ABORDAGEM MONO-OBJETIVO E MULTIOBJETIVO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS

Lucas Carvalho Oliveira Matsueda
Departamento de Computação - Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro, CEP 35.400-000, Ouro Preto - MG, Brasil
pexe_matsueda@yahoo.com.br

Flávio Vinícius Cruzeiro Martins
Instituto de Ciências Exatas e Aplicadas - Universidade Federal de Ouro Preto
Rua 36, n° 15, Loanda, João Monlevade – MG
flavio@decea.ufop.br

RESUMO

Este trabalho propõe duas ferramentas para resolver o Problema de Roteamento de Veículos Capacitados (PRVC). A primeira implementa um Algoritmo Genético (AG) mono-objetivo, cujo objetivo é minimizar a distância total percorrido pelos veículos. A segunda implementa um AG multiobjetivo (NSGA-II), cujos objetivos são: minimizar o tempo de atendimento aos consumidores e o custo total de operação. Para avaliar a eficiência e robustez dos algoritmos desenvolvidos, foram utilizadas 4 instâncias que variam entre 22 e 101 consumidores. Os resultados de ambas as ferramentas foram comparados com o modelo exato, estes apresentaram, quando não iguais, resultados bem próximos, tendo como diferencial o tempo de execução.

PALAVRAS CHAVE. Problema de Roteamento de Veículos Capacitados. Algoritmo Genético. Otimização Multiobjetivo.

ABSTRACT

This paper proposes two tools for solving the Vehicle Routing Problem Empowered. The first tool implements a Genetic Algorithm (GA) mono-objective, whose goal is to minimize the total distance traveled by vehicles. The second tool implements a multiobjective version of the AG, the NSGA-II, whose goal is to minimize the time of service to consumers and the total cost of operation. To assess the efficiency and robustness of algorithms developed were used three instances of between 22 and 101 consumers. The results of both proposals are the tools showed good since they had, if not identical, results compared very close to the tools. Its diferencial runtime.

KEYWORDS. Vehicle Routing Problem Empowered. Genetic Algorithm. NSGA-II.

1. Introdução

O Problema de Roteamento de Veículos Capacitados (PRVC) resume-se no atendimento de um conjunto de consumidores por intermédio de uma frota de veículos. Tal problema apresenta o número de pedidos e a capacidade de carga da frota de veículos. Um pedido consiste em carregar um determinado veículo (coleta) com mercadorias e entregar estas a um determinado consumidor (entrega), sendo que: (1) para cada consumidor, bem como para o depósito, é atribuído uma localização e uma demanda. (2) o processo de coleta é executado somente no depósito.

Este artigo estuda o PRVC com frota homogênea, onde um único depósito é levado em consideração. Deste modo, os veículos coletam o máximo de produtos possíveis e, em seguida partem do depósito para atender aos consumidores. A entrega é realizada até que o veículo tenha carga suficiente para atender ao próximo consumidor, caso contrário, o mesmo retorna ao depósito. O percurso feito por um veículo que inicia e finaliza no deposito é denominado de rota. O problema acaba assim que todos os consumidores tenham sido atendidos.

Os primeiros a tratarem o Problema de Roteamento de Veículos (PRV) clássico foram Dantzig e Ramser (1959), estes procuravam resolver um problema real de distribuição de combustível de um terminal graneleiro para postos de combustíveis. Os autores propuseram um modelo de programação linear inteira, adaptando a utilizada para o problema do caixeiro viajante.

Para resolver o PRV Clark e Wrigth (1964) desenvolveram uma heurística que inicia com cada rota atendendo à apenas um consumidor. Nos passos seguintes a heurística procura fazer combinações de rotas, sendo que, a cada passo o número de rotas diminui. O processo é executado até que não seja mais possível gerar soluções factíveis. Deste modo, a heurística retorna a melhor combinação de rotas gerada.

Golden et al. (1984) foram os primeiros a tratar o PRV com frotas heterogêneas. Os autores desenvolveram uma heurística baseada na de Clark e Wrigth (1964), no qual desenvolveram fórmulas de economia que consideravam os custos fixos de cada tipo de veículo pertencente à frota. Já Gendreau et al. (1999) resolveram o PRV com frotas heterogêneas utilizando busca tabu.

Vural (2003) propôs a implementação de um AG para o PRV com coleta e entrega simultânea. Gökçe (2004) tratou o mesmo problema utilizando a metaheurística Colônia de formigas juntamente com a busca local *2-opt* como mecanismo de pós-otimização. As referências Gökçe, Çatay (2006), Gajpal e Abad (2009) e Zhang et al. (2008) também utilizaram Colônia de Formigas como parte da solução proposta para resolver o PRV.

Zachariadis et al. (2009) utilizaram um método híbrido que faz uso das metaheurísticas busca tabu e Guided Local Search (GLS). Em 2010, os mesmo autores propuseram um algoritmo evolucionário que utiliza uma memória adaptativa para guardar características das melhores soluções encontradas durante a busca. Tais informações são utilizadas para gerar novas soluções em regiões vantajosas do espaço de busca, sendo estas melhoradas posteriormente pela busca tabu.

