UM ALGORITMO MEMÉTICO PARA O PROBLEMA MULTI-OBJETIVO DE ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO

Rafael de Freitas Aquino
Departamento de Informática - Universidade Federal de Viçosa - UFV
Avenida Peter Henry Rolfs, s/n - Campus Universitário -Viçosa – MG - CEP: 36570-900
rafael.aquino@.ufv.br

José Elias Claudio Arroyo
Departamento de Informática - Universidade Federal de Viçosa - UFV
Avenida Peter Henry Rolfs, s/n - Campus Universitário -Viçosa – MG - CEP: 36570-900
jarroyo@dpi.ufv.br

RESUMO

O Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT) é um caso particular do problema clássico de Roteamento de Veículos em que as demandas dos clientes devem ser atendidos dentro de uma janela de tempo estabelecido. Neste trabalho aborda-se o PRVJT considerando a otimização simultânea de múltiplos objetivos. Os objetivos a serem minimizados são: distância total de percurso dos veículos, desequilíbrio nas distâncias percorridas e desequilíbrio das cargas dos veículos. Já que o problema é NP-Difícil, para determinar uma aproximação das soluções Pareto-ótimas, propõe-se um algoritmo genético híbrido que utiliza a heurística *Iterated Greedy* para melhorar a qualidade das soluções dominantes. O algoritmo proposto foi testado em instâncias disponíveis do PRVJT (*Solomon's benchmarks*) e foi comparado com dois algoritmos genéticos multi-objetivo da literatura. Os resultados obtidos foram analisados estatisticamente e observou-se que o desempenho do algoritmo proposto é significativamente superior.

PALAVRAS CHAVE. Roteamento de veículos, Otimização multi-objetivo, Meta-heurísticas.

Área Principal: MH – Metaheuristicas, OC – Otimização Combinatória

ABSTRACT

The Vehicle Routing Problem with Time Windows (VRPTW) is a particular case of the classical Vehicle Routing Problem in which the demands of each customer should be met within an established time window. In this paper we address the VRPTW with multi-objective optimization. The objectives are to minimize the total distance, the imbalance in the distances travelled and the imbalance in the loads of the vehicles. Since the problem is NP-Hard, in order to find near Pareto-optimal solutions, we propose a hybrid Genetic Algorithm which uses an Iterated Greedy heuristic to improve the dominant solutions. The proposed algorithm is tested on the Solomon's benchmark VRPTW instances and it is compared with two existing multi-objective genetic algorithms. The obtained results were analysed statistically and we observed that the performance of our algorithm is significatively superior.

KEYWORDS. Vehicle Routing Problem, Multi-objective optimization, Meta-heuristics.

Main Area: Meta-heuristics, Combinatorial optimization

1. Introdução

O problema de roteamento de veículos (PRV) é um problema clássico e complexo de otimização combinatória. Este problema possui muitas aplicações praticas nas áreas de transporte, logística de distribuição e gestão de cadeia de suprimentos [Toth and Vigo(2001)]. O objetivo principal do problema é reduzir os custos das rotas de veículos formadas para atender as demandas de um conjunto de clientes [Alvarenga et al.(2007)]. Os custos são relacionados principalmente com a distância total de percurso dos veículos, mas no problema existem outros critérios que podem ser otimizados, tais como, número de veículos usados, tempo total gasto para as entregas, balanceamento das cargas dos veículos e balecamento das distâncias percorridas pelos veículos.

O problema de roteamento de veículos com janela de tempo (PRVJT) é um caso particular de PRV no qual as demandas de cada cliente devem começar a ser supridas dentro de uma janela de tempo estabelecida. Este problema é bastante estudado na literatura, e considera-se um dos problemas mais difíceis da otimização combinatória, tendo um considerável impacto econômico em todos os sistemas de logística [Alvarenga et al.(2007)].

Vários objetivos diferentes foram considerados no PRVJT, sendo a distância total um dos mais abordados na literatura [Laporte et al.(2000)]. No entanto, a maioria dos trabalhos da literatura consideram a otimização de um único objetivo (abordagem mono-objetivo). Para o PRVJT, na literatura existem alguns trabalhos que consideram a otimização de mais de um objetivo.

Rahoual et al. [Rahoual et al.(2001)] desenvolveram um Algoritmo Genético, para o PRVJT, baseado no bem conhecido algoritmo NSGA [Srinivas and Deb(1994)]. Eles consideram a minimização do número de rotas e a distância total, e utilizam pesos para penalizar soluções inviáveis. Jozefowiez et al. [Jozefowiez et al.(2006)], abordam o PRV, minimizando a distância total e o desequilibro das rotas, e apresentam uma implementação melhorada do algoritmo NSGA II proposto por Deb et al. [Deb et al.(2002)]. Homberger et. al. [Homberger and Gehring(2005)], propuseram uma meta-heurística híbrida de duas etapas para solucionar o PRVJT, onde na primeira etapa tem como objetivo minimizar o número de rotas utilizando uma estratégia evolutiva, e na segunda etapa o objetivo é minimizar a distância total das rotas utilizando a meta-heurística Busca Tabu. Para a minimização do número de veículos e a distância total das viagens, [Ombuki et al.(2006)] e [Tan et al.(2006)], utilizam Algoritmos Genéticos adaptados para otimização bi-objetivo.

