e (*v,y*) por (*u,y*) e (*x,y*).

O movimento M7 corresponde ao método 2-opt tradicional e os movimentos M8 e M9 aplicam o 2-opt em rotas diferentes.

Para cada movimento Mx efetuado, o custo das rotas geradas pelo cromossomo é calculado, sempre após a execução do método de particionamento do roteiro gigante. Caso o custo tenha sido reduzido, fixa-se o novo cromossomo (com as alterações propostas no movimento) e o processo de busca local é reiniciado, ou seja, *u=0* e *v=1*.

Cada iteração de busca local só termina quando um cromossmo (original ou já alterado) tem todos os movimentos examinados para todos os pares, sem que ocorra nenhuma melhora.

A busca local é feita entre todos os pares de pontos de demanda, o que resulta no elevado custo computacional desta etapa. Assim, em seu trabalho, Prins (2004), afirmou que há espaços para evolução na eficiência da etapa de busca local, que consome mais de 95% do tempo total do processamento do algoritmo.

Como regra geral, quanto maior a vizinhança, melhores são os resultados obtidos, mas a um custo computacional maior. Uma vizinhança mais restrita produz soluções inferiores a um custo computacional menor.

Definir uma vizinhança que gere boas soluções a custos computacionais melhores é o grande desafio em qualquer tipo de busca local.

4.1.6 Atualização da população

Conforme explicado anteriormente na seção 2.3.2, o *crossover* OX gera dois filhos (F1 e F2), sendo que um deles é selecionado aleatoriamente para a seqüência do processo. Este, por sua vez, tem uma probabilidade p_m de entrar no processo de busca local. Caso seja aplicado no cromossomo o processo de busca local e o cromossomo alterado não for clone, ele é inserido na população. Caso o cromossomo alterado já exista na população (ou seja, é clone), a busca local é desconsiderada e a análise volta para o cromossomo original (filho F1).

Se o cromossomo original não passar pela busca local e não for clone, ele é inserido na população. Caso contrário, a população não muda e a iteração é considerada não produtiva.

Caso uma iteração não gere filho para entrar na população, ela é considerada uma iteração não produtiva. Caso contrário, o filho entrará na população substituindo um indivíduo selecionado aleatoriamente entre os indivíduos da metade de baixo do ranking de aptidão. Este método de gerenciamento incremental da população permite que as melhores soluções continuem na população e faz com que uma boa criança a se reproduza rapidamente.

Para facilitar o entendimento desta etapa, o fluxo da Figura 4.2 mostra como se dá a inserção na população do cromossomo gerado no *crossover* segundo o algoritmo APRINS:

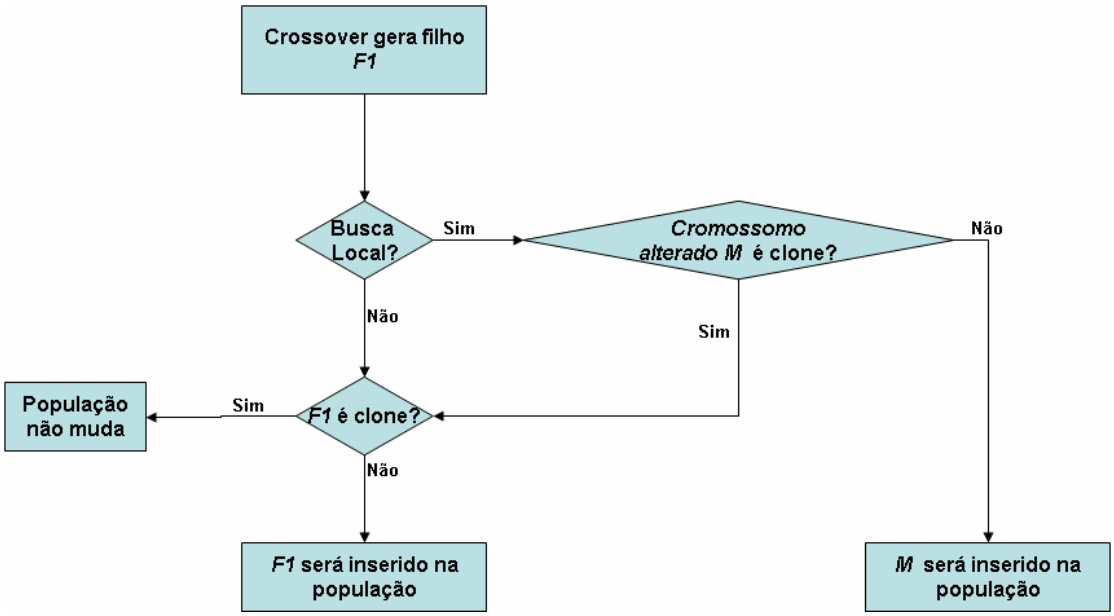


Figura 4.2: Fluxo de inserção de cromossomo na população

4.1.7 Reinicialização

O loop do algoritmo é executado 11 vezes. A fase principal, mais longa, é seguida de 10 reinicializações mais rápidas.

O algoritmo é reinicializado quando:

- atinge $\alpha_{\max}$ iterações produtivas (isto é, iterações que gerem novos indivíduos e que não resultem em clones) ou

- atinge $\beta_{\max}$ iterações sem melhoria da melhor solução encontrada.

Na fase principal, Prins (2004) consderou $\alpha_{\max} = 30000$ e $\beta_{\max} = 10000$. Nas reinicializações, $\alpha_{\max} = 2000$ e $\beta_{\max} = 2000$.

Quando o critério de parada é satisfeito, o algoritmo poderá ser reinicializado. Ele só não será reinicializado caso o número máximo de reinicializações tenha sido atingido.

O objetivo da reinicialização é introduzir cromossomos diferentes na população, a fim de restaurar a diversidade genética da população, buscando-se obter, assim, a uma solução ainda melhor.

No algoritmo APRINS, para definir os cromossomos que entram na solução, é criada uma população Ω de $\rho = 8$ cromossomos, gerados aleatoriamente e bem espaçados. Cada cromossomo ρ_k é comparado com o pior cromossomo da população vigente Π. Se sua aptidão for superior, ele entra na população Π substituindo seu pior cromossomo. Caso contrário, é feito um *crossover* entre ρ_k e cada um dos cromossomos da população $\Pi \cup \Omega$. Se o melhor filho obtido nesses *crossover*s for melhor do que o pior filho da população Π, ocorre a substituição.

Na prática, segundo Prins (2004), diversas populações Ω devem ser geradas para se obter ρ cromossomos. Limitou-se em 5 tentativas a obtenção dos cromossomos, mesmo que a quantidade ρ não tenha sido atingida.

Segundo Prins (2004), comparado com uma substituição cega e total, esta reposição parcial preserva as melhores soluções e não piora o pior custo.

4.2 Algoritmos propostos

4.2.1 Alterações propostas na geração da população inicial

Com o intuito de aumentar a quantidade de soluções viáveis e de boa qualidade na população inicial, são propostas três variantes do algoritmo básico APRINS em que são geradas soluções aleatórias iniciais do que o algoritmo APRINS, conforme indicado na Tabela 4.1.