O presente trabalho propõe o desenvolvimento de duas ferramentas para resolver o PRVC com frota homogênea. A primeira ferramenta implementa um AG que tem como objetivo traçar rotas de atendimento minimizando a distância total percorrida pela frota de veículos. A segunda ferramenta implementa um AG multiobjetivo (*Nondominated Sorting Genetic Algorithm II* (NSGA-II)) (DEBIAN, 2002) com os seguintes objetivos: traçar rotas de atendimento minimizando o custo total de operação e o tempo de atendimento aos consumidores.

2. Caracterização do Problema

O PRVC pode ser formulado da seguinte maneira, seja $G = (V, E)$ um grafo não direcionado, onde $V = \{v_0, v_1, \ldots, v_n\}$ é o conjunto dos vértices e $E = \{(v_{i,} v_j) : v_{i,} v_j \, E \, V, i < j\}$ é o conjunto de arestas. O vértice v_0 representa o depósito, sendo este a base de uma frota de veículos, enquanto os vértices remanescentes correspondem às cidades ou consumidores. Cada consumidor v_i tem uma demanda não negativa q_i e $q_0 = 0$. A cada aresta $(v_{i,} v_j)$ está associada

uma distância não negativa c_{ij} que representa a distância entre os consumidores (Junior et al., 2005). Para determinar conjuntos de rotas válidas para o PRVC é necessário respeitar as seguintes restrições (considerando um grafo completo):

- Cada rota começa e termina no depósito;
- Todo consumidor de *V*, com exceção do depósito, é visitado somente uma vez por somente um veículo;
- A demanda total de qualquer rota não deve superar a capacidade *Q* de um veículo.

Deste modo, o principal objetivo do PRVC é traçar um conjunto de rotas que atendam a todos os consumidores, otimizando aspectos relevantes para o problema. Os mais comuns são: minimizar o custo total da operação, minimizar o tempo total de transporte, minimizar a distância total percorrida, minimizar o tempo de espera, minimizar a utilização de veículos, maximizar o benefício e equilibrar a utilização dos recursos.

3. Algoritmo Genético Proposto

Para minimizar a distância total percorrida pelos veículos foi desenvolvido um AG mono-objetivo. As seções seguintes apresentam os detalhes deste algoritmo.

3.1. Codificação de uma solução

Para representar uma solução, ou seja, um indivíduo do AG foi utilizado uma lista que armazena os consumidores na sequência em que eles devem ser visitados pela frota de veículos. Sendo que, os indivíduos iniciais são gerados por uma permutação aleatória de inteiros (consumidores). Deste modo, o tamanho da lista varia de acordo com o número de consumidores e o número de rotas. Segue abaixo um exemplo que descreve um indivíduo:

Individuo: 2 1 4 3 6 7 5

Percebe-se neste exemplo que a solução proposta pelo indivíduo é que o veículo passe pelos clientes 2, 1, 4, 3, 6, 7 e 5, nesta ordem. No entanto, assim que o veículo não possuir carga suficiente para atender o próximo consumidor ele é obrigado a retornar ao depósito. No AG proposto, a população inicial é formada por uma permutação aleatória de 1 a N, onde N é o número de clientes.

3.2. Decodificação e avaliação da solução

Ao fazer a leitura do indivíduo o algoritmo inclui um zero (representação do depósito) no início e no final da lista, o que representa que o veículo inicia e termina sua rota no depósito. Visto que o veículo possui uma capacidade de carga máxima e que tal capacidade é decrementada de acordo com a demanda dos consumidores atendidos por ele, em determinado momento este veículo não possuirá carga suficiente para atender o próximo consumidor, sendo obrigados a retornar ao depósito. Deste modo, os zeros também são inseridos entre o ultimo consumidor atendido pelo veículo e o consumidor subsequente da lista.

Para o indivíduo s = [2 1 4 3 6 5], por exemplo, poderíamos ter a seguinte configuração: s = [0 2 1 4 3 0 6 7 5 0]. Assim, o veículo parte do depósito e passa pelos consumidores 2, 1, 4 e 3, nesta ordem, e volta ao depósito por não ter carga suficiente para atender ao consumidor 6. No passo seguinte o veículo parte novamente do depósito atendendo aos consumidores 6, 7 e 5 e retorna ao depósito, chegando ao fim da lista de consumidores.

A função responsável por avaliar os indivíduos calcula o somatório da distância entre os consumidores na ordem em que eles foram visitados, resultando na distância total percorrida pela frota de veículos. Considerando que cada consumidor e o depósito é representado por um ponto no plano cartesiano, a distância entre eles é dada pela distância euclidiana.

3.3. Crossover

Os métodos de *crossover* foram definidos com o intuito de gerar novos indivíduos (filhos) para a população. Estes recebem por parâmetro dois indivíduos que representam os pais e retornam dois novos indivíduos, que representam os filhos. Neste trabalho foram implementados os seguintes métodos de *crossover*: o Cruzamento de Mapeamento Parcial (PMX) e o

Cruzamento de ordem (OX). Estes foram escolhidos por sempre gerarem soluções (filhos) factíveis para o problema.