Mais recentemente Garcia-Najera et al. [Garcia-Najera and Bullinaria(2011)], desenvolvem um algoritmo genético denominado MOEA, que é baseado no NSGA II, para a minimização de dois objetivos simultaneamente: o número de rotas e a distância total das rotas. Outro trabalho recente é de Banos et al. [Banos et al.(2013)], que consideram a minimização da distância total das rotas, o desequilíbrio das distâncias e o desequilíbrio das cargas dos veículos. Banos et al. propõem um algoritmo chamado MMOEASA que é a combinação de um Algoritmo Evolutivo com Simulated Annealing [Kirkpatrick(1984)]. O desempenho do MMOEASA é comparado com os algoritmos NSGA II e SPEA2 [Zitzler et al.(2001)].

Neste trabalho propõe-se um Algoritmo Genético hibrido que utiliza a heurística *Iterated Greedy* [Ruiz and Stützle(2007)] como um procedimento de intensificação. Considera-se a minimização dos seguintes pares de objetivos: (distância total, desequilíbrio das distâncias) e (distância total, desequilíbrio das cargas). O algoritmo proposto é comparado com os algoritmos MMOEASA e NSGA II.

2. Definição do Problema

O problema de roteamento de veículos com janelas de tempo pode ser descrito da seguinte maneira: Existe um conjunto de $n+1$ locais, $V = \{0, 1, ..., n\}$, o local 0, representa o depósito, e os outros locais representam os clientes a serem atendidos. Cada cliente i possui uma demanda $q_i > 0$, uma janela de tempo $[a_i, b_i]$ e um tempo de serviço s_i (tempo necessário para suprir a demanda ao cliente). a_i e b_i são, respectivamente, os tempos (horários) mais cedo e mais tarde para

começar o atendimento ao cliente i. O depósito não tem demanda, porém possui uma janela de tempo $[a_0, b_0]$, onde a_0 representa início do horário de funcionamento do deposito, possibilitando o início da distribuição dos veículos (viagem), e b_0 corresponde ao horário onde o deposito encerra as distribuições, os veículos devem retornar ao deposito no máximo até esse horário. Considera-se que existem um número de veículos suficiente para atender todos os clientes. Os veículos são homogêneos, isto é possuem uma mesma capacidade Q. São conhecidos as distâncias (d_{ij}) e os tempos das viagens (t_{ij}) entre todo par de locais *i* e *j*. O objetivo do problema é determinar as rotas dos veículos que serão utilizados para atender as demandas dos clientes de tal maneira sejam otimizados um ou mais critérios (funções objetivo) satisfazendo as seguintes condições:

- Cada cliente deve ser atendido por um único veículo;
- A rota de um veículo deve começar e terminar no depósito;
- A demanda total atendido por um veículo não deve exceder a capacidade Q do veículo;
- Um veículo deve chegar no cliente antes do fim da janela de tempo b_i. Ou seja, o tempo (horário) de chegada do veículo no cliente i (r_i) deve ser menor que b_i ($r_i < b_i$).
- O serviço de entrega no cliente i não deve iniciar antes do tempo a_i. Ou seja, o serviço no cliente i inicia no tempo: $c_i = max\{r_i, a_i\}$. Se o veículo chegar antes do começo da janela de tempo, ele vai ter que aguardar até o tempo a_i para iniciar o serviço. Neste trabalho não consideramos penalizações por adiantamento de chegadas (ou seja, esperas) dos veículos.
- Se o cliente j é atendido imediatamente após do cliente i, o tempo de chegada do veículo no cliente j é $r_j = c_i + s_i + t_{ij}$.

Neste trabalho, considera-se que uma unidade de distância corresponde a uma unidade de tempo de viajem, ou seja, $d_{ij} = t_{ij}$.

As funções objetivo a serem minimizadas neste trabalho são: distância total, desequilíbrio das distâncias percorridas pelos veículos e o desequilíbrio das cargas dos veículos. Com estes objetivos, desejam-se que os veículos percorram as menores distâncias possíveis, os veículos percorram as mesmas distâncias (balanceamento das distâncias) e a carga dos veículos sejam as mesmas (balanceamento das cargas).

Suponha que, numa solução s do PRVJT, são utilizadas nv veículos (rotas), e sejam D_k e Q_k a distância total (percorrida) e a carga total do veículo k, respectivamente. Então, a distância total (f_1), o desequilíbrio das distâncias (f_2) e o desequilíbrio das cargas (f_3) são matematicamente definidas como segue:

$$f_1(s) = \sum_{k=1}^{nv} D_k \tag{1}$$

$$f_2(s) = \max_{k=1...nv} \{D_k\} - \min_{k=1...nv} \{D_k\} \tag{2}$$

$$f_3(s) = \max_{k=1...nv} \{Q_k\} - \min_{k=1...nv} \{Q_k\} \tag{3}$$

Neste trabalho são otimizados dois objetivos simultaneamente, ou seja, considera-se uma abordagem bi-objetivo. Como a distância total (f_1) pode ser considerada como o objetivo mais importante (é a mais abordada na literatura), então combina-se este objetivo com os outros. Ou seja, são considerados os pares de objetivos (f_1, f_2) e (f_1, f_3).

Em otimização multi-objetivo, geralmente, não é possível encontrar uma única solução s que determine o valor ótimo de todos os objetivos. Assim, os problemas de otimização multi-objetivo possuem como resposta um conjunto de soluções denominadas Pareto-ótimas ou eficientes

(melhores soluções para o problema). Essas soluções são incomparáveis, ou seja não é possível definir qual a melhor solução no conjunto.

