RAFAEL DE FREITAS AQUINO

ALGORITMOS DE OTIMIZAÇÃO MULTI-OBJETIVO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2015

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
A657a
2015

Aquino, Rafael de Freitas, 1990-
Algoritmos de otimização multi-objetivo para o problema de Roteamento de Veículos com janelas de tempo / Rafael de Freitas Aquino. – Viçosa, MG, 2015.
x, 76f. : il. (algumas color.) ; 29 cm.

Orientador: José Elias Claudio Arroyo.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f.70-76.

1. Otimização matemática. 2. Heurística. 3. Logística. 4. Levantamentos de rotas - Modelos matemáticos. 5. Veículos a motor - Frotas. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-graduação em Ciência da Computação. II. Título.

CDD 22. ed. 519.6

RAFAEL DE FREITAS AQUINO

ALGORITMOS DE OTIMIZAÇÃO MULTI-OBJETIVO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 20 de novembro de 2015.

André Gustavo dos Santos

Alan Robert Resende de Freitas

José Elias Cláudio Arroyo
(Orientador)

Agradecimentos

Agradeço ao meu orientador, aos meus colaboradores, à seção administrativa, à CAPES, que financiou minha pesquisa, aos meus amigos e à minha família.

Sumário

Lista de figuras

Lista de tabelas

Lista de Algoritmos

Resumo

AQUINO, Rafael de Freitas, M.Sc., Universidade Federal de Viçosa, Novembro de 2015. **Algoritmos de otimização multi-objetivo para o problema de roteamento de veículos com janelas de tempo.** Orientador: José Elias Cláudio Arroyo

O Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT) é uma variação do problema clássico de Roteamento de Veículos em que as demandas dos clientes devem ser atendidas dentro de uma janela de tempo estabelecida. Neste trabalho, aborda-se o PRVJT objetivando 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 duas abordagens heurísticas. A primeira abordagem é baseada na meta-heurística Iterated Local Search, que utiliza uma etapa de intensificação a qual consiste na combinação de soluções dominantes. A segunda abordagem é um algoritmo genético com uma fase de intensificação baseada na heurística de busca local Iterated Greedy, que consiste em melhorar as soluções dominantes. Os desempenhos dos algoritmos heurísticos foram testados em um conjunto de instâncias disponíveis na literatura, denominado Solomon's benchmarks, e os resultados foram comparados com os resultados de dois algoritmos multiobjetivos da literatura. Os resultados obtidos foram analisados estatisticamente e observou-se um desempenho superior dos algoritmos propostos.

Abstract

AQUINO, Rafael de Freitas, M.Sc., Universidade Federal de Viçosa, November, 2015. **Multi-objective optimization algorithms for the vehicle routing problem with time windows.** Adiviser: José Elias Cláudio Arroyo

The Vehicle Routing Problem with Time Windows (VRPTW) is a variant 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 traveled and the imbalance in the loads of the vehicles. Since the problem is NP-Hard, in order to find near Paretooptimal solutions, two heuristic approaches were proposed. The first approach is based on the meta-heuristic Iterated Local Search that uses an intensification stage that consists in combination of non-dominated solutions. The second approach is a genetic algorithm with an intensification stage based on the local search heuristic Iterated Greedy that consists in improving the non-dominated solutions. The heuristic algorithms' performance was tested with a set of problems available in the literature, known as Solomon's benchmarks, and the results were compared with two multi-objective algorithms in the literature. The results were statistically analyzed and revealed superior performance of the proposed algorithms.

1. Introdução

O problema de Roteamento de Veículos (PRV) - conhecido como *Vehicle Routing Problem* (VRP) na língua inglesa - é um problema clássico e complexo de otimização combinatória. Nesse problema, temos uma frota de veículos que deve suprir a demanda de um conjunto de N clientes; cada veículo k deve atender um número de clientes sem que a soma das demandas destes clientes ultrapasse sua capacidade *Q*. O problema consiste em determinar as rotas dos veículos de forma a minimizar um ou mais objetivos, entre os quais temos: distância total percorrida, número de veículos necessários, tempo total do transporte, desequilíbrio das distâncias das rotas, desequilíbrio das cargas dos veículos, em meio a outros.

O PRV tem grande importância, pois possui grande aplicação em problemas reais de entregas de sistemas de distribuição e logística (Baldacci *et al*., 2008). Além disso, os sistemas de distribuição representam aproximadamente de 10% a 20% dos custos finais dos bens distribuídos (Toth &Vigo, 2001). Ainda analisando os custos, estima-se que os custos de distribuição sejam responsáveis por quase metade do total dos custos de logística (De Backer *et al*., 1997).

O PRV é bastante estudado na literatura, sendo considerado um dos problemas mais difíceis da otimização combinatória, tendo um considerável impacto econômico em todos os sistemas de logística (Alvarenga *et al*., 2007). Dantzig and Ramser (1959) foram os primeiros a formular o PRV na literatura científica ao estudaram distribuição de gasolina em estações de venda de combustível. Após esse primeiro trabalho, diversas variantes do problema surgiram, provocando o reajuste de certos aspectos do PRV de acordo com as circunstâncias envolvidas. Algumas das principais variações são:

- PRV Capacitado: todos os clientes solicitam entregas e as demandas são conhecidas *a priori* e não podem ser divididas. Os veículos são idênticos e existe um único depósito, sendo impostas somente restrições de capacidade dos veículos.