3.3.1. Cruzamento de Mapeamento Parcial

O operador PMX transfere informações de ordem e de posição das rotas dos pais para as rotas dos filhos. Uma parte da sequência de um pai é mapeada e uma parte da sequência do outro pai é preservada no filho, o restante das informações é trocado entre os pais (Malaquias, 2006).

Utilizando-se como exemplo a sequência (1 2 3 4 5 6 7 8) como a rota do pai 1 e (3 7 5 1 6 8 2 4) como a rota do pai 2, o operador PMX inicialmente seleciona dois pontos de corte aleatoriamente para a divisão dos pais. Partindo do princípio que o primeiro ponto de corte está entre o terceiro e quarto elemento (gene), e o segundo ponto de corte entre o sexto e sétimo elemento os pais são representados como segue na Figura 1.

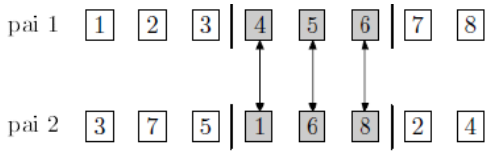

Figura 1. Exemplo da seção de Mapeamento do PMX (Malaquias, 2006).

No passo seguinte o PMX copia os genes da segunda parte do pai 1 para a segunda parte do filho 2. Da mesma forma, copiam-se os genes da segunda parte do pai 2 para a segunda parte do filho 1. O conjunto de genes pertencentes à segunda parte dos indivíduos é denominado de seção de mapeamento. A Figura 2 ilustra o processo de cópia da seção de mapeamento.

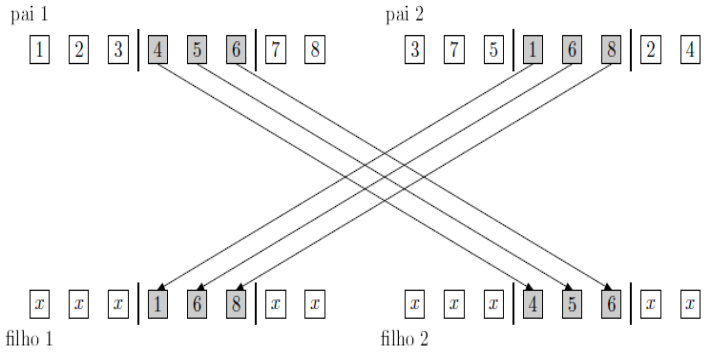

Figura 2. Cópia dos elementos da seção de mapeamento dos pais para os filhos (Malaquias, 2006).

Por fim, o PMX copia o restante dos genes do pai 1 para as respectivas posições do filho 1, começando pelo primeiro elemento da terceira parte. Caso o pai 1 tente inserir algum gene já existente no filho 1, o PMX verifica a posição deste gene no filho e, tenta inserir outro gene do pai correspondente a essa posição. Isso é feito até que se encontre um gene inédito para o filho. O mesmo processo acontece para o pai 2 e o filho 2. Assim, os filhos são preenchidos como segue na Figura 3.

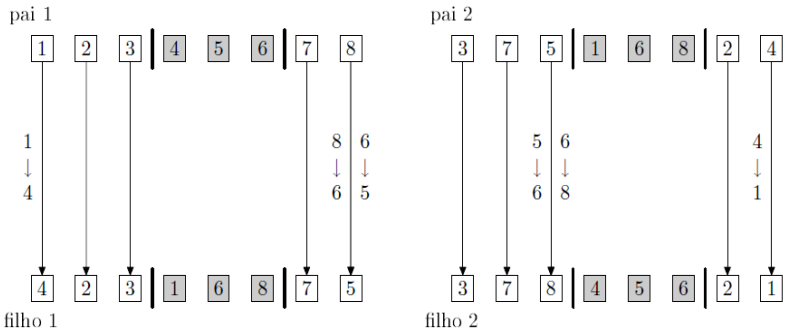

Figura 3. Cópia dos elementos restantes dos pais para os filhos (Malaquias, 2006).

3.3.2. Cruzamento e Ordem

O operador OX explora a propriedade de representação do caminho, em que a ordem é importante e não a posição. Ele escolhe um sub-roteiro de um dos pais preservando a ordem dos elementos do outro pai (Malaquias, 2006).

Assim como no PMX o operador OX inicia gerando dois pontos de corte aleatoriamente e copiando as seções de mapeamento dos pais para os filhos. Este processo é ilustrado nas Figuras 4 e 5.

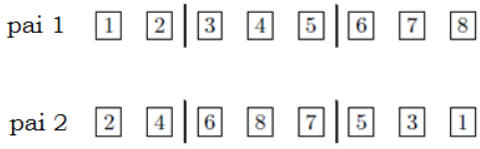

Figura 4. Exemplo da seção de Mapeamento do OX (Malaquias, 2006).

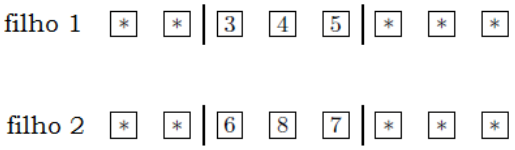

Figura 5. Resultado da cópia de seção de mapeamento nos filhos (Malaquias, 2006).