Sem perda de generalidade, consideremos a minimização de todos os objetivos. Em otimização multi-objetivo, as soluções são classificadas utilizando o conceito *relação de dominância*. A seguir são apresentados alguns conceito utilizados em otimização multi-objetivo.

- Sejam s e s' duas soluções. s domina s' ($s \preceq s'$) se $f_i(s) \leq f_i(s')$ para todo os objetivos i, e $f_i(s) < f_i(s')$ para pelo menos um objetivo i.
- Uma solução s é Pareto-otima (ou eficiente) se não existe s' que domine s. O conjunto de solução Pareto-otimas é chamado de Fronteira Pareto-otima.
- Os valores dos no objetivos de todas as soluções viáveis de um problema determina um conjunto de pontos $(f_1, ..., f_{no})$ no espaço R^{no} (espaço objetivo).

Em problemas de otimização multi-objetivo, geralmente, existe uma grande quantidade de soluções Pareto-otimas. O objetivo dos métodos heurísticos é obter, em uma única execução, um conjunto de soluções que é uma aproximação da Fronteira Pareto-otima. O conjunto obtido por um método heurístico, neste trabalho, é chamado de conjunto de soluções dominantes (*D*), pois de todas as soluções analisadas pelo método, as soluções que não são dominadas por nenhuma solução são retornadas.

3. Algoritmo Memético Proposto

Nesse trabalho propõe-se um Algoritmo Memético [Moscato and Cotta(2003)] para resolver o PRVJT bi-objetivo. Ele consiste de um Algoritmo Genético (GA) que utiliza uma procedimento de melhoria baseado na heurística *Iterated Greedy* (IG) proposto por [Ruiz and Stützle(2007)]. O GA é uma meta-heurística, baseada no conceito de dominância, que melhora uma população de soluções usando operações de crossover e mutação. Já o IG é um procedimento de busca local, aplicado em soluções dominantes, que procura soluções melhores utilizando uma estratégia gulosa.

Algorithm 1 GA-IG($Psize$, $prob_s$, $prob_c$, $prob_m$)

1: Gera população inicial *P* de tamanho $Psize$;
2: *D* = Dominantes(*P*);
3: **while** (*Critério de parada* = *false*) **do**
4: M = Intensificação-IG(*D*);
5: S = Seleção($D, P \cup M, prob_s$);
6: Q = Crossover($S, prob_c$);
7: Q = Mutação($Q, prob_m$);
8: R = Classificação($P \cup M \cup Q$);
9: P = Sobrevivência(R);
10: D = Dominantes($D \cup P$); *//atualiza D*
11: **end while**
12: **Return** D;

O algoritmo proposto é denominado GA-IG. As etapas principais do algoritmo são apresentadas no Algoritmo 1. O algoritmo começa criando uma população *P* com *Psize* soluções iniciais (Linha 1). Em seguida, o conjunto de soluções não-dominadas (ou dominantes) *D* da população atual é determinado (Linha 2). O laço principal do algoritmo começa com o procedimento de Intensificação-IG aplicado nas soluções dominantes do conjunto *D* (Linha 4) obtendo como resultado um conjunto de soluções melhores *M*. O procedimento de Seleção (Linha 5) escolhe as soluções pais que serão utilizadas no *crossover*. Este procedimento seleciona soluções dominantes de $P \cup M$ de acordo probabilidade $prob_s$. As soluções selecionadas são armazenadas num conjunto *S*. A partir do conjunto S, gera-se um conjunto *Q* de soluções filhas aplicando o operador *Crossover* (Linha 6). Aplica-se o operador de Mutação (Linha 7), de acordo com a probabilidade $prob_m$, a cada solução filha pertencente ao conjunto *Q*. Em seguida é feito uma classificação das soluções do conjunto $P \cup M \cup Q$ (Linha 8). Esta classificação primeiro identifica as soluções dominantes de $P \cup M \cup Q$, e as soluções dominadas são ordenadas de acordo com sua proximidade às soluções

dominantes. A partir das soluções classificadas (conjunto R), aplica-se o procedimento de sobrevivência para obter uma população P, com *Psize* soluções, que será utilizada na próxima iteração (Linha 9). Nesta nova população estarão (sobreviverão) as soluções dominantes do conjunto R e algumas soluções que estejam próximas às dominantes, assim, completando *Psize* soluções. A proximidade de soluções é medida através da distancia euclidiana no espaço objetivo. Finalmente conjunto de soluções dominantes D é atualizado (Linha 10). Os procedimentos do GA-IG são executados até um critério de parada seja satisfeito. A Figura 1, ilustra o funcionamento do Algoritmo GA-IG.