- PRV com Coleta e Entrega: é uma generalização do PRV capacitado, na qual cada cliente tem uma entrega e uma coleta, alterando um pouco as restrições dos veículos, cuja capacidade não pode ser ultrapassada em nenhum ponto da rota.

- PRV com Janelas de Tempo (PRVJT): apresenta as mesmas restrições do PRV Capacitado, com o agravante de que a cada cliente deve ser adicionado um intervalo de tempo (janela de tempo). O veículo deve iniciar o serviço com o cliente dentro da janela de tempo e permanecer até que o atendimento termine. Se o veículo chegar antes do início da janela de tempo, ele deve esperar o seu começo.

- PRV com múltiplos depósitos: difere do PRV Capacitado, no quesito de que, além de definir a rota dos clientes, deve ser determinado o depósito do qual sairão as entregas. Os veículos podem reabastecer em outros depósitos, porém devem sempre retornar no final de suas rotas ao depósito de origem.

Neste trabalho, abordamos o PRVJT. Várias situações-problema diferentes foram estudadas dentro desse problema, sendo a distância total percorrida uma dos mais abordadas na literatura (Laporte *et al*., 2000). No entanto, a maioria dos trabalhos considera a otimização de um único objetivo (abordagem mono-objetivo). Assim, este trabalho propõe o estudo do PRVJT multiobjetivo. Foram consideradas duas abordagens baseadas em meta-heurísticas. A primeira é baseada na meta-heurística Iterated Local Search (Lourenço *et al*., 2003). Como busca local, foram utilizadas a heurística Random Variable Neighborhood (Mladenović & Hansen, 1997) e uma etapa de intensificação através da combinação de soluções para gerar novas soluções não dominadas a fim de aplicar novamente a busca local. A segunda é um Algoritmo Memético (Moscato & Cotta, 2003), o qual é baseado no algoritmo genético e utiliza como etapa de intensificação o método de busca local Iterated Greedy Ruiz and Stützle (2007).

1.1. Motivação

No mundo contemporâneo, existem diversos problemas nos quais o número de possibilidades de soluções válidas é muito grande, e, com a

tecnologia atual, muitas vezes, é inviável encontrar exatamente a melhor solução. Todavia, é possível gerar boas aproximações que facilitam a resolução dos problemas que afligem a sociedade atual, envolvendo transporte, prestação de serviço, coleta de produtos, aviação, entre outros. Isto mostra a real importância de estudos para obtenção de algoritmos que consigam chegar a boas respostas em um tempo computacional viável. O PRVJT é um problema pertencente à classe dos problemas NP-Difícil (Lenstra & Kan, 1981). Encontrar soluções eficientes para problemas dessa classe é um desafio para os pesquisadores. Essa dificuldade é demonstrada no grande volume de trabalhos sobre o PRVJT. Sua variação multiobjetivo para o caso de otimização aumenta ainda mais o desafio.

Para a sociedade, o PRVJT é um problema de grande impacto econômico. Toth & Vigo (2001) declaram que utilizar procedimentos computacionais no planejamento de distribuição resulta em uma economia de 5% a 10% nos custos de transporte. Essa redução de custo de transporte faz com que o valor de diversos produtos possam ser reduzidos. Deve ser levado em consideração que os planejamentos assistidos por procedimentos computacionais elevam o nível de satisfação dos clientes, com o serviço, que passa a ser executado de maneira mais econômica e eficiente.

1.2. Objetivos da Dissertação

O objetivo principal deste trabalho é desenvolver algoritmos heurísticos eficientes para o Problema de Roteamento de Veículos com Janelas de Tempo, considerando a otimização de mais de uma função-objetivo. Os objetivos específicos do trabalho são:

- fazer uma revisão de literatura sobre o PRVJT multiobjetivo e suas estratégias de resolução;
- estudar heurísticas e meta-heurísticas para a resolução do problema de otimização multiobjetivo;
- desenvolver algoritmos heurísticos para resolver o PRVJT multiobjetivo;

• avaliar o desempenho dos algoritmos desenvolvidos, compará-lo com os resultados da literatura, demonstrando a performance das heurísticas propostas.

1.3. Organização do Trabalho

Este trabalho está organizado da seguinte maneira:

• No capítulo 2, é apresentada a descrição genérica de um problema de otimização combinatória multiobjetivo, são apresentados os conceitos básicos e as métricas mais utilizadas para avaliar a qualidade das soluções.

• No capítulo 3, são apresentados a descrição do Problema de Roteamento de Veículos com Janelas de Tempo, a definição do problema, o modelo matemático do problema mono-objetivo, os problemas-teste utilizados, a descrição dos objetivos a serem otimizados e a revisão de trabalhos que abordam a otimização multiobjetivo do PRVJT.

• No capítulo 4, é apresentada a descrição completa dos algoritmos propostos.

• No capítulo 5, são apresentados uma descrição de como os testes foram realizados, as características do computador em que eles foram aplicados e os resultados obtidos, juntamente com a análise destes.

• O capítulo 6 contém as conclusões dos resultados obtidos, as considerações finais e a proposta de trabalhos futuros.