Por fim, o operador OX copia o restante dos elementos dos pais para as respectivas posições dos filhos, começando pelo primeiro elemento da terceira parte. A diferença entre o PMX e o OX é que, caso o pai tente inserir algum elemento já existente no filho, o operador OX busca o próximo elemento a direita contido no pai, até que se encontre um elemento inédito para o filho. Os filhos são representados pela Figura 6.

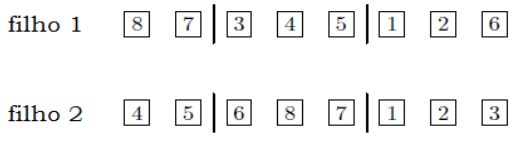

Figura 6. Resultado da execução do OX (Malaquias, 2006).

3.4. Mutação

Neste trabalho foram implementadas a mutação por inserção (ISM), mutação por inversão simples (SIM), mutação por troca (EM) e a mutação hibrida.

3.4.1. Mutação Por Inserção

O operador ISM escolhe aleatoriamente um elemento na sequência, remove este elemento e o insere em uma posição escolhida aleatoriamente (Malaquias, 2006). Partindo do princípio de que o indivíduo selecionado tenha a seguinte sequência de genes (1 2 3 4 5 6 7 8) e

que o quarto gene foi escolhido para preencher a sétima posição, o resultado da mutação é (1 2 3 5 6 7 4 8). A Figura 7 ilustra o processo feito pelo ISM.

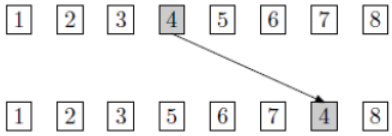

Figura 7. Representação do processo feito pelo ISM (Malaquias, 2006).

3.4.2. Mutação Por Inversão Simples

O operador SIM seleciona aleatoriamente dois pontos de corte e inverte os elementos do subconjunto formado a partir dos pontos de corte (Malaquias, 2006). Suponhamos que o indivíduo selecionado tenha a seguinte sequência de genes (1 2 3 4 5 6 7 8) e que o primeiro e o segundo ponto de corte estão entre o primeiro e o segundo gene, e o quinto e o sexto gene, respectivamente. Invertendo o subconjunto (2 3 4 5) o resultado da mutação é (1 5 4 3 2 6 7 8). A Figura 8 ilustra o processo feito pelo SIM.

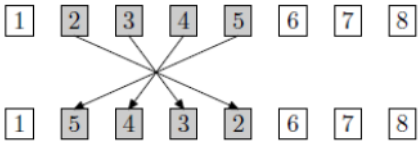

Figura 8. Representação do processo feito pelo SIM (Malaquias, 2006).

3.4.3. Mutação Por Troca

O operador de mutação por troca (EM) seleciona aleatoriamente duas posições no indivíduo e troca suas posições (Malaquias, 2006). Considerando a sequência (1 2 3 4 5 6 7 8) como exemplo e que os genes da terceira e da quinta posição foram selecionados, tem-se (1 2 5 4 3 6 7 8) como resultado da mutação. A Figura 9 ilustra o processo feito pelo EM.

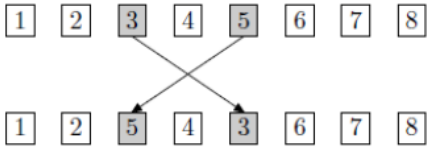

Figura 9. Representação do processo feito pelo EM (Malaquias, 2006).

3.4.4. Mutação Hibrida

O operador de mutação hibrida aplica os operadores ISM, SIM e EM no indivíduo selecionado. Tal método recebe um indivíduo e aplica o operador ISM sobre o mesmo. Caso haja melhora na função objetivo que calcula a distância dos veículos, este método retorna o indivíduo alterado, caso contrário o indivíduo não sofre alterações. A mutação hibrida aplica este mesmo processo utilizando os métodos SIM e EM, alterando os indivíduos caso estes tenham melhora no valor de FO.

3.5. Definição da População Sobrevivente e Elitismo

Foi utilizado no algoritmo o método de seleção por torneio e o elitismo simples. O elitismo simples é o processo em que, ao final de cada seleção o melhor indivíduo encontrado até o momento é inserido na população atual. Deste modo, o retorno do resultado é estabelecido depois da avaliação da última população, no qual, retorna o melhor indivíduo pertencente à mesma.

4. Algoritmo Multiobjetivo Proposto

Para minimizar o tempo de atendimento aos consumidores e o custo total de operação foi implementado um algoritmo multiobjetivo baseado no NSGA-II (DEBIAN, 2002).

4.1. Representação das Soluções

Para a representação dos indivíduos foram utilizadas duas listas. A primeira (Lista A) armazena os consumidores na sequência em que eles devem ser visitados pela frota de veículos. Enquanto a segunda (Lista B) armazena quais veículos irão atender as quais rotas.