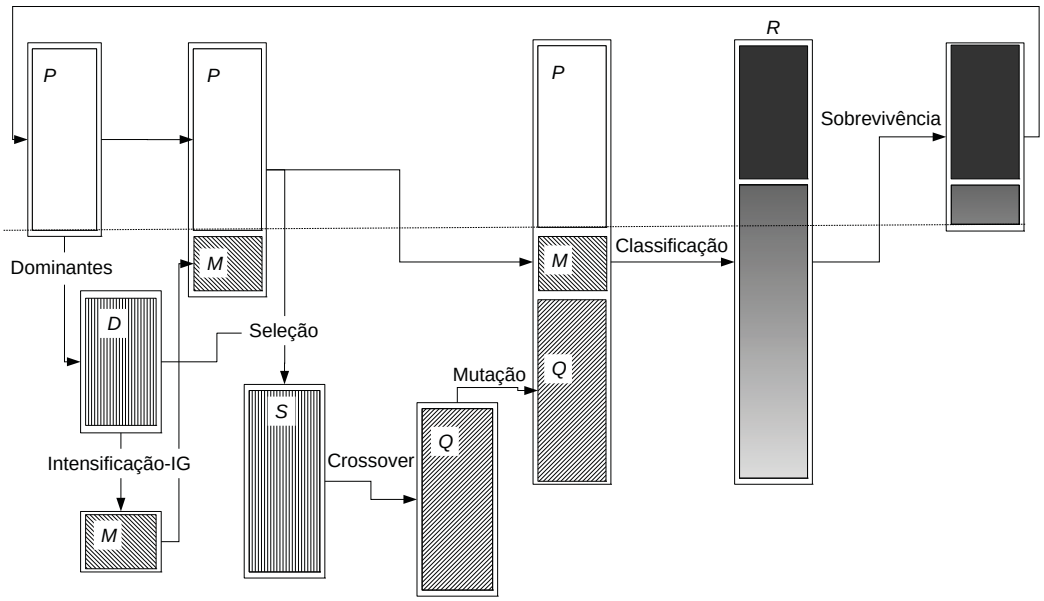

Figura 1: Etapas do algoritmo GA-IG.

3.1. Solução Inicial

As soluções inicias são geradas da seguinte maneira. Primeiramente uma ordem de inserção aleatória de clientes é gerada. Cada cliente é inserido numa rota de acordo com a ordem de gerada. Iniciando com uma rota vazia, os clientes sempre são inseridos no final da rota. Se a rota se tornar inválida com a inserção de um cliente i (i.e. a capacidade do veículo ou a janela de tempo for violada), a rota é finalizada e uma nova rota é criada e cliente i é inserido nela. Esta nova rota se torna a rota atual. Esse processo é repetido até que todos os clientes tenham sido inseridos.

3.2. Intensificação

O procedimento de intensificação é baseada no algoritmo heurístico *Iterated Greedy* proposto inicialmente por Ruiz e Stützle (2007). O IG possui duas fases: Destruição e Reconstrução. A Destruição consiste em remover alguns elementos da solução corrente obtendo uma solução parcial. Na Reconstrução, cada elemento removido é reinserido, de forma gulosa, na solução parcial. O IG retorna a melhor solução completa obtida.

O procedimento de Intensificação-IG proposto neste trabalho é apresentado no Algoritmo 2. Neste algoritmo, o IG é aplicado de duas maneiras: IG mono-objetivo e IG bi-objetivo. O IG mono-objetivo é aplicado às duas soluções dominantes, s_1 e s_2 ($s_1, s_2 \in D$), que possuem os menores valores de $f1$ e f_2, respectivamente. Na fase de Destruição do IG mono-objetivo, remove-se aleatoriamente d_1 clientes da solução s_i. Ou seja, estes clientes são retiradas de rotas, também, selecionadas aleatoriamente. Assim, obtém-se uma solução parcial. Na fase de Reconstrução, cada cliente removido é reinserido na melhor posição possível da solução parcial. Ou seja, para cada cliente procura-se a rota onde ocupe a melhor posição. A melhor posição é terminada de acordo ao valor da função objetivo (f_1 ou f_2). Note que, na reinserção de um cliente i, são gerados um

número ns_i de soluções parciais (para cada posição possível do cliente, obtém-se uma solução parcial). Dentre as ns_i soluções parciais seleciona-se a melhor (suponha s_m). Então o próximo cliente removido deve ser reinserido na melhor posição de s_m. O IG mono-objetivo finaliza quando o último cliente removido for reinserido na solução, assim é obtido a melhor solução completa s'.

O IG bi-objetivo é aplicado a cada solução s do conjunto D. Na fase de Destruição remove-se aleatoriamente d_2 clientes da solução s. Na fase de Reconstrução, também, cada cliente removido é reinserido na melhor solução possível da solução parcial. Das ns_i soluções parciais obtidas na reinserção de um cliente, determina-se o conjunto das soluções parciais dominantes (D_i). O próximo cliente removido será reinserido na melhor posição de cada solução parcial do conjunto D_i. Então, um novo conjunto de soluções parciais dominantes será obtido. O procedimento finaliza quando é obtido um conjunto D_i de soluções dominantes completas, ou seja, com todos os clientes já reinseridos. No final do procedimento de Intensificação-IG tem-se o conjunto M das soluções dominantes obtidas no IG mono-objetivo e bi-objetivo.

Algorithm 2 Intensificação-IG(D)

```
1: M = ∅;
2: for all função objetivo f_i do
3:     s_i = EncontraSoluçãoComMenorObjetivo(f_i, D);
4:     s' = IG-mono-objetivo(s_i, f_i);
5:     M = Dominantes(M ∪ {s'});
6: end for
7: for all solução dominante s de D do
8:     D_i = IG-bi-objetivo(s);
9:     M = Dominantes(M ∪ D_i);
10: end for
11: retorna M;
```

3.3. Seleção

O procedimento de Seleção consiste em determinar um conjunto S com $Psize$ soluções pais para serem utilizadas pelo operador de *crossover*. Este procedimento tem com entrada os conjuntos D (soluções dominantes), $P \cup M$ e uma probabilidade de seleção $prob_s$. Essa probabilidade é utilizada da seguinte maneira. Para cada solução a ser inserida em S, gera-se um número aleatório r, $0 \leq r \leq 1$. Se $r \leq prob_s$, seleciona-se aleatoriamente uma solução dominante do conjunto D, caso contrário, seleciona-se aleatoriamente uma solução do conjunto $P \cup M$, ou seja, a probabilidade para escolher uma soluções dominante é $prob_s$, e a probabilidade para escolher uma solução de $P \cup M$ é $1 - prob_s$.