2. Otimização combinatória Multi-objetivo

Em problemas de otimização combinatória, temos como objetivo encontrar os menores valores para uma função-objetivo. Estes problemas possuem variáveis discretas, portanto são chamados problemas combinatórios, que geralmente envolvem buscas por um elemento em um conjunto finito ou, em alguns casos, infinito contável, cujas principais ocorrências envolvem um subconjunto de variáveis, uma permutação, ou um gráfico (Papadimitriou & Steiglitz, 1998).

Em otimização combinatória multi-objetivo, não consideramos apenas uma única função como em otimização mono-objetiva. Nesse caso, $f(x)$ se torna um vetor de m objetivos: $f(x)$ = ($f_1(x)$, ..., $f_m(x)$). Geralmente um problema de otimização multi-objetivo é composto de m funções objetivo, v variáveis de decisão e r restrições. Sem perda de generalidade, consideramos a minimização de todos os problemas circunscritos a cada objetivo. O problema pode ser formulado como:

$$\text{Minimizar } y = f(x) = (f_1(x), ..., f_m(x))$$
$$\text{Sujeito a } e(x) = (e_1(x), e_2(x), ..., e_r(x)) \leq 0$$
$$x = (x_1, x_2, ..., x_v) \in X$$
$$y = (y_1, y_2, ..., y_m)) \in Y$$

Em que x é o vetor de variáveis de decisão, y = $f(x)$ é o vetor de funções objetivo, X é o espaço de decisão, Y o espaço de objetivos e $e(x)$ é um vetor de restrições. Se para um dado x determina-se o valor mínimo das m funções, $f_1(x)$,...,$f_m(x)$, então teremos a solução ideal do problema. Para cada solução x em X, existe um ponto $f(x)$ no espaço objetivo Y. O caso mais interessante da otimização multiobjetivo é quando temos objetivos conflitantes, isto é, a otimização de um dos objetivos causa a deterioração dos outros objetivos. Nesse caso, teremos um conjunto de soluções como resultado do processo de otimização, conforme é mostrado na Figura 1 para o caso biobjetivo. Nessa figura, podemos observar, à esquerda X, o espaço de decisão e, à direita Y, o

espaço de objetivos. Então, em otimização multiobjetivo, tem-se um conjunto de soluções no espaço X e um conjunto de pontos no espaço Y. O problema consiste em determinar o conjunto de soluções que otimizam simultaneamente todos os objetivos (Soluções Pareto-ótimas).

2.1. Dominância e Pareto-ótimo

Em otimização multiobjetivo, são potencializados dois ou mais objetivos simultaneamente. Esses objetivos podem ser conflitantes entre si, o que implica que não seria suficiente uma otimização mono-objetivo. Em otimização multiobjetivo, o conceito de otimalidade baseia-se no conceito de dominância introduzido por Edgeworth, em 1881, e depois generalizado por Vilfredo Pareto, em 1896.

O conceito de dominância de Pareto consiste em fornecer uma forma de comparar duas soluções viáveis para o problema. Considerando a minimização de todos os objetivos e duas soluções viáveis x e y, diz-se que x domina y ($x \preceq y$) se as condições abaixo forem respeitadas:

- A solução x é melhor ou igual à y em todas as funções objetivo, $f_i(x) \leq f_i(y)$, $\forall i \in \{1,..., m\}$.
- A solução x é melhor que a solução y em pelo menos uma função objetivo, $\exists j \in \{1,..., m\}$ tal que $f_j(x) < f_j(y)$.

Uma solução que não é dominada por nenhuma solução do espaço X é chamada de Pareto-ótimo. O conjunto de soluções não dominadas é chamado de Conjunto Pareto-ótimo. Os respectivos pontos no espaço objetivo determinam uma fronteira denominada Fronteira Pareto-ótima. Na Figura 1, destacam-se em azul as soluções Pareto-ótimas e a respectiva Fronteira Pareto-ótima.

Os métodos exatos de otimização multiobjetivo determinarão as soluções Pareto-ótimas. As soluções retornadas por algoritmos heurísticos podem ser classificadas como soluções não dominadas (ND). Esse conjunto de soluções é composto pelas melhores soluções encontradas por um algoritmo, isto é, as soluções encontradas não dominadas. O objetivo dos algoritmos heurísticos é determinar soluções que estejam as mais próximas possíveis das soluções Pareto-ótimas.

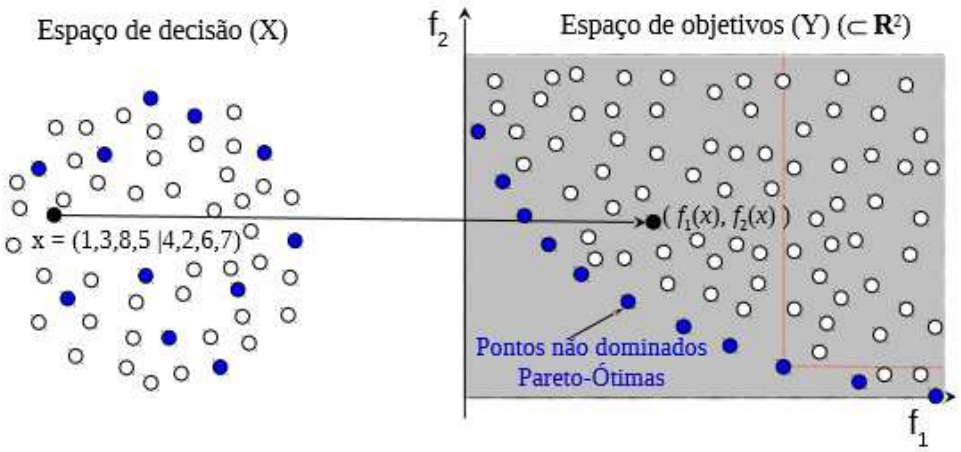

Figura 1 - Solução x do espaço de soluções mapeada no espaço de objetivos, com m = 2