A "Lista A" é gerada de forma aleatória, composta por uma permutação de inteiros. Para gerar a "Lista B" o NSGA-II verifica o número de rotas "r" necessárias para atender a todos os consumidores, a seção 3.2 descreve o processo que determina a quantidade de rotas necessárias para atender os consumidores uma determinada solução. Deste modo, a Lista B terá o tamanho igual ao número de rotas verificado. Em seguida é gerado um número aleatório "x", sendo $0 < x \leq r$, que representa a quantidade de veículos disponíveis (pertencentes à frota de veículos) para o atendimento aos consumidores. No passo seguinte o NSGA-II preenche a lista gerando números aleatórios entre -1 e –x. Desta forma, o primeiro elemento (código do veículo) da "Lista B" irá atender a primeira rota, o segundo elemento irá a atender a segunda rota e assim por diante. O algoritmo garante que todos os veículos da frota atenderão pelo menos uma rota.

Suponhamos que o indivíduo (Lista A = 1 2 3 4 5 6 7 8 9 10 11 12) necessite de 4 rotas para atender a todos os consumidores, resultando na sequência (0 1 2 3 0 4 5 6 0 7 8 9 0 11 12 0). Suponhamos ainda, que no procedimento de codificação sejam determinados 2 veículos para o atendimento dessas 4 rotas, resultando na sequência (Lista B = -2 -1 -1 -2). Dado as listas que representam o indivíduo (Lista A e Lista B), o NSGA-II o interpreta da seguinte forma:

Indivíduo = (-2 1 2 3 -1 4 5 6 -1 7 8 9 -2 11 12)

Percebe-se neste indivíduo que o veículo 2 atende a primeira (1 2 3) e a quarta (11 12) rota enquanto o veículo 1 atende a segunda (4 5 6) e a terceira (7 8 9) rota.

4.2. Avaliação da População

Para a avaliação dos indivíduos o NSGA-II proposto considera duas funções objetivo. A primeira ($f_1(x)$) retorna o custo total da operação enquanto a segunda ($f_2(x)$) retorna o tempo necessário para atender a todos os consumidores. Onde x é um candidato a solução do problema.

4.2.1. Função Objetivo 1

O cálculo do custo total da operação ($f_1(x)$) é uma das contribuições deste trabalho, no entanto aqui é apresentado um custo estimado de forma simplificada, mas de fácil adaptação.

$$f_1(x) = \left(\left(\frac{Km(x)}{Kl}\right) * Pc\right) + Nv(x) * Cv$$

onde: Km: distância total percorrida pela frota de veículos.
Kl: quilômetros por litro feito pelos veículos. *
Pc: preço do combustível (litro). *
Nv: número de veículos que pertencem à frota.
Cv: custo por veículo. *
*Kl, Pc e Cv são entradas do sistema, as demais variam de acordo com o indivíduo avaliado.

4.2.2. Função Objetivo 2

O tempo total de atendimento de um indivíduo é calculado a partir da distância percorrida por cada veículo da frota. Para cada veículo pertencente à frota o NSGA-II verifica a distância total percorrida pelo mesmo e divide este valor por uma velocidade média de tráfego dos veículos. Assim, cada veículo da frota está associado a um valor que representa o tempo gasto por ele. A seguinte equação apresenta o cálculo que retorna o tempo gasto por um determinado veículo.

$$T_i = \frac{D_i}{V}$$

onde: T_i: tempo gasto pelo veículo i;
D_i: distância percorrida pelo veículo i;
V: velocidade média da frota de veículos.

$$f_2(x) = T_{max} = max\{T_1, T_2, T_3, \ldots, T_n\}$$

onde: n representa o número de veículos que atenderam aos consumidores.

Contudo, o valor assumido $f_2(x)$ é igual ao tempo do veículo que apresentou o maior período de atendimento aos consumidores (maior T_i). Suponhamos, por exemplo, que um determinado indivíduo tenha a seguinte configuração (-1 1 2 3 -2 4 5 6 -2 7 8 0) e que para o atendimento da primeira, segunda e terceira rota gasta-se um tempo igual a 30, 20 e 15 minutos, respectivamente. Visto que o veículo 1 atende somente a primeira rota (1 2 3) e que o veículo 2 atende a segunda (4 5 6) e terceira (7 8) rota, conclui-se que o tempo gasto pelo veículo 1 é igual a 30 e o tempo gasto pelo veículo 2 é igual a 35 (20+15). Desta forma, o valor de $f_2(x)$ para este indivíduo é igual a 35. Dado que o veículo 2 foi aquele que apresentou o maior tempo de atendimento aos consumidores.

4.3. Cruzamento, Mutação e Seleção

Segundo Takahashi (2007), um Algoritmo Genético multiobjetivo pode ser construído a partir de modificações introduzidas em qualquer Algoritmo Genético mono-objetivo. Desta forma, os métodos de cruzamento, mutação e seleção aplicados no NSGA-II são semelhantes aos métodos aplicados no AG.

5. Resultados e Discussões

Neste capítulo, serão apresentados os resultados obtidos pelos algoritmos desenvolvidos para cada instância testada. Paralelamente aos resultados obtidos, será apresentada uma comparação do AG com os resultados do método exato (ME) proposto por (Fukasawa et al, 2006) e a comparação do NSGA-II com um algoritmo mono-objetivo. Os testes foram executados em um computador Pentium Dual Core 2.00GHz com 3GB RAM, utilizando apenas um núcleo para processamento.