3.4. *Crossover*

Neste trabalho foi utilizado o operador de *crossover* proposto por Garcia-Nareja e Bullinaria (2011). Neste operador, duas soluções pais, s_1 e s_2, são selecionadas do conjunto S. Uma nova solução filho $s*$ é gerada combinando s_1 e s_2. A combinação é da seguinte maneira: um número de rotas são selecionadas aleatoriamente da primeira solução s_1 e copiadas para o filho $s*$. Todas as rotas de s_2, cujos clientes ainda não estejam em $s*$, são copiadas para $s*$. Finalmente, os clientes remanescentes são inseridos em $s*$ mantendo a ordem em que aparecem em s_2. Estes clientes são inseridos nas rotas de $s*$ que produzam o menor valor da distância total. Caso não seja possível fazer a inserção viável de um cliente em nenhuma rota de $s*$, uma nova rota é criada em $s*$ para inserir esse cliente. O operador de crossover é repetido até obter uma população Q com $Psize$ soluções filhos. Na Figura 2 mostra-se um exemplo do funcionamento do crossover.

3.5. Mutação

No GA-IG é utilizada a mutação proposta por [Garcia-Najera and Bullinaria(2011)]. Aplica-se mutação a cada filho gerado pelo *crossover* com uma probabilidade de $prob_m$. São aplicadas três operadores de mutação. Estas mutações utilizam as seguintes operações básicas:

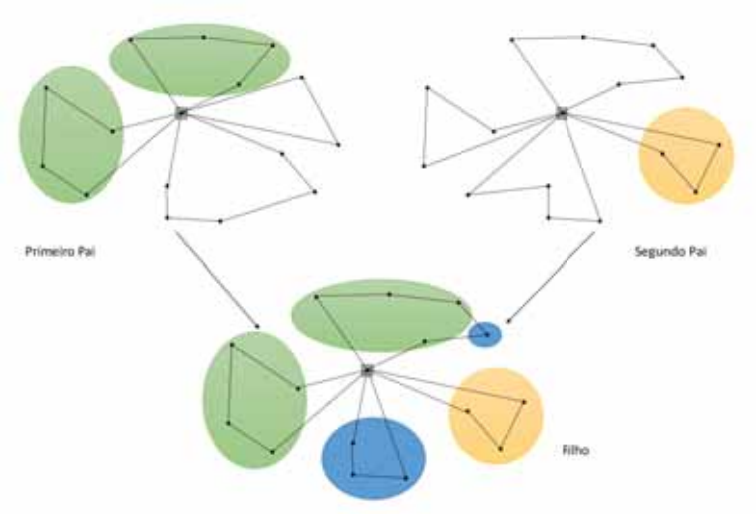

Figura 2: Exemplo do funcionamento do operador *crossover*.

- *SelecionaRota*: consiste em selecionar aleatoriamente uma rota de acordo com a razão entre a distância de viagem da rota e o número de clientes que ela possui. Ou seja, rotas longas com poucos cliente tem maior chance de serem escolhidas.
- *SelecionaClientes*: seleciona aleatoriamente um cliente de uma rota especifica de acordo com a média do tamanho de seus arcos de entrada e de saída. Clientes com maior média possuem maior chance de serem escolhidos.
- *InsereClientes*: esta operação tenta inserir um conjunto de clientes, um de cada vez, em uma rota especifica onde seja obtida a menor distância de viagem. Se nenhuma rota for especificada, todas as rotas são testadas.

Os operadores de mutação são:

- *Realocação*: consiste em mover um número de clientes de uma rota para outra. Dois clientes são escolhidos de uma rota fornecida usando a operação $SelecionaClientes$. Estes clientes são removidos da rota, incluindo os clientes entre eles. Por fim, a operação $InsereClientes$ tenta inserir os cliente removidos em todas as possíveis rotas existentes.
- *Troca*: este operador troca sequências de clientes entre duas rotas escolhidas pela operação $SelecionaRota$. Primeiramente, dois clientes são selecionados em cada rota utilizando a operação $SelecionaClientes$. A sequência de clientes entre os clientes escolhidos de cada rota é removida da sua rota e a operação $InsereClientes$ tenta inserir essa sequencia de clientes em outra rota. Se um ou mais clientes não puderem ser inseridos, a troca é cancelada e as rotas originais são mantidas.
- *Reposicionar*: utiliza as operações $SelecionaClientes$ e $InsereClientes$ para selecionar um cliente de uma rota especifica e reinserir-lo na mesma rota, respectivamente.

Os operadores de mutação funcionam da seguinte maneira: duas rotas são selecionadas utilizando $SelecionaRota$. Se elas forem iguais, o operador de *Realocação* é aplicado, caso contrario o operador *Troca* é aplicado. Em seguida, a operação $SelecionaRota$ escolhe uma terceira rota e aplica-se o operador *Reposicionar*.

4. Experimentos Computacionais

Nesta seção testa-se o desempenho do algoritmo proposto GA-IG através da comparação com os algoritmos MMOEASA (proposto por [Banos et al.(2013)]) e o algoritmo NSGAII (proposto por [Deb et al.(2002)]). O primeiro algoritmo foi proposto para resolver o PRVJT multi-objetivo, já o NSGAII é um algoritmo genético muito utilizado na literatura para resolver diferentes problemas de otimização multi-objetivo.