Tabela 4.1: Algoritmos com alterações na geração da população inicial

algoritmo	CW	GM	MJ	Aleatórias
AINI05	2	2	1	25
AINI10	5	4	1	20
AINI15	7	7	1	15

A seguir será explicada a maneira em que são geradas mais de uma solução inicial distinta utilizando as heurísticas de Clarke e Wright (1964) e Gillet e Miller (1974). A heurística de Mole e Jameson (1976) não será descrita, pois foi implementada sem alterações e gera apenas uma solução inicial viável.

Heurística de economias de Clarke e Wright

A heurística de economias de Clarke e Wright (1964) em sua forma básica cria apenas uma solução para um problema. Para que ele possa gerar diversas soluções, como proposto neste item, será inserido um parâmetro λ na função que calcula a economia que a criação de uma viagem entre os pontos *i-j* traz para o problema.

Assim, a função que calcula a economia s_{ij} referente à inclusão da rota entre os pontos *i-j* incorpora um fator de forma γ e é dada pela seguinte expressão:

$$s_{ij} = c_{iO} + c_{Oj} - \gamma * c_{ij}$$

onde:

c_{ij} é o custo da rota entre os pontos i, j

O é a origem (garagem)

$0.5 < \gamma < 2$, com γ sendo determinado de maneira aleatória

O intervalo de valores que γ pode assumir foi definido a partir de testes preliminares.

De resto, o algoritmo de Clarke e Wright (1964) utilizado neste trabalho será idêntico ao básico:

1. Calcular as economias s_{ij} para todos os pares i, j $(i \neq j, i \neq O, j \neq O)$
2. Classificar em ordem decrescente as economias s_{ij}
3. Para todo par i, j, na ordem definida no item 3, verificar se i e j estão ou não incluídos em um roteiro já existente.
   i. Se i e j não estiverem incluídos em nenhum dos roteiros já abertos, então, se as restrições (distância percorrida pelo veículo, capacidade do veículo) forem respeitadas, criar um novo roteiro com os nós i e j.
   ii. Se exatamente um dos pontos i ou j já pertence a um roteiro pré-estabelecido, verificar se esse ponto é o primeiro ou o último do roteiro. Em caso afirmativo, se todas as restrições forem respeitadas, acrescentar o arco i, j a esse roteiro. Caso contrário, ir para o próximo par.
   iii. Se os nós i e j já pertencerem a dois roteiros (diferentes) pré-estabelecidos, verificar se ambos são extremos dos respectivos roteiros. Se verdadeiro e se as restrições forem respeitadas, fundir os dois roteiros em um só. Caso contrário, ir para o próximo par.
   iv. Se ambos os nós i e j pertencerem a um mesmo roteiro, ir para o próximo par.
4. Caso algum ponto não tenha incluído em nenhuma rota, são formados roteiros individualizados, ligando o depósito a cada ponto e retornando à base.

Heurística de Varredura de Gillet e Miller

Para que o algoritmo da varredura de Gillet e Miller (1974) possa gerar soluções diferentes, neste trabalho o ângulo α a partir do qual a varredura de pontos é iniciada será definido de maneira aleatória, entre 0 e 2π, tendo como referência o depósito.

De resto, o algoritmo de varredura é mantido tal qual o proposto por Gillet e Miller (1974):

1. Define-se a coordenada polar em que se iniciará a varredura.
2. Seleciona-se um ponto *i*, o primeiro ponto não alocado cuja coordenada polar é maior ou igual à definida no item 1. Caso todos os pontos estejam alocados a alguma rota, o algoritmo é encerrado.
3. Se *i* respeita as restrições de capacidade/distância percorrida do veículo, ele será inserido na rota *r* .
4. Seleciona-se ponto *j*, o primeiro ponto não alocado cuja coordenada polar é maior ou igual à definida no item 1.
5. Se a inserção de *j* na rota *r* respeita a restrição de capacidade do veículo, é testado o ponto da rota em que sua inserção resulta na rota mais econômica. Se a inserção de *j* não respeita a restrição de capacidade do veículo, a rota *r* é finalizada e inicia-se uma nova rota, voltando-se ao passo 2.
6. Se a inserção de *j* no ponto selecionado em 5 respeita a restrição de distância percorrida pelo veículo, ele é inserido na rota *r*. Caso contrário, a rota *r* é finalizada e inicia-se uma nova rota, voltando-se ao passo 2.

4.2.2 Alterações propostas no particionamento do cromossomo

O fato de um cromossomo ter uma boa aptidão, ou seja, de gerar boas rotas, significa que a ordem de visita dos pontos que o compõem é de boa qualidade.

Partindo desta premissa, é possível que, mantida a ordem do cromossomo, mas iniciando seu particionamento de um outro ponto, outras soluções de boa qualidade - possivelmente melhores - possam ser obtidas. O mesmo pode ocorrer se também invertermos a ordem dos pontos do cromossomo. São essas 2 alterações que são propostas relacionadas ao particionamento do cromossomo.

O cromossomo codificado por Prins (2004), sem delimitadores de rotas, pode ser tratado como um roteiro gigante, sendo suas rotas geradas pelo método de particionamento simples do roteiro gigante.

Assim, para alterar o ponto em que é iniciado o particionamento do cromossomo, será utilizado o método de particionamento múltiplo do roteiro gigante, descrito anteriormente na seção 2.3.5.

Foram implementados dois tipos diferentes de algoritmos que, ao invés de aplicarem apenas o método de particionamento simples do roteiro gigante (como em APRINS) nos cromossomos gerados, dependendo da qualidade da solução gerada, podem aplicar também o método de particionamento múltiplo do roteiro gigante. São eles:

→ APAM03, em que o método de particionamento múltiplo será aplicado no cromossomo, caso a solução obtida pelo particionamento simples seja uma das 3 melhores classificadas até aquele estágio;

→ APIN03, em que o método de particionamento múltiplo será aplicado duas vezes: no cromossomo em sua ordem normal e no cromossomo invertido, caso a solução obtida pelo particionamento simples seja uma das 3 melhores classificadas.

4.2.3 Alterações propostas na seleção do filho gerado no *crossover*

O presente trabalho propõe dois algoritmos, com alterações na maneira em que o filho resultado do *crossover* é escolhido:

→ ACRF1, que se baseia na afirmação de Prins (2004) de que a melhor maneira de se escolher o cromossomo que entrará na solução é aleatoriamente, ou seja, é indiferente se o cromossomo escolhido é F1 ou F2. Assim, para reduzir o custo computacional do algoritmo, é gerado apenas o filho F1. Espera-se que, gerando apenas um cromossomo, haja ganho no custo computacional do algoritmo sem prejuízo para a qualidade da solução.

→ ACRME, em que se seleciona o melhor entre os filhos F1 e F2. Espera-se com isso melhorias na qualidade das soluções.

4.2.4 Alterações propostas na busca local

A seguir, serão apresentados 2 algoritmos propostos em que a etapa de busca local é alterada, com o objetivo de reduzir o custo computacional do algoritmo, sem prejuízo para as soluções obtidas.

→ ABLAM: Este algoritmo se baseia na hipótese de que pode haver alguma relação entre a qualidade do cromossomo que entra no processo de busca local e a qualidade do cromossomo que dela resulta. A idéia do algoritmo ABLAM é estabelecer uma aptidão mínima (ou seja, um custo máximo) a partir da qual o cromossomo estivesse habilitado a participar do processo de busca local. O valor desta aptidão mínima será obtido pela multiplicação entre um fator z =1,20 (estabelecido a partir de testes de calibração preliminares) e a melhor aptidão.