5.1. Definição dos Métodos e Parâmetros

Executou-se o AG 33 vezes para a definição dos métodos e parâmetros que configuraram o AG e o NSGA-II. O desvio padrão e a média do valor de função objetivo foram considerados para tal escolha. A instância utilizada nos testes foi a eil51 (Reinelt, 2013).

Visto que existe uma possível interdependência dos objetos (métodos e parâmetros) colocados em questão, os testes podem ter apresentado resultados tendenciosos, sugerindo a escolha de métodos e parâmetros favorecidos pelas configurações utilizadas nos testes. A Tabela 1 mostra os parâmetros definidos para o AG e o NSGA-II.

Parâmetro	Valores
Tamanho da População	300
Operador de Cruzamento	OX
Operador de Mutação	Hibrida
Taxa de Cruzamento	100%
Taxa de Mutação	60%
Método de Seleção	Torneio
Número Médio de Gerações	4000

Figura 10. Parâmetros do AG e do NSGA-II.

5.2. Resultados do Algoritmo Genético Mono-objetivo

Nos testes realizados o AG foi executado de duas formas distintas. A primeira executa o algoritmo sem restrição do número de rotas (AGSR), ou seja, o AG não pune as soluções que apresentam o número de rotas diferente da solução apresentada pelo ME. Na segunda forma, o algoritmo é executado com restrição do número de rotas (AGCR), no qual o AG pune as soluções que apresentam o número de rotas diferente da solução apresentada pelo ME.

Como penalidade o AG soma o valor da FO do indivíduo punido ao valor da FO do pior indivíduo encontrado até o momento. Este processo favorece os indivíduos da população que apresentam o mesmo número de rotas da solução do ME. Desta forma, o AG força que os indivíduos apresentados como resultados tenham o mesmo número de rotas da solução apresentado pelo ME. O ME utilizado como comparação é uma versão do branch-and-cut proposto por Fukasawa et al (2006).

Para os testes foram utilizadas quatro instâncias apresentadas em (Reinelt, 2013), em que os tamanhos variam entre 30 e 101 locais de demanda (consumidores). Para cada instância testada o AGSR e o AGCR foram executados 33 vezes tendo como critério de parada 100 gerações sem melhoria no valor de FO.

Os resultados são apresentados pela Figura 11. Nos testes foram consideraremos 5-Quantis, sendo o q-00, q-25, q-50, q-75 e q-100 as soluções 1, 9, 17, 25 e 33 do conjunto ordenado de execuções do algoritmo. Para o nome de cada instância o número após o 'n' indica a quantidade de consumidores enquanto o número após o 'k' indica a quantidade de rotas traçadas pelo ME.

Instância	E-n30-k3 (eil30)				
Quantil:	q-00	q-25	q-50	q-75	q-100
AGSR	508	517	528	543	556
AGCR	537	546	549	557	563
ME	534				

Instância	E-n51-k5 (eil51)				
Quantil:	q-00	q-25	q-50	q-75	q-100
AGSR	532	564	580	595	612
AGCR	525	564	577	600	632
ME	521				

Instância	E-n76-k10 (eil76)				
Quantil:	q-00	q-25	q-50	q-75	q-100
AGSR	877	919	933	973	1037
AGCR	894	930	948	964	1044
ME	830				

Instância	E-n101-k8 (eilA101)				
Quantil:	q-00	q-25	q-50	q-75	q-100
AGSR	882	933	967	994	1043
AGCR	899	958	971	990	1097
ME	815				

Figura 11. Resultados do AG para as instâncias testadas.

A Figura 12 apresenta uma comparação dos melhores resultados entre os métodos.

E-n30-k3 (eil30)			
	FO	GAP	NR
AGSR	508	-4,86%	4
AGCR	537	0,56%	3
ME	534	-	3

E-n51-k5 (eil51)			
	FO	GAP	NR
AGSR	532	2,11%	5
AGCR	525	0,76%	5
ME	521	-	5

E-n76-k10 (eil76)			
	FO	GAP	NR
AGSR	877	5,66%	10
AGCR	894	7,71%	10
ME	830	-	10

E-n101-k8 (eilA101)			
	FO	GAP	NR
AGSR	882	8,22%	8
AGCR	899	10,30%	8
ME	815	-	8

Figura 12. Comparação da melhor solução entre o AGCR, AGSR e o ME. FO: distância total percorrida pela frota de veículos; GAP: porcentagem de aproximação do resultado das ferramentas em comparação com os resultados do ME; NR: número de rotas.

Para a instância E-n30-k3 o AGSR apresentou resultados melhores quando comparados com a solução proposta pelo ME, apresentando -4,86% como valor de GAP. Tais resultados foram atingidos pelo fato de que o AGSR utilizou uma rota a mais para o atendimento dos consumidores nestas soluções, diminuindo consideravelmente o valor de FO. Já o AGCR teve valor de GAP igual a 0,56% para a instância E-n30-k3, apresentando uma solução próxima da apresentada pelo ME.