2.2. Métricas de Análise de Qualidade

Problemas multiobjetivos geralmente têm como solução um conjunto contendo as melhores soluções encontradas. Esse fato acarreta dificuldade ao se realizar uma análise dos resultados. É necessário utilizar métodos para comparar os algoritmos de uma maneira significativa que expresse corretamente a qualidade das soluções encontradas. Neste trabalho, foram adotados dois métodos comumente utilizados na literatura: o *hypervolume* (Zitzler & Thiele, 1998) e o *epsilon+* (Zitzler *et al*., 2003), tendo em vista que ambas as métricas lidam corretamente com as comparações de soluções multiobjetivo.

Para comparar os resultados de dois ou mais algoritmos, para uma determinada instância do problema, primeiramente é necessário definir um conjunto de referência, *Ref*, chamado de "melhor Pareto-ótimo conhecido", que contém as soluções não dominadas encontradas por todos os algoritmos avaliados. Supondo que ND_1, ND_2 e ND_3 sejam conjuntos de soluções não dominadas de três algoritmos, o conjunto de referência *Ref* é composto pelas soluções não dominadas de todos os três conjuntos, Ref = Soluções não dominadas de ($ND1 \cup ND2 \cup ND3$). Esse conjunto é utilizado nas métricas *hypervolume* e *epsilon+.*

2.2.1. *Hypervolume*

A métrica denominada *hypervolume* (Zitzler & Thiele, 1998) consiste em calcular o tamanho do espaço coberto por um conjunto de soluções não dominadas, limitadas por um ponto de referência adequado. A Figura 2, para o caso biobjetivo apresenta graficamente como é feito o cálculo do *hypervolume* correspondente a um conjunto ND = {s1, s2, s3}. Na figura, os pontos s1, s2, s3 cobrem a área preta limitada pelo ponto de referência P. O *hypervolume* é a união das áreas dos retângulos formados por cada um desses pontos e o ponto P. Para problemas de maximização, o ponto de referência pode ser definido como a origem (0,0). Já em problemas de minimização, os valores do ponto P de referência devem exceder o máximo de cada objetivo. Essa métrica é utilizada para comparar dois ou mais algoritmos.

Como os valores de *hypervolume* resultam em grandes números, calculamos o desvio percentual relativo de um conjunto de soluções não dominadas ND_i com relação ao conjunto *Ref*. Essa métrica é definida da seguinte maneira:

$$H(ND_i)\% = 100 \times \frac{H_{Ref} - H_{ND_i}}{H_{Ref}} \qquad (1)$$

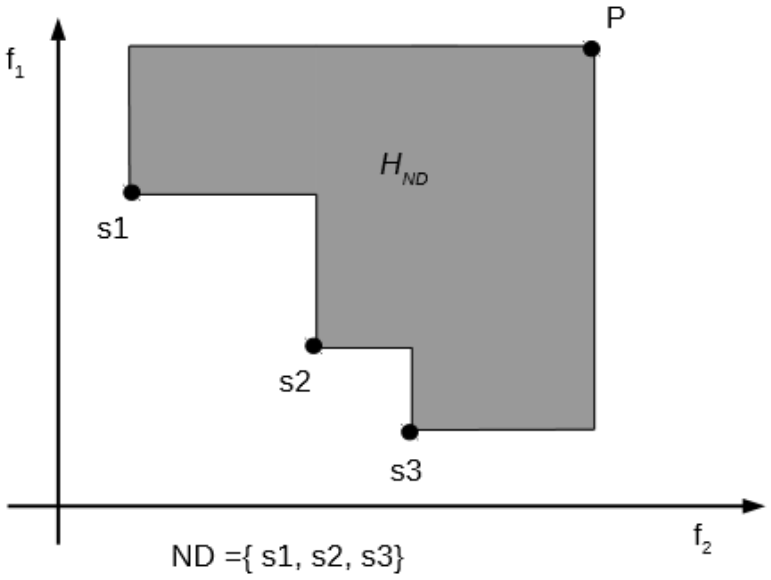

Figura 2 -Exemplo do cálculo do *hypervolume*.

Em que H_{Ref} e H_{NDi} são os *hypervolumes* correspondentes aos conjuntos Ref e ND_i, respectivamente. Menores valores de $H(ND_i)$% correspondem a soluções de maior qualidade no conjunto ND_i, pois sua aproximação do conjunto *Ref* é maior, ou seja, a cobertura do espaço feita por esse conjunto é melhor.

2.2.2. *Epsilon*

A métrica denominada *epsilon*+ (Ie+) (Zitzler *et al*. (2003)), de maneira diferente da métrica anterior visa comparar 2 conjuntos não dominados. A métrica consiste em definir o menor valor de ε tal que um conjunto $ND1$ seja melhor ou igual a um segundo conjunto $ND2$ acrescido de ε em todos os objetivos de todas as suas soluções.