→ ABLRE: A premissa deste algoritmo é de que a probabilidade de se melhorar uma solução trocando pares de pontos de lugar (processo que é realizado nos 9 tipos de movimento) é função da distância entre eles. Quanto maior é a distância entre dois pontos, menor é a chance de a solução ser melhorada. Esta idéia foi adotada com sucesso na heurística busca tabu granular, proposta por Toth e Vigo (2003) para problemas de roteirização de veículos. Nela, os arcos de maior custo - aqueles que ultrapassarem um limite de “granularidade” - são eliminados do grafo associado ao problema, já que a probabilidade de pertencerem a uma solução de boa qualidade é baixa. No entanto, a aplicação desta premissa requer algumas precauções, pois pode trazer prejuízos ao algoritmo, fazendo com que ele convirja para uma solução ótima local.

Para reduzir o tamanho da vizinhança na etapa de busca local do algoritmo de Prins (2004) é proposto um método em que o tamanho da vizinhança é diferente em cada

estágio da busca. No início, para aumentar a exploração do espaço de soluções, a vizinhança é maior, e à medida que a busca avança, ela vai se restringindo.

Os pares de pontos *i, j* que podem ser trocados na busca local são aqueles cuja distância d_{ij} é menor do que $\alpha * d_{\max}$, onde:

- $d_{\max}$ é a distância máxima entre dois pontos do grafo
- $0 < \alpha \leq 1$ é o fator de redução da vizinhança.

São propostos 10 valores de α diferentes, cada um deles sendo utilizado por um mesmo número de iterações $n = \frac{N}{10}$, onde N é o total de iterações do algoritmo original. No início, $\alpha = 1$ (exploração máxima) e se reduz à fração de um décimo.

4.2.5 Alterações propostas na reinicialização do algoritmo

Para garantir que a população reinicializada seja diversificada e também para garantir que soluções de qualidade sejam inseridas na população, o presente trabalho propõe dois algoritmos com alterações na reinicialização. A diversificação é garantida com a entrada de cromossomos gerados aleatoriamente, e a inclusão de novas soluções de qualidade será garantida a partir das heurísticas CW e MJ, descritas na seção 4.2.1. Os melhores cromossomos da população vigente (soluções de elite) são mantidos na população reinicializada.

Os algoritmos propostos são os seguintes:

→ AREI10, em que a população reinicializada será composta por cromossomos gerados de 3 diferentes maneiras, sendo: 10 cromossomos de elite, 10 cromossomos aleatórios e 10 cromossomos gerados a partir de heurísticas, sendo 5 CW (com $0.5 < \lambda < 2$) e 5 GM (α aleatório).

→ AREI20, em que a população reinicializada será composta por cromossomos gerados de 3 diferentes maneiras, sendo: 20 cromossomos de elite, 6

cromossomos aleatórios e 4 cromossomos gerados a partir de heurísticas, sendo 2 CW (com $0.5 < \lambda < 2$) e 2 GM (α aleatório).

4.3 Considerações finais do capítulo

Neste capítulo foram apresentados o algoritmo genético desenvolvido por Prins (2004) e algoritmos com alterações em suas diversas etapas.

A Tabela 4.2 resume os algoritmos propostos neste trabalho, que serão avaliados no próximo capítulo:

Tabela 4.2: Algoritmos propostos

Algoritmo	Tipo de melhoria proposta	Objetivo
APRINS	Algoritmo básico	-
AINI05	Inicialização	Qualidade da Solução
AINI10	Inicialização	Qualidade da Solução
AINI15	Inicialização	Qualidade da Solução
ABLRE	Mutação (busca local)	Tempo de processamento
ABLAM	Mutação (busca local)	Tempo de processamento
APIN03	Particionamento do cromossomo	Qualidade da Solução
APAM03	Particionamento do cromossomo	Qualidade da Solução
AREI10	Reinicialização	Qualidade da Solução
AREI20	Reinicialização	Qualidade da Solução
ACRF1	Seleção do filho no crossover	Tempo de processamento
ACRME	Seleção do filho no crossover	Qualidade da Solução

5 EXPERIMENTOS COMPUTACIONAIS

Neste capítulo são avaliados os algoritmos propostos no capítulo 4 para resolver o problema de roteirização de veículos.

A seção 5.1 descreve os problemas nos quais os algoritmos são aplicados. Na seção 5.2 são apresentados e analisados os resultados dos algoritmos, sempre os comparando com o algoritmo original APRINS. A seção 5.3 propõe e analisa os resultados do novo algoritmo AMELHO, que terá características combinadas dos algoritmos melhores avaliados na seção 5.2. As análises e considerações finais do capítulo estão na seção 5.4.

5.1 Instâncias-teste de Christofides

A fim de se avaliar o desempenho dos algoritmos propostos, eles serão testados nas instâncias de Christofides et al. (1979), detalhadas na Tabela 5.1.

Tabela 5.1**:** Instâncias de Christofides et al. (1979)

Instância	Número de Pontos	Melhor solução conhecida	Capacidade do Veículo	Custo máximo do veículo	Custo de carga
vrp1	50	524,61	160	999999	0
vrp2	75	835,26	140	999999	0
vrp3	100	826,14	200	999999	0
vrp4	150	1028,42	200	999999	0
vrp5	199	1291,45	200	999999	0
vrp6	50	555,43	160	200	10
vrp7	75	909,68	140	160	10
vrp8	100	865,94	200	230	10
vrp9	150	1162,55	200	200	10
vrp10	199	1395,85	200	200	10
vrp11	120	1042,11	200	999999	0
vrp12	100	819,56	200	999999	0
vrp13	120	1541,14	200	720	50
vrp14	100	866,37	200	1040	90

O custo dos roteiros é variável em função da distância percorrida pelo veículo e do tempo de carga do veículo nos pontos atendidos, que se aplica nas instâncias vrp6, vrp7, vrp8, vrp9, vrp10, vrp13 e vrp14.

As melhores soluções conhecidas para as instâncias vrp1 e vrp12 são comprovadamente as soluções ótimas destas (Prins, 2004).

Os algoritmos foram implementados em linguagem C no Microsoft Visual Studio 2005. Os testes foram realizados em um PC Athlon 64 3200+ equipado com 512 MB de memória RAM.

5.2 Avaliação dos algoritmos propostos

Nesta seção são apresentados e analisados os resultados dos algoritmos propostos neste trabalho. A comparação é feita entre os algoritmos que propõem alterações na mesma etapa, sempre comparados com o algoritmo original APRINS.

Para cada algoritmo, cada instância-teste será executada 5 vezes e o resultado apresentado dependerá do objetivo para o qual o algoritmo proposto fora construído:

- se o objetivo do algoritmo proposto for o de melhorar a qualidade da solução, o resultado apresentado para a instância será aquele que gerar a melhor solução em termos de qualidade. Isto porque, como os algoritmos gerados neste trabalho dependem, em diversos momentos, de números gerados aleatoriamente, os resultados podem variar nos processamentos. Além disso, se aplicado a problemas reais, a orientação seria a de executar o algoritmo mais de uma vez, utilizando o que resultasse na melhor solução entre todos os processamentos.