Já para as instâncias E-n51-k5, E-n76-k10 e E-n101-k8 o AGSR e o AGCR não retornaram soluções melhores que a do ME, como mostrado na Figura 12. Os resultados seguiram o mesmo padrão, conseguindo chegar a resultados próximos ao do ME. No entanto as

heurísticas são justificáveis, pois conseguem resolver o problema para instâncias com um maior número de consumidores em um tempo hábil.

Na Figura 13 são mostrados os tempos de execução (em segundos) de cada um dos métodos. Sendo que o AGSR e o AGCR apresentam a média dos tempos referente às 33 execuções. O ME foi executado em um processador Pentium IV, 2.4 GHz, com 1GB de memória RAM. Tal comparação de tempo computacional é plausível, pois os processadores utilizados são similares e a ordem de grandeza dos resultados apresentados serem diferentes.

Tempo de Execução (s)			
	AGSR	AGCR	ME
E-n30-k3	3	4	7
E-n51-k5	8	9	65
E-n76-k10	20	21	80722
E-n101-k8	41	38	801963

Figura 13. Comparação dos tempos de execução entre o AGCR, AGSR e o ME.

5.3. Resultados para o Algoritmo Multiobjetivo.

Para cada instância testada o Algoritmo Multiobjetivo (AM) foi executado 33 vezes, gerando 33 Paretos com "n" soluções, sendo n > 0. As soluções resultantes (soluções pertencentes aos Paretos das 33 execuções) passaram pelo processo de não dominância, ou seja, todos os indivíduos que foram retornados pelo algoritmo como parte do resultado foram comparados entre si, sendo que as soluções não dominadas geraram um único Pareto denominado de Pareto-médio.

Para validar os resultados apresentados pelo AM foi implementado um Algoritmo Genético (mono-objetivo) que minimiza o custo total de operação das soluções (desconsiderando o tempo de atendimento), tal algoritmo é denominado de Algoritmo Genético Comparativo (AGC). Tal algoritmo foi implementado para a utilização da técnica épsilon-restrito. Este algoritmo utiliza os mesmos parâmetros e métodos propostos pelo AM considerando algumas diferenças:

1. O tempo de atendimento aos consumidores e o valor referente ao número de veículos pertencentes à frota passam a serem restrições. A violação de qualquer uma delas é tratada por penalidade.
2. Como penalidade o AGC soma o valor de FO do indivíduo punido ao valor de FO do pior indivíduo encontrado até o momento. Assim, o indivíduo que não atende ao tempo de atendimento e ao número de veículos estabelecido têm o valor de FO do pior indivíduo somado duas vezes ao seu valor de FO.

O AGC foi executado 33 vezes para cada nicho do Pareto-médio, a solução que apresentou o menor tempo de atendimento por nicho foi à escolhida para a comparação. A partir desta solução são definidas as restrições de tempo de atendimento e do número de veículos. Por exemplo, em um nicho que apresente 3 soluções, sendo 4, 5 e 6 os valores de tempo de atendimento apresentados pelas soluções deste nicho, a solução que irá representar o nicho será a de menor valor, no caso do exemplo a de valor 4.

Desta forma, ao executar o AGC para cada nicho espera-se que este retorne soluções que minimizem o custo de operação atendendo as restrições de tempo e veículos. Estes algoritmo é utilizado para fazer uma otimização Ɛ-restrita, formando o Pareto para comparação.

Em média o AGC apresentou 3,5 resultados factíveis das 33 execuções por nicho para a instância E-n51-k5. Se considerarmos somente as soluções factíveis teríamos os resultados como segue na Figura 14. Os círculos azuis representam as soluções comparadas do Pareto-médio e os asteriscos vermelhos representam as médias das soluções retornadas pelo AGC para cada nicho dos Paretos-médios.

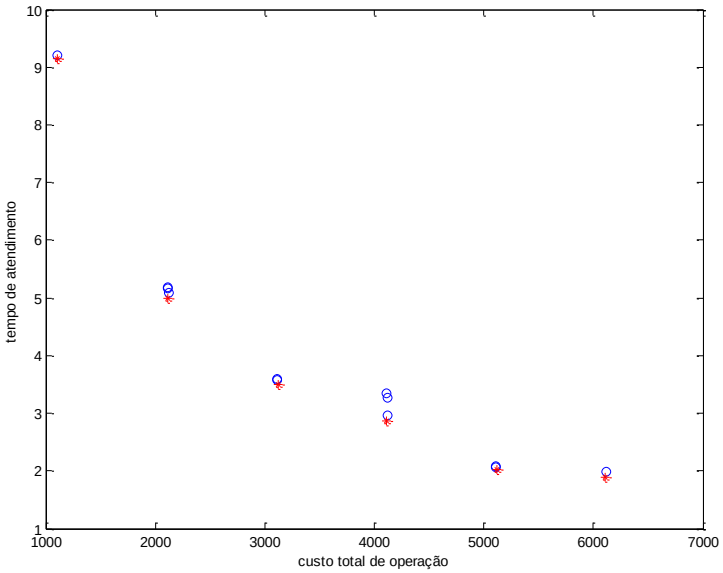

Figura 14 - Resultados do AGC (pontos factíveis) para a instância E-n51-k5.