Para comparar mais de dois conjuntos de soluções, considera-se o conjunto de referência Ref, de modo que cada conjunto será comparado com o conjunto Ref. O conjunto a obter menores valores de Ie+ será o mais próximo de Ref e, portanto, o melhor conjunto. Observe na Figura 3 que o conjunto *Ref* domina o conjunto ND. O valor de ε nessa figura é exatamente o suficiente para que *Ref* acrescido de ε seja igual ou pior que ND em todas as soluções e em todos objetivos. Abaixo temos a fórmula utilizada para calcular o *epsilon+.*

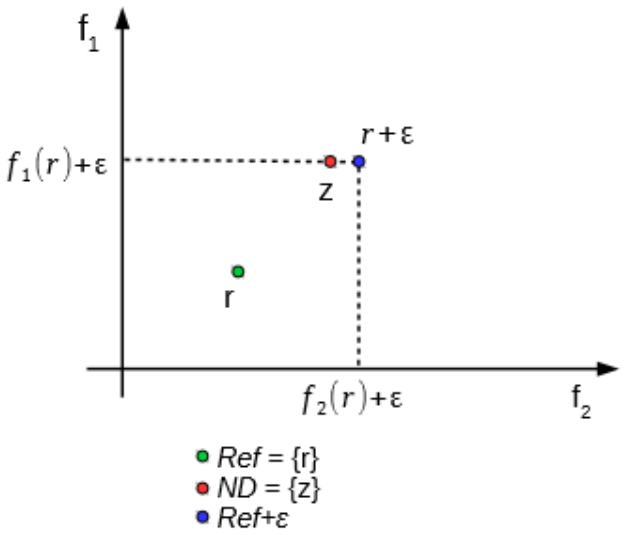

Figura 3 - Exemplo do cálculo do *epsilon*+.

$$I_{\varepsilon+}(D_i, Ref) = \max_{z^2 \in Ref} \{\min_{z^1 \in D_i} \{\max_{1 \le i \le m} \{z_i^1 - z_i^2\}\}\} \quad (2)$$

Onde $z^1 = (z_1^1, ..., z_m^1)$ e $z^2 = (z_1^2, ..., z_m^2)$ são pontos no espaço objetivo e *m* é o número de objetivos minimizados.

2.3. Principais Algoritmos da Literatura

Métodos multi-objetivo geralmente enfocam dois aspectos: aproximar a distância da fronteira Pareto-ótima e manter a diversidade da população. Os principais métodos utilizados para problemas multiobjetivos são baseados em algoritmos genéticos. Uma das dificuldades é atribuir valor de *fitness* para as soluções, levando em conta que a diversidade da população também deve ser mantida, ou seja, é necessário evitar que a população fique toda idêntica (Zitzler *et al*., 2004). A maneira de se atribuir o fitness e a forma de diversificar as soluções são as principais diferenças existentes nos algoritmos genéticos multiobjetivo. Nas próximas subseções, descrevemos métodos heurísticos amplamente utilizados para resolver problemas multiobjetivo.

2.3.1. *Strength Pareto Evolutionary Algorithm 2* (SPEA2)

O algoritmo SPEA2 foi proposto por Zitzler *et al*. (2001). O SPEA2 é uma abordagem evolutiva que utiliza duas populações P e Q, sendo P a população gerada pelas operações evolutivas. A população Q é denotada como população externa, apenas com soluções não dominadas encontradas pelo algoritmo. Denotamos por P_t e Q_t as populações P e Q na geração t.

A Figura 4 apresenta o funcionamento do algoritmo SPEA2. Na inicialização, as populações são criadas, Q_0 é povoada com a população inicial, sabendo que P_0 corresponde à população de filhos vazia. Posteriormente, o valor do *fitness* é calculado para os indivíduos em P_t e Q_t. Na seleção, os indivíduos não dominados por $P_t \cup Q_t$ são adicionados em Q_t+1. Caso o número de soluções seja maior que a população, um corte é realizado para reduzir o número de soluções. O critério de parada é avaliado. Caso não tenha sido alcançado, o cálculo do algoritmo continua. Na seleção de pares para o *crossover*, é utilizado um torneio binário em Q_t+1 para selecionar os pais necessários. Em seguida, os operadores evolutivos de *crossover* e mutação são aplicados, gerando a

população P_t+1. Após esse procedimento, o algoritmo volta para a função de atribuição de *fitness*.

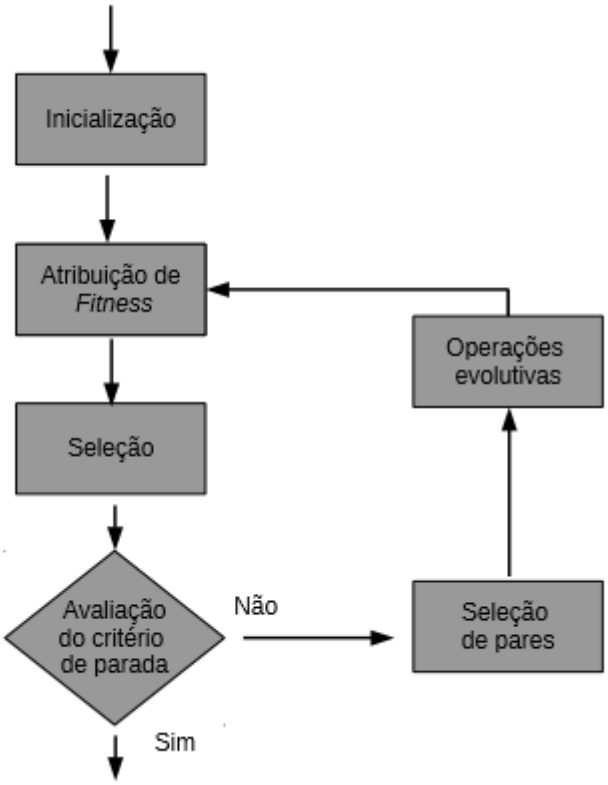

Figura 4 - Fluxograma do algoritmo SPEA2.