- se o objetivo do algoritmo proposto for o de reduzir o tempo de processamento, o resultado apresentado para cada instância será a média dos 5 processamentos. Não seria correto utilizar o menor tempo de processamento, pois quando aplicado a problemas reais, a orientação seria de executar o algoritmo mais de uma vez. Logo, o importante em termos de tempo de processamento seria a média de cada execução, e não o tempo da melhor delas.

A comparação entre os algoritmos propostos, já tendo os resultados das diversas instâncias-teste (conforme descrito acima), também dependerá do objetivo para o qual o algoritmo tiver sido desenvolvido.

Caso tenha sido construído para melhorar a qualidade das soluções, o algoritmo é analisado por dois critérios, sendo o segundo utilizado apenas em caso de empate no primeiro item:

1. A quantidade de vezes em que o algoritmo é o melhor para as 14 instâncias (se para uma mesma instância dois algoritmos tiverem o mesmo resultado, ambos receberão 1 ponto);
2. A soma do custo total das 14 instâncias.

Caso o objetivo do algoritmo seja o de reduzir o custo computacional, ele será medido através dos seguintes critérios, pela ordem:

1. A quantidade de vezes em que o algoritmo é o melhor para cada instância (se para uma mesma instância dois algoritmos tiverem o mesmo resultado, ambos receberão 1 ponto);
2. A soma do tempo de processamento total das 14 instâncias.

Nas tabelas seguintes, estão marcados em verde (para qualidade) e azul (para tempo de processamento) os melhores algoritmos para cada instância-teste e para o total consolidado.

5.2.1 Algoritmos com alterações no particionamento do cromossomo

A Tabela 5.2 mostra os resultados dos algoritmos com alterações no particionamento do cromossomo, comparados com o algoritmo APRINS:

Tabela 5.2: Algoritmos com alterações no particionamento do cromossomo

	APRINS		APIN03		APAM03	
Instância	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)
vrp1	524,9	6,9	524,6	11,5	524,6	8,1
vrp2	840,1	16,5	836,7	24,8	836,8	20,4
vrp3	832,1	33,8	829,7	54,8	826,1	46,0
vrp4	1.056,1	101,2	1.043,2	175,6	1.040,2	146,8
vrp5	1.328,4	207,7	1.325,6	347,5	1.328,7	273,4
vrp6	559,0	7,3	555,4	12,3	556,7	7,8
vrp7	912,5	20,9	910,9	22,6	929,6	22,4
vrp8	875,2	42,5	872,2	53,9	871,1	47,2
vrp9	1.194,1	129,9	1.183,4	171,7	1.208,0	156,4
vrp10	1.434,5	305,7	1.422,4	439,0	1.439,0	283,9
vrp11	1.045,7	80,2	1.045,5	97,0	1.045,7	81,1
vrp12	819,6	25,4	819,6	37,3	819,6	28,7
vrp13	1.555,1	70,5	1.553,0	109,1	1.549,5	98,3
vrp14	866,5	32,6	866,4	41,3	866,4	33,6
TOTAL	**13.844**	**1.081**	**13.789**	**1.598**	**13.842**	**1.254**
MELHOR	**0**	**13**	**8**	**0**	**6**	**1**

Pode-se ver que ambos os algoritmos propostos com alterações no particionamento do cromossomo resultaram em soluções de melhor qualidade, mas o algoritmo APIN03, em que o particionamento múltiplo do cromossomo é feito também no cromossomo em que os genes estão invertidos, se mostrou bem superior, tendo a melhor solução do grupo em 8 das 14 instâncias.

Por executar uma maior quantidade de particionamentos, o algoritmo APIN03 também é que tem o maior tempo de processamento (1.598 no total).

Apesar disso, como o objetivo dos algoritmos propostos nesta seção é o de melhorar a qualidade das soluções geradas, o tipo de particionamento do cromossomo do algoritmo APIN03 é o que será inserido no algoritmo AMELHO.

5.2.2 Algoritmos com alterações na geração da população inicial

A Tabela 5.3 mostra os resultados dos algoritmos com alterações na composição de suas populações iniciais:

Tabela 5.3: Algoritmos com alterações na geração da população inicial

	APRINS		AINI05		AINI10		AINI15	
Instância	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)
vrp1	524,9	6,9	524,6	6,7	524,6	7,2	524,6	18,1
vrp2	840,1	16,5	836,0	17,9	839,4	17,3	837,6	14,5
vrp3	832,1	33,8	832,6	28,1	829,9	32,4	830,7	32,4
vrp4	1.056,1	101,2	1.044,0	82,6	1.049,0	78,5	1.040,7	84,0
vrp5	1.328,4	207,7	1.325,2	205,4	1.329,5	222,3	1.331,6	223,6
vrp6	559,0	7,3	559,0	7,4	555,4	9,1	560,2	7,5
vrp7	912,5	20,9	909,7	25,1	910,8	19,4	924,7	15,2
vrp8	875,2	42,5	869,1	27,9	877,6	33,5	871,3	36,1
vrp9	1.194,1	129,9	1.210,6	85,9	1.183,0	112,2	1.182,0	102,0
vrp10	1.434,5	305,7	1.442,2	211,6	1.428,7	250,9	1.431,6	268,6
vrp11	1.045,7	80,2	1.045,5	76,2	1.045,7	63,6	1.045,5	88,9
vrp12	819,6	25,4	819,6	28,1	819,6	25,0	819,6	23,5
vrp13	1.555,1	70,5	1.554,6	63,4	1.555,1	85,0	1.551,1	72,4
vrp14	866,5	32,6	866,5	31,7	866,5	36,9	866,5	39,2
TOTAL	13.844	1.081	13.839	898	13.815	993	13.818	1.026
MELHOR	1	1	7	8	6	2	5	3

Os três algoritmos propostos com alterações na geração da população inicial obtiveram resultados melhores do que o algoritmo original APRINS, cumprindo seus objetivos de melhorar a qualidade das soluções obtidas.

O melhor algoritmo entre os três é o AINI05, em que, das 30 soluções da população inicial, 5 são geradas pelas heurísticas descritas na seção 4.4.1, e 25 são geradas aleatoriamente. Em seguida, vem o algoritmo AINI10, que contém 10 soluções geradas pelas heurísticas, e por último, está o AINI15, com 15 soluções geradas por tais heurísticas.

Isto demonstra que gerar boas soluções iniciais é importante, mas um excesso de boas soluções de mesma origem (heurísticas CW e GM) reduz a diversidade da população, explorando poucas regiões do espaço de busca.

É importante observar também que o processo de geração de soluções heurísticas eleva o tempo de processamento do algoritmo, como pode ser visto ao se comparar o crescimento do tempo de processamento total do algoritmo AINI05 para o AINI15.