Nos testes realizados o custo por veículo (Cv), quilômetros por litro (km/l), preço do combustível (Pc) e velocidade média dos veículos (V) receberam os valores \$1000,00, 10 km/l, \$2,00 e 60 km/h, respectivamente. A instância utilizada para os testes foi a E-n51-k5.

A Figura 15 apresenta o tempo médio de execução, em segundos, do NSGA-II e do AGC para a instância testada.

	Tempo do AGC	Tempo do NSGA-II
E-n51-k5	29,28	43,99

Figura 15 – Tempo de execução em segundos do NSGA-II e do AGC para a instância testada.

Tratando-se do tempo de execução o AGC apresentou valores menores comparados ao NSGA-II. No entanto, AGC gastou em média 29,28 segundos para retornar uma única solução para as instâncias testadas, já o NSGA-II retornou todo o Pareto em 43,99 segundos em média. Para o AGC formar todo o Pareto, por otimização Ɛ-restrita, ele necessita em média 178,8 segundos.

6. Considerações Finais

Neste trabalho foram desenvolvidas duas ferramentas para resolver o PRVC. A primeira implementou um AG e visou minimizar a distância total percorrida pela frota de veículos. A segunda implementou um algoritmo multiobjetivo baseado no *Nondominated Sorting Genetic Algorithm II* (NSGA-II) que procurou minimizar o custo total de operação e o tempo de atendimento aos consumidores. As soluções encontradas pelo AG foram comparadas com um método exato proposto por Fukasawa et al (2006). Diante dos testes realizados o AG se mostrou eficiente, visto que os resultados apresentados pelo mesmo, quando não alcançaram os resultados do Método Exato, se mostraram próximo, porém com um tempo computacional menor. Uma modelagem multiobjetivo foi sugerida, seus resultados foram comparadas com uma versão mono-objetivo do AG (AGC) que tinha o objetivo de minimizar o custo de operação respeitando as restrições de tempo e veículos. Os resultados da versão multiobjetivo também se mostram eficientes, visto que estes foram bem próximos aos da otimização Ɛ-restrita utilizando o AGC. Uma das grandes vantagens da abordagem apresentada é o fato de que o NSGA-II necessita de somente uma execução para retornar o Pareto-ótimo enquanto o AGC necessita de várias execuções para montar o mesmo, o que aumenta o tempo de execução.

7. Referências

Clark, G. e Wright, J., Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research, v. 12, p. 568-581, 1964.

Dantzig, G. B. e Ramser, J. H., The truck dispatching problem. Management Science, 6:80–91, 1959.

Debian, K., Pratap, A., Agarwal, S. e Meyarivan, T. (2002), A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 6(2), pp. 181–197.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E. e Werneck, R. F. (2006). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3):491-511.

Gendreau, M., Laporte, G., Musaraganyi, C. e Taillard, E. D., (1999) A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem. *Computers & Operations Research*, v. 26, p. 1153-1173.

Gökçe, E. I., A revised ant colony system approach to vehicle routing problems, Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University., 2004.

Golden, B., Assad, A., Levy, L. e Gheysens, F., The Fleet Size and Mix Vehicle Routing. Computers & Operations Research. Grã-Bretanha, v. 11, n. 1, p. 49-66, 1984.

GOMES Jr., A.C., SOUZA, M.J.F. e MARTINS, A.X., Algoritmos Simulated Annealing eficientes para resolução do problema de roteamento de veículos com janelas de tempo. Anais doXXXVII Simpósio Brasileiro de Pesquisa Operacional - SBPO, Gramado, p. 1270-1281, 2005.

Malaquias, N. G. L., Yamanaka, K. e Teixeira, T. R., (2006), Uso dos algoritmos genéticos para a otimização de rotas de distribuição, Universidade Federal de Uberlândia, Dissertação de mestrado.

Reinelt, G., TSPLIB-a traveling salesman problem library. ORSA Journal on Computing, 3:376-384, 1991.Disponível em http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

Takahashi, R. H. C., (2007), Otimização Escalar e Vetorial, Notas de Aula – Departamento de Matemática – UFMG.

Vural, A. V., A ga based meta-heuristic for capacitated vehicle routing problem with simultaneous pick-up and delivery, Master's thesis, Faculty of Management, Istanbul Technical University, 2003.

Werneck, R. F., Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming Series A, 106(3):491-511, 2006.

Zachariadis, E. E., Tarantilis, C. D. e Kiranoudis, C. T., A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service, Expert Systems with Applications, 36(2):1070–1081, 2009.

Zachariadis, E. E., Tarantilis, C. D. e Kiranoudis, C. T., An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, 202:401–411, 2010.

Zhang, T., xin Tian, W., jie Zhang, Y., e xin Liu, S., Improved ant colony system for vrpspd with maximum distance constraint, Systems Engineering - Theory and Practice, 28(1):132– 140, 2008.

Çatay, B., An ant based algortihm for the vehicle routing problem with simultaneous delivery and pick-up, INFORMS Meeting, 2006.