A função de *fitness* indica quais indivíduos são mais aptos a sobreviverem. Nesse algoritmo, é utilizado o número de indivíduos dominados (Domination count, (Zitzler *et al*., 2001)) como *fitness*. Esse valor é calculado avaliando as soluções de ambas as populações P e Q.

O método de corte reduz o número de indivíduos da população. Essa redução auxilia na preservação da diversidade da população. Posteriormente, é realizado o cálculo da média das distâncias euclidianas no espaço objetivo. Essa média é calculada entre as duas soluções mais próximas entre si. A solução com menor média é removida. Essa remoção faz com que o algoritmo mantenha uma fronteira de soluções não dominadas com maior variedade.

2.3.2. *Non-dominated Sorting Genetic Algorithm II* (NSGA II)

Quase todos os algoritmos genéticos multiobjetivo de maior sucesso são baseados no *Non-dominated Sorting Genetic Algorithm II* (NSGA II), de Deb *et al*. (2002). Diferentemente do SPEA2, o NSGA II não precisa armazenar as soluções não dominadas, pois é elitista, sempre preservando somente as melhores soluções na população.

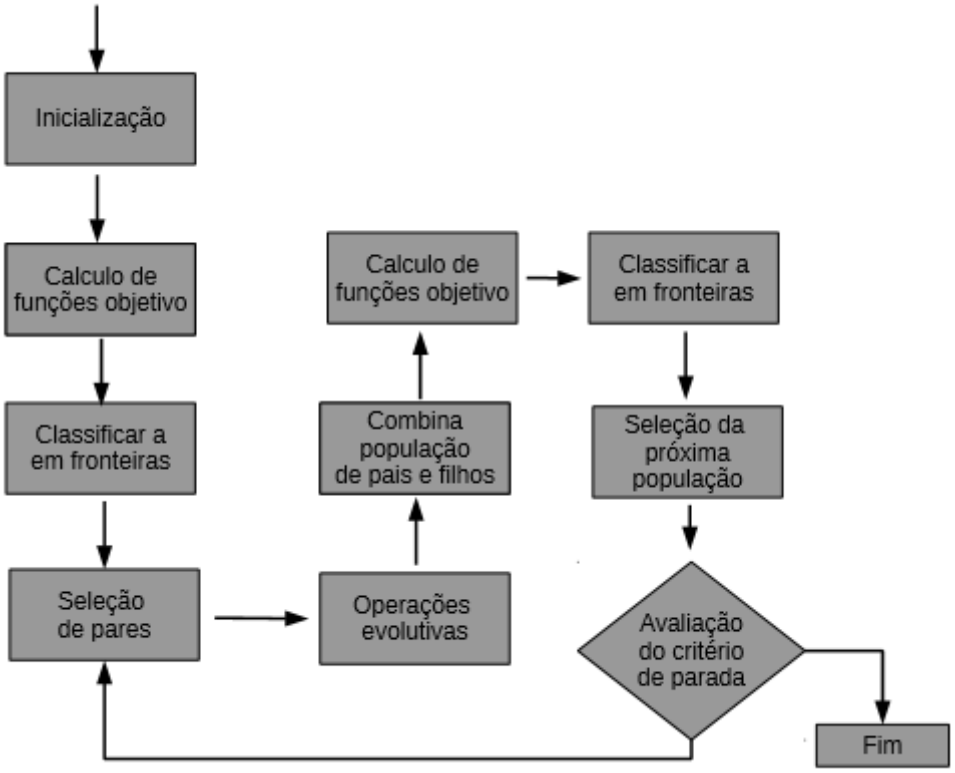

Figura 5 - Fluxograma do algoritmo NSGAII.

Na Figura 5, temos o fluxograma do algoritmo NSGAII. A população inicial é inicializada como P_0, suas funções-objetivo são calculadas e ela é classificada em fronteiras. Após esse início, o *loop* principal do algoritmo começa e os pares de pais são selecionados utilizando torneio binário em P_t, em que a classificação de fronteira e a medida *crowding distance* são utilizadas como critérios de seleção. Após esse processo, as operações evolutivas de *crossover* e mutação são aplicadas, gerando a população de filhos Q_t. Em seguida, as populações de pais e filhos são combinadas e classificadas em fronteiras. A seleção é feita de maneira elitista, utilizando a classificação em fronteira e a medida *crowding distance* quando necessário. Caso o critério de parada não tenha sido atingido,

o algoritmo retorna ao passo de seleção dos pares para o *crossover*. Caso contrário, a primeira fronteira da população P atual será o conjunto de soluções não dominadas do algoritmo.