5.2.3 Algoritmos com alterações na seleção do filho gerado no *crossover*

A Tabela 5.4 mostra os resultados dos algoritmos com alterações na seleção do filho gerado no *crossover:*

Tabela 5.4: Algoritmos com alterações na seleção do filho gerado no *crossover*

	APRINS		ACRF1		ACRME	
Instância	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)
vrp1	524,9	6,9	524,6	7,6	524,6	7,8
vrp2	840,1	16,5	838,6	17,7	835,9	17,0
vrp3	832,1	33,8	832,0	36,2	826,4	34,8
vrp4	1.056,1	101,2	1.042,1	78,5	1.048,2	80,3
vrp5	1.328,4	207,7	1.336,0	216,2	1.325,5	180,7
vrp6	559,0	7,3	560,2	6,7	555,4	8,3
vrp7	912,5	20,9	914,1	19,9	928,3	21,5
vrp8	875,2	42,5	874,2	45,3	867,4	41,8
vrp9	1.194,1	129,9	1.205,7	106,6	1.189,4	272,9
vrp10	1.434,5	305,7	1.434,9	259,3	1.441,9	446,6
vrp11	1.045,7	80,2	1.045,5	59,8	1.044,5	91,4
vrp12	819,6	25,4	819,6	25,0	819,6	25,2
vrp13	1.555,1	70,5	1.552,1	71,8	1.551,1	112,4
vrp14	866,5	32,6	866,4	29,9	866,4	30,4
TOTAL	**13.844**	**1.081**	**13.846**	**980**	**13.825**	**1.371**
MELHOR	**2**	**4**	**2**	**8**	**11**	**2**

Tanto o objetivo do algoritmo ACRF1 (reduzir o tempo de processamento do algoritmo APRINS, com a exclusão da etapa de geração de número aleatório para escolher o cromossomo filho), como o objetivo do algoritmo ACRME (melhorar a qualidade das soluções ao selecionar o melhor cromossomo filho entre os filhos F1 e F2), foram atingidos.

O fato de o algoritmo ACRF1 ser melhor do que o APRINS em termos de tempo de processamento, sem prejuízo de qualidade de solução (na comparação entre os dois algoritmos, o primeiro obtem melhores resultados em 9 dos problemas contra 6 do APRINS), mostra que não há vantagem em proceder, no momento da escolha do filho gerado pelo *crossover*, como no algoritmo APRINS, em que é gerado um número aleatório para selecionar o filho.

O algoritmo ACRME, por calcular a aptidão de cada um dos filhos, acaba tendo um custo computacional maior do que os outros dois algoritmos. Mesmo assim, como o objetivo principal de um algoritmo é o de gerar boas soluções, no algoritmo AMELHO, será inserido o tipo de seleção do filho gerado no *crossover* do algoritmo ACRME.

5.2.4 Algoritmos com alterações na busca local

A Tabela 5.5 mostra os resultados dos algoritmos com alterações na busca local:

Tabela 5.5: Algoritmos com alterações na busca local

	APRINS		ABLAM		ABLRE	
Instância	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)
vrp1	524,9	6,9	524,6	6,9	524,9	6,8
vrp2	840,1	16,5	838,4	16,3	840,1	16,1
vrp3	832,1	33,8	834,6	28,0	832,1	36,1
vrp4	1.056,1	101,2	1.054,0	75,8	1.056,1	108,6
vrp5	1.328,4	207,7	1.335,6	150,1	1.336,0	216,2
vrp6	559,0	7,3	556,7	7,9	559,0	7,6
vrp7	912,5	20,9	**910,5**	19,9	**910,8**	21,6
vrp8	875,2	42,5	881,7	39,0	875,2	45,5
vrp9	1.194,1	129,9	1.206,0	125,6	1.205,7	106,6
vrp10	1.434,5	305,7	1.448,1	305,2	1.434,9	259,3
vrp11	1.045,7	80,2	1.045,7	54,9	1.045,5	59,8
vrp12	819,6	25,4	821,2	25,6	819,6	27,1
vrp13	1.555,1	70,5	1.552,4	72,9	1.552,1	71,8
vrp14	866,5	32,6	867,0	28,3	866,5	34,9
TOTAL	**13.844**	**1.081**	**13.876**	**957**	**13.859**	**1.018**
MELHOR	**7**	**3**	**5**	**7**	**6**	**4**

Conforme a Tabela 5.5, ambos os algoritmos propostos com alterações na busca local reduziram o tempo de processamento com relação ao algoritmo APRINS.

Entre os dois algoritmos, o ABLAM (que estabelece uma aptidão mínima a partir da qual o cromossomo está habilitado a participar do processo de busca local) se mostrou mais eficiente, por reduzir o custo computacional em 7 das 14 instâncias-teste.

No entanto, como a qualidade das soluções geradas pelo algoritmo ABLAM são piores do que as do algoritmo original APRINS, ele não será considerado no algoritmo AMELHO.

5.2.5 Algoritmos com alterações na reinicialização do algoritmo

A Tabela 5.6 mostra os resultados dos algoritmos com alterações na etapa de reinicialização do algoritmo:

Tabela 5.6: Algoritmos com alterações na etapa de reinicialização do algoritmo

	APRINS		AREI10		AREI20	
Instância	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)	Custo da solução	Tempo processam. (s)
vrp1	524,9	6,9	524,6	7,3	524,6	7,2
vrp2	840,1	16,5	836,8	17,0	840,1	16,1
vrp3	832,1	33,8	831,9	34,6	832,1	35,2
vrp4	1.056,1	101,2	1.052,0	92,4	1.047,8	107,9
vrp5	1.328,4	207,7	1.326,2	203,4	1.328,2	202,9
vrp6	559,0	7,3	556,7	7,7	556,7	6,7
vrp7	912,5	20,9	912,5	17,7	912,5	15,4
vrp8	875,2	42,5	870,3	46,4	870,7	59,7
vrp9	1.194,1	129,9	1.215,2	108,6	1.209,7	138,1
vrp10	1.434,5	305,7	1.443,4	295,2	1.443,7	375,8
vrp11	1.045,7	80,2	1.044,6	117,6	1.043,7	185,0
vrp12	819,6	25,4	819,6	25,6	819,6	24,4
vrp13	1.555,1	70,5	1.555,0	96,7	1.556,7	167,3
vrp14	866,5	32,6	866,5	41,2	866,5	40,0
TOTAL	**13.844**	**1.081**	**13.855**	**1.111**	**13.852**	**1.382**
MELHOR	**4**	**6**	**10**	**3**	**7**	**5**

A reinicialização do algoritmo com a inserção de cromossomos gerados pelas heurísticas adaptadas de Clarke e Wright (1964) e de Gillet e Miller (1974) gerou resultados positivos, melhorando a qualidade das soluções finais das instâncias-teste.

Possivelmente por inserir mais soluções novas na população reinicializada, aumentando a diversidade da população, o algoritmo AREI10 (mantém apenas 10 soluções na população vigente) mostrou-se superior ao AREI20 (mantém 20 soluções). Assim, a reinicialização existente no algoritmo AREI10 estará no algoritmo AMELHO.

Como era de se esperar, o custo computacional aumentou nos dois algoritmos propostos.

5.3 Algoritmo AMELHO

A construção do algoritmo AMELHO tem como objetivo criar um algoritmo que, combinando as características dos algoritmos propostos na seção 5.2, seja melhor do que todos eles.

De acordo com os resultados da seção 5.2, os algoritmos cujas características irão compor o algoritmo AMELHO são os seguintes:

Tabela 5.7: Algoritmos que irão compor algoritmo AMELHO

Tipo de melhoria proposta	Algoritmo
Seleção do filho no crossover	ACRME
Inicialização	AINI05
Reinicialização	AREI10
Particionamento do cromossomo	APIN03

Executados os 5 processamentos do algoritmo AMELHO para cada uma das 14 instâncias-teste de Christofides et al. (1979), verificou-se que o algoritmo AMELHO tem a melhor solução em 9 delas, sendo assim o melhor entre os algoritmos propostos.