Os algoritmos GA-IG, MMOEASA e NSGAII foram codificados em *C++* e executados em um computador Intel i5 2500 com 3.3GHz e 8GB de memória, rodando Windows 8.1. Neste trabalho utiliza-se o mesmo critério de parada para todos os algoritmos. Este critério é baseado no tempo de CPU. Depois de realizar alguns testes computacionais, esse tempo foi fixado em 40 segundos. Então, para resolver uma instância do problema, todos os algoritmos são executados 40 segundos. Nas seções a seguir, são apresentados as instâncias do problema utilizadas para os testes, as métricas usadas para medir o desempenho dos algoritmos, os parâmetros utilizados nos algoritmos e os resultados e analises das comparações feitas.

4.1. Instâncias de teste para o PRVJT

Nesse trabalho foram utilizadas o conjunto de instâncias de teste propostos por Solomon [Solomon(1987)]. Estas instâncias são muito utilizadas parta testar algoritmos para o PRVJT. Esse conjunto inclui 56 instâncias, todas com 100 clientes. Estas instâncias são separadas em três categorias:

Conjuntos *C1* e *C2*: os clientes são distribuídos em clusters.

Conjuntos *R1* e *R2*: os clientes são distribuídos de forma aleatória.

Conjuntos *RC1* e *RC2*: com distribuição de clientes, parte em clusters e parte aleatória.

As instâncias dos conjuntos *C1*, *R1* e *RC1* possuem veículos com capacidades baixas de carga, e clientes com menores janelas de tempo, tornando assim suas rotas menores e forçando o uso de um número maior de veículos. Já as instâncias dos conjuntos *C2*, *R2* e *RC2* apresentam veículos com maiores capacidades de carga, e com maiores janelas tempo, assim sendo possível gerar rotas mais longas.

4.2. Métricas para a avaliação de desempenho de algoritmos

Para cada instância do problema, comparam-se os conjuntos de soluções dominantes obtidos pelos algoritmos. Como a fronteira Pareto-ótima não é conhecida, um conjunto de referência é construído, utilizando as soluções dominantes encontradas por todos os algoritmos. Suponha que D_1, D_2 e D_3 sejam, respetivamente, os conjuntos de soluções dominantes obtidos pelos algoritmos GA-IG, MMOEASA e NSGAII. O conjunto de refenrêcia (melhor fronteira conhecida) é denotado por *Ref* é o conjunto das soluções dominantes de $(D_1 \cup D_2 \cup D_3)$. O desempenho dos algoritmos é medido em termos da qualidade do conjunto de soluções dominantes D_i com relação ao conjunto *Ref*. Neste trabalho, utilizam-se duas métricas de avaliação:

- A métrica denominada *hypervolume* que calcula o desvio percentual relativo do um conjunto D_i com relação ao conjunto *Ref*. Essa métrica é definida da seguinte maneira:

$$H(D_i)\% = 100 \times \frac{H_{Ref} - H_{D_i}}{H_{Ref}} \tag{4}$$

  onde H_X representa o hypervolume (área, no caso de dois objetivos) do conjunto X, que é a porção do espaço objetivo que é dominada pelas soluções de *X*. Em outras palavras, H_X é o tamanho do espaço objetivo coberto pelo conjunto *X* [Zitzler and Thiele(1999)]. Para determinar H_{Ref} e H_{D_i}, no caso da minimização de dois objetivos, é usado um ponto de referencia (x, y) para limitar as coberturas, onde x e y são limites superiores dos objetivos f_1 e f_2, respectivamente.

  Menores valores de $H(D_i)$ correspondem a soluções de maior qualidade no conjunto D_i, ou seja, uma melhor cobertura do espaço objetivo.

- Métrica *epsilon+* ($I_{\varepsilon+}$) é definida da seguinte maneira [Zitzler et al.(2003)]:

$$I_{\varepsilon+}(D_i, Ref) = \underset{z^2 \in Ref}{max} \{ \underset{z^1 \in D_i}{min} \{ \underset{1 \le i \le no}{max} \{z_i^1 - z_i^2\}\}\} \tag{5}$$

  onde $z^1 = (z_1^1, ..., z_{no}^1)$ e $z^2 = (z_1^2, ..., z_{no}^2)$ são pontos no espaço objetivo e no é o número de objetivos minimizados.

  Esta métrica determina a maior distância entre um ponto de referência e o ponto mais próximo de D_i (determinado por uma heurística). Então, quanto menor o valor de $I_{\varepsilon+}$, melhor é a aproximação do conjunto D_i ao conjunto *Ref*.

4.3. Parâmetros dos Algoritmos

O critério de parada utilizado no algoritmo proposto GA-IG é baseado em tempo de execução do algoritmo. Como todas instâncias utilizadas são do mesmo tamanho, o tempo de execução do algoritmo foi fixado em 40 segundos. Para o algoritmo GA-IG, quatro parâmetros foram calibrados: $Psize$ (tamanho da população), $prob_s$ (probabilidade para a seleção de soluções dominantes), $prob_c$ (probabilidade de crossover) e $prob_m$ (probabilidade de mutação). Para cada parâmetro, os seguintes valores foram testados: $Psize \in \{100, 150, 200\}$; $prob_s \in \{0{,}3, 0{,}5, 0{,}8\}$; $prob_c \in \{0{,}8, 1{,}0\}$; $prob_m \in \{0{,}1, 0{,}2, 0{,}3\}$.