A classificação das soluções é feita utilizando o método chamado *non-dominated sort* (Golberg, 1989). Através desse método, determina-se o *fitness* de cada solução. Os autores desenvolveram o algoritmo para classificar as soluções em tempo O ($m * popSize_2$), sendo m é o número de funções-objetivo. A ideia é classificar as soluções em fronteiras, de acordo com o nível de dominância. Primeiramente, determinam-se as soluções não dominadas da população que compõem as soluções da primeira fronteira; em seguida, excluindo as soluções da primeira fronteira, determinam-se as soluções não dominadas que compõem a segunda fronteira, continuando esse processo até que todas as soluções sejam inseridas em uma fronteira. É utilizado o método de dominância por classificação em *ranks* (Fonseca *et al*., 1993), em que, a cada iteração, as soluções cujo *ranking* é 0 são classificadas com o número da fronteira atual. Para cada solução dessa fronteira, as soluções dominadas por elas têm o ranking reduzido em 1 unidade, assim passando para a próxima iteração.

Como diversificação, o *fitness* é utilizado para escolher os pais que participarão do *crossover*. As soluções que tiverem melhores valores de *fitness* são escolhidas. Esse critério também é utilizado para selecionar as soluções da próxima geração. Para diferentes soluções dentro de uma fronteira, é utilizada uma medida chamada *crowding distance*. Essa medida calcula a distância média entre uma solução da fronteira e suas duas soluções mais próximas. As soluções são ordenadas de acordo com a *crowding distance*, e as soluções mais distantes no espaço possuem maiores probabilidades de serem escolhidas.

2.3.3. Comparação entre NSGA II e SPEA2

Na literatura, foram apresentados diferentes algoritmos genéticos, que podem ser encontrados nos seguintes trabalhos: Deb and Jain (2002), Laumanns *et al*. (2002), Khare *et al*. (2003), Deb *et al*. (2005), Coello & Zacatenco (2006), Konak *et al*. (2006), Kollat & Reed (2006), Knowles *et al*. (2006), Tan *et al*. (2006), Aguirre & Tanaka (2007), Deb (2007), Deb & Tiwari

(2008), Zitzler *et al*. (2001), Deb *et al*. (2002). Em geral, o SPEA2 e o NSGA II possuem desempenho idêntico em convergência e manutenção da diversidade (Knowles *et al*. (2006), Kollat & Reed (2006), Aguirre & Tanaka (2007), Deb & Tiwari (2008)), entretanto a operação de redução de soluções não dominadas do SPEA2 é computacionalmente mais custosa que a função que realiza o cálculo da *crowding distance* do NSGA II (Deb & Jain (2002), Khare *et al*. (2003)).Em outros trabalhos, observou-se o NSGA II convergindo mais rapidamente para o Conjunto Pareto-ótimo do que o SPEA2 (Deb *et al*., 2005).

De acordo com Coello & Zacatenco (2006), devido a seus mecanismos inteligentes, a boa performance do NSGA II o tornou muito popular nos últimos anos, estabelecendo-se como principal algoritmo genético multiobjetivo em comparação aos demais algoritmos.

3. Problema de Roteamento de Veículos com Janelas de Tempo

O Problema de Roteamento de Veículos clássico (PRV) é um dos problemas mais estudados de otimização. Como restrições comuns, temos: capacidade dos veículos, distância máxima que os veículos podem percorrer, tempo de chegada aos clientes, precedência de clientes ao realizar serviços, entre outros. Para instâncias pequenas do PRV, as soluções ótimas podem ser determinadas usando métodos exatos (Gendreau *et al*. (1995), Toth & Vigo (1997), Toth & Vigo (1998), Toth & Vigo (2002), Baldacci *et al*. (2007), Baldacci *et al*. (2010), Kallehauge (2008)). Entretanto, o tempo computacional aumenta de maneira considerável para instâncias maiores (Desrochers *et al*., 1992). Foi demonstrado por Lenstra & Kan (1981) que o PRV é NP-difícil, e não existe nenhum método de tempo polinomial para solucionar esse problema. Para esse tipo de questão, podemos utilizar heurísticas ou métodos exatos para retornar soluções aproximadas para o problema. Neste trabalho, tem-se como objetivo estudar métodos heurísticos; por isso decidiu-se pela utilização essa abordagem.

As heurísticas para o PRV são divididas em três grupos: 1) heurísticas construtivas, 2) heurísticas de melhoria e 3) meta-heurísticas. As heurísticas construtivas são algoritmos que visam criar soluções iniciais. Essas heurísticas constroem uma rota para cada veículo usando regras de decisão para selecionar os clientes a serem inseridos na rota e para determinar a ordem de atendimento.

Uma heurística de melhoria é utilizada para, de maneira iterativa, melhorar uma solução inicial através de movimentos que geram soluções vizinhas (método de busca local).

Por fim, as meta-heurísticas podem ser utilizadas para forçar soluções a escapar de um ótimo local, uma solução que a busca local não consegue melhorar. Geralmente, podem ser utilizados métodos de busca inteligente ou guiada para evitar que os veículos fiquem presos nesses locais.