A tabela 5.8 mostra a quantidade de vezes que cada um dos algoritmos propostos no trabalho obteve o melhor resultado para as instâncias-teste.

Tabela 5.8: Quantidade de vezes em que cada algoritmo proposto obteve o melhor resultado

Classificação	Tipo de melhoria proposta	Algoritmo	Qtd de vezes menor custo
1º	Diversas melhorias	AMELHO	9
2º	Seleção do filho no crossover	ACRME	6
3º	Inicialização	AINI05	6
4º	Particionamento do cromossomo	APAM03	6
5º	Particionamento do cromossomo	APIN03	5
6º	Inicialização	AINI10	3
7º	Inicialização	AINI15	3
8º	Seleção do filho no crossover	ACRF1	3
9º	Reinicialização	AREI20	3
10º	Reinicialização	AREI10	2
11º	Algoritmo original	APRINS	1
12º	Mutação (busca local)	ABLRE	1
13º	Mutação (busca local)	ABLAM	1

A tabela 5.9 sumariza os resultados de todos os algoritmos propostos neste trabalho.

Tabela 5.9: Detalhamento dos resultados dos algoritmos propostos

	APRINS		AINI05		AINI10		AINI15		ABLRE		ABLAM		APIN03		APAM03		AREI10		AREI20		ACRF1		ACRME		AMELHO	
Instância	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)	Custo sol.	Tempo (s)
vrp1	524,9	6,9	524,6	6,7	524,6	7,2	524,6	18,1	524,9	6,8	524,6	6,9	524,6	11,5	524,6	8,1	524,6	7,3	524,6	7,2	524,6	7,6	524,6	7,8	524,6	10,4
vrp2	840,1	16,5	836,0	17,9	839,4	17,3	837,6	14,5	840,1	16,1	838,4	16,3	836,7	24,8	836,8	20,4	836,8	17,0	840,1	16,1	838,6	17,7	835,9	17,0	835,9	19,8
vrp3	832,1	33,8	832,6	28,1	829,9	32,4	830,7	32,4	832,1	36,1	834,6	28,0	829,7	54,8	826,1	46,0	831,9	34,6	832,1	35,2	832,0	36,2	826,4	34,8	828,3	47,8
vrp4	1.056,1	101,2	1.044,0	82,6	1.049,0	78,5	1.040,7	84,0	1.056,1	108,6	1.054,0	75,8	1.043,2	175,6	1.040,2	146,8	1.052,0	92,4	1.047,8	107,9	1.042,1	78,5	1.048,2	80,3	1.037,2	115,5
vrp5	1.328,4	207,7	1.325,2	205,4	1.329,5	222,3	1.331,6	223,6	1.336,0	216,2	1.335,6	150,1	1.325,6	347,5	1.328,7	273,4	1.326,2	203,4	1.328,2	202,9	1.336,0	216,2	1.325,5	180,7	1.323,0	231,5
vrp6	559,0	7,3	559,0	7,4	555,4	9,1	560,2	7,5	559,0	7,6	556,7	7,9	555,4	12,3	556,7	7,8	556,7	7,7	556,7	6,7	560,2	6,7	555,4	8,3	556,7	11,4
vrp7	912,5	20,9	909,7	25,1	910,8	19,4	924,7	15,2	910,8	21,6	910,5	19,9	910,9	22,6	929,6	22,4	912,5	17,7	912,5	15,4	914,1	19,9	928,3	21,5	909,7	19,0
vrp8	875,2	42,5	869,1	27,9	877,6	33,5	871,3	36,1	875,2	45,5	881,7	39,0	872,2	53,9	871,1	47,2	870,3	46,4	870,7	59,7	874,2	45,3	867,4	41,8	867,2	44,8
vrp9	1.194,1	129,9	1.210,6	85,9	1.183,0	112,2	1.182,0	102,0	1.205,7	106,6	1.206,0	125,6	1.183,4	171,7	1.208,0	156,4	1.215,2	108,6	1.209,7	138,1	1.205,7	106,6	1.189,4	272,9	1.178,6	129,8
vrp10	1.434,5	305,7	1.442,2	211,6	1.428,7	250,9	1.431,6	268,6	1.434,9	259,3	1.448,1	305,2	1.422,4	439,0	1.439,0	283,9	1.443,4	295,2	1.443,7	375,8	1.434,9	259,3	1.441,9	446,6	1.422,5	266,5
vrp11	1.045,7	80,2	1.045,5	76,2	1.045,7	63,6	1.045,5	88,9	1.045,5	59,8	1.045,7	54,9	1.045,5	97,0	1.045,7	81,1	1.044,6	117,6	1.043,7	185,0	1.045,5	59,8	1.044,5	91,4	1.045,7	80,5
vrp12	819,6	25,4	819,6	28,1	819,6	25,0	819,6	23,5	819,6	27,1	821,2	25,6	819,6	37,3	819,6	28,7	819,6	25,6	819,6	24,4	819,6	25,0	819,6	25,2	819,6	34,3
vrp13	1.555,1	70,5	1.554,6	63,4	1.555,1	85,0	1.551,1	72,4	1.552,1	71,8	1.552,4	72,9	1.553,0	109,1	1.549,5	98,3	1.555,0	96,7	1.556,7	167,3	1.552,1	71,8	1.551,1	112,4	1.551,7	89,9
vrp14	866,5	32,6	866,5	31,7	866,5	36,9	866,5	39,2	866,5	34,9	867,0	28,3	866,4	41,3	866,4	33,6	866,5	41,2	866,5	40,0	866,4	29,9	866,4	30,4	866,4	32,5
TOTAL	13.844	1.081	13.839	898	13.815	993	13.818	1.026	13.859	1.018	13.877	957	13.789	1.598	13.842	1.254	13.855	1.111	13.853	1.382	13.846	980	13.825	1.371	13.767	1.134
MELHOR	1	0	6	5	3	0	3	3	1	0	1	5	5	0	6	0	2	0	3	0	3	1	6	0	9	0

5.4 Considerações finais do capítulo

À exceção dos algoritmos ABLAM e ABLRE, que propõem alterações na busca local e que apesar de reduzirem o tempo de processamento, prejudicam a qualidade da solução, os outros algoritmos construídos geraram resultados satisfatórios, sendo melhores do que o algoritmo original APRINS.

O algoritmo AMELHO, composto pelas alterações propostas em algoritmos selecionados neste trabalho, se mostra ainda mais promissor, pois teve o melhor resultado entre todos os algoritmos construídos.

Um resultado importante deste trabalho é que para a instância-teste vrp7, os algoritmos AINI10, AINI05, APIN03, ABLAM, ABLRE e AMELHO obtiveram uma solução melhor do que a obtida por Prins (2004).

6 CONSIDERAÇÕES FINAIS

6.1 Conclusões

Este trabalho tratou o problema de roteirização de veículos sem janelas de tempo e com frota uniforme e infinita.

A revisão bibliográfica mostrou os métodos utilizados para solucionar o problema de roteirização de veículos, que por ser um problema combinatório do tipo NP-difícil, não permite sua solução por métodos exatos para problemas reais. Assim, para resolvê-los são utilizados diferentes tipos de heurísticas, entre elas, as metaheurísticas, que podem ser definidas como estratégias e técnicas que guiam outras heurísticas a fim de se encontrar soluções melhores, ultrapassando o ponto de parada das heurísticas tradicionais (Cunha, 2006).