Para determinar a melhor combinação de parâmetros, testes computacionais foram realizados. Todos os resultados foram analisados utilizando um teste de Análise de Variância (ANOVA) [Montgomery(2008)], utilizando as duas medidas, *hypervolume* e *epsilon+*, como variáveis resposta. Os resultados dos testes não são apresentados por falta de espaço. Os melhores parâmetros encontrados para o algoritmo GA-IG foram: $Psize$ = 100, $prob_s$ = 0,5, $prob_c$ = 1,0 e $prob_m$ = 0,1.

O algoritmo MMOEASA é re-implementado seguindo o artigo original de [Banos et al.(2013)]. O algoritmo usa uma população de tamanho 40, probabilidades de crossover e mutação igual a 2,5.

O algoritmo NSGAII é adaptado para o PRVJT bi-objetivo. Neste algoritmo são usados, população de tamanho 100, probabilidade de crossover 100% e probabilidade de mutação 10%.

Os algoritmos MMOEASA e NSGAII também utilizam 40 segundos de execução como condição de parada.

4.4. Comparação dos Algoritmos

Nesta seção comparamos os resultamos dos algoritmos GA-IG, MMOEASA e NSGAII. Para avaliar os algoritmos, as duas métricas são utilizadas: *hypervolume* e *epsilon+*, definidas, respectivamente, nas Equações (4) and (5). Cada instância do problema foi resolvida 30 vezes por cada algoritmo. Para cada algoritmo i, determina-se o conjunto D_i de soluções dominantes obtidas nas 30 rodadas. Estes conjuntos são comparados com o conjunto de soluções referência *Ref* que é formado pelas soluções dominantes obtidos por todos os algoritmos.

Dois tipos de experimentos são realizados. No primeiro experimento, comparam-se os resultados dos algoritmos na minimização dos objetivos (f_1, f_2) = (distância total, desequilíbrio das distâncias). No segundo experimento, os algoritmos são comparados considerando os objetivos (f_1, f_3) = (distância total, desequilíbrio das cargas).

4.4.1. Experimentos com os objetivos f_1 e f_2

Neste seção, os resultados dos algoritmos, obtidos na minimização dos objetivos f_1 e f_2, são comparados através das duas métricas. Nas Tabelas 1 e 2 são apresentados, para cada classe de instância e para cada algoritmo, os valores médios das métricas *hypervolume* e *epsilon+*, respectivamente. Nas Tabelas observa-se que o algoritmo GA-IG obteve as menores médias, das medidas $hypervolume$ e $epsilon+$, para todas as classes de instâncias. O algoritmo MMOEASA apresentou as piores médias.

Para validar os resultados, um teste ANOVA paramétrico é aplicado utilizando os valores das duas métricas $H(D_i)$ e $I_{\varepsilon+}(D_i, Ref)$. Para aplicar o teste foram verificadas as três pressuposições: normalidade, igualdade de variância e a independência dos resíduos. Dado que o valor-P do teste resultou em 0,00 e este valor é menor que 0,05, pode-se concluir que as diferenças dos resultados são estatisticamente significativas.

As Figuras 3 e 4, mostram os gráficos de médias resultantes do teste Tukey da Diferença Honestamente Significativa (HSD) com nível de confiança de 95% para as os algoritmos testados, considerando as métricas *hypervolume* e *epsilon+*, respectivamente. Nas duas Figuras observa-se que, os intervalo do algoritmo GA-IG não sobrepõe nenhum outro intervalo, o que significa que a diferença deste algoritmo é significativa com relação aos outros algoritmos. Pode-se observar também que, o GA-IG é o melhor algoritmo (apresenta as menores médias) e NSGAII é o segundo melhor algoritmo.

Tabela 1: Médias do *hypervolume* ($H(D_i)$).

Problemas	GA-IG	MMOEASA	NSGAII
C1	**7,6**	30,1	15,0
C2	**1,3**	34,6	6,4
R1	**7,2**	30,7	16,0
R2	**4,5**	49,2	18,0
RC1	**6,9**	28,0	15,5
RC2	**4,8**	41,7	19,3
Média	**5,4**	35,7	15,0

Tabela 2: Médias do *epsilon+* ($I_{\varepsilon+}$).

Problemas	GA-IG	MMOEASA	NSGAII
C1	**53,3**	286,6	81,4
C2	**19,1**	582,7	91,0
R1	**35,6**	214,3	97,2
R2	**60,1**	651,0	228,9
RC1	**43,3**	232,0	102,5
RC2	**79,6**	707,7	318,0
Média	**48,5**	445,7	153,2

Figura 3: Médias e Intervalos HSD de Tukey com nível de confiança de 95% - *Hypervolume*.

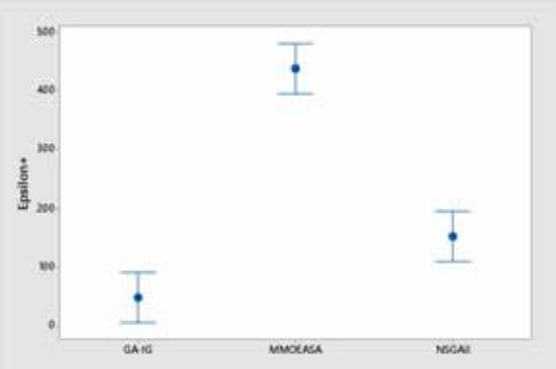

Figura 4: Médias e Intervalos HSD de Tukey com nível de confiança de 95% - *Epsilon+*.