Uma das metaheurísticas utilizadas é o algoritmo genético. Aplicado com muito sucesso em problemas de roteirização de veículos com janelas de tempo, esta metaheurística não apresentava bons resultados para problemas sem janelas de tempo. O trabalho publicado por Prins (2004), no entanto, alterou esse panorama e inseriu os algoritmos genéticos no grupo das metaheurísticas com bons resultados para a solução de problemas de roteirização de veículos sem janelas de tempo.

O sucesso do algoritmo genético desenvolvido por Prins (2004) é devido principalmente ao seu cromossomo sem delimitadores de rotas. As rotas são geradas através de um eficiente método de particionamento do roteiro gigante (no caso, o cromossomo), que encontra a solução ótima para dada seqüência de pontos. Isto significa que há um cromossomo ótimo que irá gerar a solução ótima para o problema, ficando a cargo do paralelismo do algoritmo genético encontrar tal cromossomo.

Este trabalho propôs algoritmos genéticos inspirados em Prins (2004), com melhorias nas seguintes etapas do processamento: geração da população inicial,

operação de *crossover*, busca local, reinicialização e particionamento do cromossomo.

Os algoritmos propostos tinham um dos dois objetivos: melhorar a qualidade das soluções geradas ou reduzir o custo computacional do processo.

Aplicados às instâncias de Christofides et al. (1979), os algoritmos propostos com alterações em uma mesma etapa foram comparados entre si e com o algoritmo original APRINS.

À exceção dos algoritmos que propõem alterações na busca local, todos os outros algoritmos elaborados no presente trabalho apresentaram melhoras sobre o algoritmo original APRINS. Aqueles com os melhores resultados em cada uma das etapas foram combinados na construção do algoritmo AMELHO, que se mostrou ainda mais promissor. Alguns dos algoritmos propostos, inclusive, obtiveram resultados superiores aos publicados por Prins (2004).

6.2 Recomendações e próximos passos

Apesar de os resultados dos algoritmos propostos neste trabalho terem sido promissores, há ainda bastante espaço para melhorias.

As etapas de inicialização/reinicialização do algoritmo, em que para inserção de cromossomos são utilizadas as heurísticas clássicas desenvolvidas por Clarke e Wright (1964) e Gillet et Miller (1974) com a adição de parâmetros, podem ser melhoradas com a utilização de outras heurísticas para uma maior diversificação da população.

A etapa de busca local, que tem um custo computacional muito elevado e que este trabalho não conseguiu reduzir, pode ser melhorada com a alteração dos parâmetros aqui propostos para os algoritmos ABLRE e ABLAM. Outras tentativas válidas seriam utilizar alguma outra metaheurística nesta fase, como a busca tabu.

Os algoritmos também podem ganhar maior robustez, bastando na modelagem do problema adicionar à função objetivo, outros fatores que considerem não só o custo variável, mas também o custo fixo dos veículos. Os algoritmos propostos, com algumas modificações, também podem tratar o problema de frota heterogênea.

7 BIBLIOGRAFIA

ABRAHÃO, F. T. M. (2006*). A meta-heurística colônia de formigas para solução do problema de programação de manutenção preventiva de uma frota de veículos com múltiplas restrições: aplicação na Força Aérea Brasileira*. 2006. 137 f. Tese (Doutorado) – Escola Politécnica, Universidade de São Paulo, São Paulo.

AHUJA, R. K., MAGNANTI, T.L. e. ORLIN, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, NJ.

ASSAD, A. A. (1988). Modeling and implementation issues in vehicle routing. *In: Vehicle Routing: Methods and Studies*, B.L.Golden, A.A.Assad (eds), North Holland, Amsterdam, p. 7-46.

ATKINSON, J.B. (1998). A greedy randomized search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *Journal of the Operational Research Society*, v. 49, p.700-708

BALLOU, R.H. (1998). *Business Logistics Management: Planning, Organizing and Controlling the Suply Chain*. Prentice Hall, Uper Saddle River, New Jersey.

BEASLEY, J.E. (1983). Route-first cluster-second methods for vehicle routing. *Omega,* 11:403–8.

BELFIORE, P. (2006). *Scatter Search para problemas de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas*.2006. 115 f. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo.

BENT, R; VAN HENTENRYCK, P. (2004). A two stage hybrid local search for the vehicle routing problem with time windows. Transportation Science, v.38, p. 515-530.

BERGER, J., BARKAOUI, M. (2003). A hybrid genetic algorithm for the capacitated vehicle routing problem. *Cant´u-Paz*, E., ed.: GECCO03. LNCS 2723, Illinois, Chicago, USA, Springer-Verlag, p. 646–656.

BODIN, L.D.; GOLDEN, B.L.; ASSAD, A.; BALL, M. (1983). Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research* 10(2).

BONASSER, U. O. (2005). *Meta-heurísticas híbridas para solução do problema de roteirização de veículos com múltiplos depósitos e frota heterogênea fixa: aplicação na força aérea brasileira*. 2005. 288 f. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo.

BREJON, S. R. C. (1998*). Algoritmo para resolução do problema de programação do transporte de suprimentos para unidades marítimas de exploração de petróleo*. 1998. 171 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo.

CHRISTOFIDES, N.; MINGOZZI, A.; TOTH, P. (1979). The vehicle routing problem. In: *Combinatorial Optimization*, N. Christofides, A. Mingozzi e P. Toth (eds), Wiley, Chichester, p.315-338.

CHU, P.C., BEASLEY, J.E. (1997). A genetic algorithm for the generalized assignment problem. *Computers and Operations Research* 24(1), 17-23.

CLARKE, G.; WRIGHT, J. (1964). Scheduling of a Vehicule from a Central Depot to a Number of Delivery Points. *Operations Research,*12[4] :568-581.

COLORNI, A.; DORIGO, M.; MAFFIOLI, F.; MANIEZZO, V.; RIGHINI, G; TRUBIAN, M.; (1996). Heuristics from nature for hard combinatorial optimization problems. *International Transactions in Operational Research*, pages 1-21.

COOK, W. (2005). *The traveling salesman problem*. Acessível em http://www.tsp.gatech.edu/.

CORDEAU, J.F. ; GENDREAU, M ; HERTZ, A. ; LAPORTE, G. ; SORMANY, J.S. (2004). New heuristics for the vehicle routing problem, *Le cahiers du GERAD* G-2004-33.

CORMEN, T.H.; LEISERSON, C.E.; RIVEST, A.; RONALD, L.; STEIN, C. (2001). *Introduction to Algorithms, Second Edition*. Cambridge : MIT Press, 1180p

CUNHA, C.B. (1997). *Uma contribuição para o problema de roteirização de veículos com restrições operacionais*. São Paulo. 222p. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Transportes.

CUNHA, C.B. (2000). Aspectos práticos da aplicação de modelos de roteirização de veículos a problemas reais. *Transportes*, v.8, n.2, p.51-74.

CUNHA, C.B. (2006). *Contribuição à Modelagem de Problemas em Logística e Transportes*. 315p. Tese (Livre Docência) – *Escola Politécnica, Universidade de São Paulo, São Paulo*.

DANTZIG, G.B.; RAMSER, J.H. (1959). The Truck Dispatching Problem. *Management Science,* vol. 6, p. 80-91.

FEO, T.A.; RESENDE, M.G.C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v.6, p. 109-133.

FERIANCIC, G. (2005). *Modelagem matemática do problema de programação de entregas de derivados de petróleo*. 2005. 104 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo,.

FONSECA, C.H.F. (2002). *Alocação de Atendimentos de Coleta no Transporte de Remessas Expressas em Grandes Centros Urbanos*. Dissertação (Mestrado) Depto. de Eng. De Transportes, Escola Politécnica, Universidade de São Paulo. São Paulo.

GALVÃO, F.A. (2004). *Otimização do sistema de coleta de resíduos de biomassa de madeira para fins energéticos*. São Paulo. 196p. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, Programa de Mestrado em Engenharia de Sistemas Logísticos. São Paulo.

GENDRAU, M.; LAPORTE, G.; MUSARAGANYI, C. & TAILLARD, É.D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*. Vol. 26, p. 1153-1173.

GENDREAU, M.; LAPORTE, G.; POTVIN, J-Y. (1998). *Metaheuristics for the vehicle routing problem*. GERAD research report G-98-52, Montreal, Canada.

GILLET, B.E.: MILLER, L.R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, v.22, p.240-249.

GLOVER, F. (1986). Future paths in Integer Programming and links to Artificial Intelligence. *Computers and Operations Research*. Vol. 13, n.5, p. 533-549, 1986.

GLOVER, F.; LAGUNA, M. (1997). *Tabu Search*, Kluwer Academic Publishers, Boston.

GOLDEN, B.L.; ASSAD, A.; LEVY, L.; GHEYSENS, F. (1982). The fleet size and mix routing problem. *Management Science & Statistics Working Paper* n. 82-020, University of Maryland at College Park, 1982.

GOLDEN, B.L.; WASIL, E.A.; KELLY, J.P.; CHAO I.M. (1998). The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic TG, Laporte G, editors. *Fleet Management and logistics*. Dordrecht: Kluwer, p. 33–56.

GREFENSTETTE, J. J. (1987). Incorporating Problem Specific Knowledge in Genetic Algorithms. In: L. Davis (Ed.), *Genetic Algorithms and Simulated Annealing*, p. 42–60. Los Altos, CA: Morgan Kaufmann

HALL, R.W.; PARTYKA, J.G. (1997). On the road to efficiency. *OR/MS Today*, p.38-47, jun/97.

KIRKPATRICK, S.; GELATT, C.D.; VECCHI, M.P. (1983). Optimization by Simulated Annealing. *Science*, vol 220, n. 4598, p 671-680.

HO, S.; HAUGLAND, D. (2004). Local search heuristics for the probabilistic dial-a-ride problem. Technical Report 286, Department of Informatics. *University of Bergen, Norway*.

HOLLAND, J.H. (1975). Adaptation in Natural and Artificial Systems. *University of Mighigan Press, Ann Arbor*.

LAPORTE, G. (1992). The vehicle routing problem: an overview of exact and aproximate algorithms. *European Journal of Operational Research*, v.59, n.3, p.345-358.

LAPORTE, G.; GENDREAU, M.; POTVIN, J-Y.; SEMET, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research* 7(4), p.285-300.

LIN, S.; KERNIGHAN, B.W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, Vol. 21, No. 2, p. 498-516.

MICHALEWICZ, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edition. Springer-Verlag. Berlin, Germany.

MITCHELL, G.G.; O'DONOGHUE, D.; BARNES, D.; MCCARVILLE, M. (2003). GeneRepair - A Repair Operator for Genetic Algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '03*, LBP, 88-93.

MOLE, R.H.; JAMESON, SR (1976). A sequential route-building algorithm employing a generalized savings criterion. *Operational Research Quaterly,* 27:503–11.

MOSCATO, P.; COTTA, C. (2003). A gentle introduction to memetic algorithms. In: Glover, F., and Kochenberger, G.A. (eds), *Handbook of Metaheuristics*, pages 105–144. Kluwer, Boston.

MOURAD, F. A. (2005). *O problema de roteirização de veículos com carga completa e janelas de tempo*. 248 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo.

PIRLOT, M. (1996). General Local Search Methods. *European Journal of Operational Research*, v.92, p. 493-511.

POTVIN, J-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:339–70.

POTVIN, J-Y; BENGIO, S. (1996). The vehicle routing problem with time windows - Part II: genetic search. *INFORMS Journal on Computing*, 8:165–72.

PRINS, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31:1985–2002.

SCHMITT, L.J. (1995). *An evaluation of a genetic algorithmic aproach to the VRP*. Working paper, Department of Information Technology Management, Christian Brothers University, Memphis.

SOLOMON, M.M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, v.35, n.2, p.254-265

SOUZA, E. C. (1999). *Modelagem e resolução de um problema de transporte do tipo: "carga única - coleta e entrega" com janelas de tempo*. São Paulo. 89p. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia Naval e Oceânica.

TAILLARD, E.D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, v.23, p.661-673.

TAILLARD, É.D. (1999). *A heuristic column generation method for the heterogeneous fleet VRP. RAIRO*. Vol. 33, p.1-34.

TEIXEIRA, R.G. (2001). *Heurísticas para o Problema de Dimensionamento e Roteirização de uma* Rota *Heterogênea utilizando o Algoritmo Out-of-kilter*. 118p. Dissertação (Mestrado) – Departamento de Engenharia de Transportes, *Escola Politécnica, Universidade de São Paulo*.

THANGIAH, SR (1993*). Vehicle routing with time windows using genetic algorithms*. Report SRU-CpSc-TR-93-23, Slipery Rock University, Slipery Rock.

TORTELLY JR., A.; OCCHI, L. S. (2006). Um GRASP eficiente para problemas de roteamento de veículos. *Tendências em Matemática Aplicada e Computacional*, São José do Rio Preto, v. 7, p. 149-158.

TOTH, P.; VIGO, D. (1996). Fast local search algorithms for the handicaped persons transportation problem. In: *Meta-Heuristics: Theory and Aplications*, I. H. Osman, J. P. Kelly (eds.), Kluwer, Norwell, p.677-690.
ULUSOY, G. (1985): The Fleet Size and Mix Problem for Capacitated Arc Routing. *European Journal of Operational Research* v. 22, p. 329-337.

VAN BREEDAM, A. (1996). *An analysis of the e$ect of local improvement operators in GA and SA for the vehicle routing problem*. RUCA working paper 96/14, University of Antwerp, Belgium.

WREN, A.; HOLIDAY, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly,* v.23, p.333-344.

WU, L. (2007). *O problema de roteirização periódica de veículos*. 2007. 109 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo.

XU J.; KELLY J. (1996). A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science* v.30. p.379–93.

WASSAN, N.A. & OSMAN, I.H. (2002). Tabu search variants for the mix fleet vehicle routing problem. *Journal of the Operational Research Society*. Vol. 53, p. 768-782.

ZNAMENSKY, A. (2000). *Um modelo para a roteirização e programação do transporte de deficientes*. São Paulo. 144p. Dissertação (Mestrado) - Departamento de Engenharia de Transportes, Escola Politécnica da Universidade de São Paulo, São Paulo.