4.4.2. Experimentos com os objetivos f_1 e f_3

Esta seção apresenta os resultados dos algoritmos testados na minimização do par de objetivos f_1 e f_3 (distância total e equilíbrio das cargas). Para cada algoritmo, as médias das medidas de *hypervolume* e *epsilon+* são apresentadas nas Tabelas 3 e 4, respectivamente. Nestes testes, o algoritmo GA-IG também apresentou as melhores médias para todos as classes de instâncias. Das 56 instâncias testadas, o GA-IG apresentou o melhor desempenho em todas instâncias considerando a métrica de *hypervolume*. Já considerando a medida *epsilon+*, o NSGAII apresentou o melhor resultado em 1 instância.

Para as métricas *hypervolume* e *epsilon+*, as Figuras 5 e 6, respectivamente, mostram os gráficos de médias e intervalos HSD de Tukey com nível de confiança de 95% para as os algoritmos comparados no par de objetivos f_1 e f_3. Nestas Figuras podemos observar que, o algoritmo proposto GA-IG tem um desempenho significativamente superior em comparação os demais algoritmos.

5. Conclusões

Neste trabalho foi abordado o problema de Roteamento de Veículos com Janelas de Tempo considerando a otimização de dois objetivos simultaneamente (distância total e equilíbrio das distâncias) e (distância total e equilíbrio das cargas). Com a tentativa de encontrar soluções Pareto-ótimas, foi proposto um algoritmo *memético*, chamado GA-IG. É um algoritmo genético baseado no conceito de dominância de soluções e utilizando a heurística *Iterated Greedy* como um procedimento de intensificação. O algoritmo foi testado em instâncias disponíveis na literatura e foi comparado com dois algoritmos genéticos da literatura. Após os experimentos computacionais realizados e a análise estatística dos resultados, pode-se concluir que o algoritmo proposto GA-IG apresentou um

Tabela 3: Médias do *hypervolume* ($H(D_i)$).

Problemas	GA-IG	MMOEASA	NSGAII
C1	**11,9**	30,9	29,9
C2	**2,8**	74,3	11,4
R1	**5,6**	33,5	22,6
R2	**6,2**	62,0	26,1
RC1	**5,5**	29,8	17,0
RC2	**5,2**	49,6	21,0
Média	**6,2**	46,7	21,4

Tabela 4: Médias do *epsilon+* ($I_{\varepsilon+}$).

Problemas	GA-IG	MMOEASA	NSGAII
C1	**48,3**	319,9	126,2
C2	**20,2**	557,5	81,1
R1	**30,9**	240,3	119,2
R2	**51,2**	504,6	206,5
RC1	**47,1**	361,5	114,2
RC2	**75,9**	722,7	294,9
Média	**45,6**	451,1	157,0

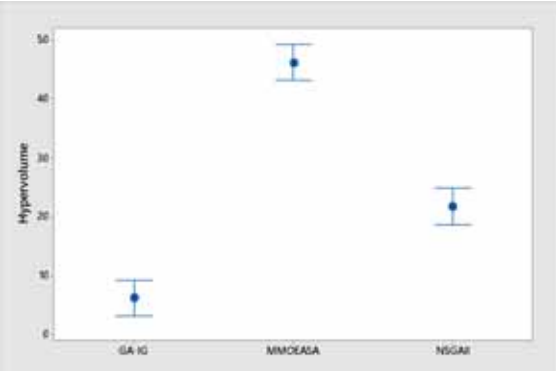

Figura 5: Médias e Intervalos HSD de Tukey com nível de confiança de 95% - *Hypervolume*.

Figura 6: Médias e Intervalos HSD de Tukey com nível de confiança de 95% - *Epsilon+*.

excelente desempenho na resolução do problema. O algoritmo apresentou comportamentos similares nos dois pares de objetivos testados. Como estudos futuros, sugere-se testar outros objetivos, tais como a minimização do número de veículos, e aplicar o algoritmo GA-IG para minimizar, simultaneamente, três objetivos.

Agradecimentos

Este trabalho foi parcialmente financiado pela CNPq, CAPES e FAPEMIG.

Referências

Alvarenga, G. B., Mateus, G. R., and De Tomi, G. (2007). A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34(6):1561–1584.

Banos, R., Ortega, J., Gil, C., Marquez, A. L., and De Toro, F. (2013). A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. *Computers & Industrial Engineering*, 65(2):286–296.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

Garcia-Najera, A. and Bullinaria, J. A. (2011). An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 38(1):287–300.

Homberger, J. and Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238.

Jozefowiez, N., Semet, F., and Talbi, E.-G. (2006). Enhancements of nsga ii and its application to the vehicle routing problem with route balancing. In *Artificial evolution*, pages 131–142. Springer.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986.

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300.

Montgomery, D. C. (2008). *Design and analysis of experiments*. John Wiley & Sons.

Moscato, P. and Cotta, C. (2003). A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, pages 105–144. Springer.

Ombuki, B., Ross, B. J., and Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24(1):17–30.

Rahoual, M., Kitoun, B., Mabed, M.-H., Bachelet, V., and Benameur, F. (2001). Multicriteria genetic algorithms for the vehicle routing problem with time windows. In *4th Metaheuristics International Conference*, pages 527–532.

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Srinivas, N. and Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248.

Tan, K., Chew, Y., and Lee, L. (2006). A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications*, 34(1):115–151.

Toth, P. and Vigo, D. (2001). *The vehicle routing problem*. Siam.

Zitzler, E., Laumanns, M., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L., and Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2):117–132.