Alguns estudos listados abaixo fornecem diversas abordagens heurísticas para variantes do PRV: Laporte *et al*. (2000), Gendreau *et al*. (2001), Baldacci *et al*. (2007), Baldacci *et al*. (2010), Laporte (2009). Para o PRV multiobjetivo, temos o estudo dos algoritmos da literatura feito por Jozefowiez *et al*. (2008).

3.1. Revisão Bibliográfica do Problema de Roteamento de Veículos MultiObjetivo

Para o problema PRVJT, existem alguns trabalhos na literatura que consideram a otimização de mais de um objetivo. Rahoual *et al*. (2001) desenvolveu um algoritmo genético para o problema PRVJT baseado no algoritmo NSGA (Srinivas & Deb, 1994), no qual são minimizados o número de rotas e a distância total. Este trabalho considera soluções inválidas, que violam alguma restrição do problema, trabalhando com penalidades sobre tais rotas. As restrições consideradas foram capacidade, distância e duração das rotas, em adição à janela de tempo. Neste algoritmo, a população inicial é gerada de maneira aleatória e é utilizado um operador de *crossover* de ponto único (*single point crossover*, De Jong (1975)) e uma mutação que consiste em mudar um cliente de uma rota para outra de maneira aleatória. Os autores apresentam os resultados de algumas instâncias dos problemas de teste de Solomon, porém muitos deles apresentam soluções inviáveis.

Jozefowiez *et al*. (2008) abordaram o VRP Capacitado, minimizando a distância total e o número de rotas. Eles implementaram uma versão melhorada do NSGA II (Deb *et al*., 2002) utilizando paralelismo. Eles consideram duas operações de *crossover* chamadas *RBX of Potvin and Bengio* (Potvin & Bengio, 1996) e *the Split function of Prins* (Prins, 2004). A operação de mutação foi baseada na operação 2-opt.

Ombuki *et al*. (2006) consideram o PRVJT como um problema biobjetivo, em que o número de veículos e a distância total são minimizados. Eles utilizaram um algoritmo genético no qual 90% da população inicial são gerados de maneira aleatória e os 10% restantes das soluções são gerados pela heurística gulosa do vizinho mais próximo. Os autores introduziram o *crossover* da melhor rota, que tem como foco minimizar o número de veículos e a distância total enquanto mantém a viabilidade das soluções. A mutação utiliza inversão na sequência de clientes.

Tan *et al*. (2006a) propõem uma heurística híbrida baseada em um algoritmo genético para o PRVJT, minimizando o número de veículos e a distância total. Essa abordagem começa gerando uma população inicial aleatória que, então, passa por um *crossover* o qual utiliza troca de rotas e uma mutação de múltiplas operações, sendo elas: *partial swap*, *split route* e *merge routes*, das quais somente uma é executada. Adicionalmente, a busca local *λ-interchange* (Osman, 1993) e as buscas locais propostas chamadas *intra route* e *shortest pf* foram implementadas, sendo executadas a cada 50 iterações em toda a população. A análise encontrou que, nos problemas das categorias C1 e C2, os objetivos não são conflitantes e, nas categorias restantes de problemas R1, R2, RC1, RC2, há objetivos conflitantes. Os autores adaptaram o algoritmo para outros problemas derivados do PRV capacitado, o *Truck and Trailer PRV* (Tan *et al*., 2006a) e o PRV com divisão de entregas (Tan *et al*., 2007).

Ghoseiri & Ghannadpour (2010) apresentaram um estudo utilizando programação por metas (*goal programming*) para formulação do problema e uma abordagem utilizando um algoritmo genético para solucionar essa mesma questão. Os objetivos estudados foram distância total e número de veículos utilizados. Uma parte da população inicial foi inicializada utilizando a heurística I1 (Solomon,1987) e busca local *λ-interchange* (Osman, 1993), e o restante das soluções foi gerado de maneira aleatória. Os autores introduziram o *crossover* da melhor rota em relação ao custo, o qual seleciona melhor rota de cada solução pai, de maneira similar ao *crossover* proposto por Ombuki *et al*. (2006), com pequenas diferenças. O operador de mutação é um operador de recombinação, pois ele seleciona dois arcos de maneira aleatória para removê-los em cada solução, e então, é realizada a troca nas rotas antes e depois do ponto de remoção dos arcos, produzindo duas novas soluções filhas. Duas buscas locais foram incorporadas ao algoritmo genético, sendo elas executadas ao final de cada geração, em uma porção da população. A aproximação de Pareto de cada instância dos problemas de Solomon desse estudo está disponível para consulta.

Pacheco & Martí (2006) estudaram o problema de roteamento de ônibus escolares, que consiste no transporte dos estudantes de suas casas até a escola. Sua abordagem utilizou o método Busca Tabu (Minsky, 1961). Eles

consideraram a minimização do número de ônibus e do tempo máximo que um estudante passa no ônibus, ou seja, do tempo que corresponde à rota mais longa. O problema foi resolvido considerando as duas funções-objetivo separadamente. Como no primeiro objetivo, o número de veículos é um número discreto (depende somente do número de localizações), os autores seguiram um simples método, que consiste em minimizar o segundo objetivo, o tempo máximo de uma rota, isto para cada valor possível de número de veículos. O algoritmo utiliza dois métodos construtivos propostos por Corberán et al. (2002) e um por Fisher and Jaikumar (1981). Em seguida, a Busca Tabu aplica uma versão modificada do *CROSS exchange* proposto por Taillard et al. (1997) como busca local.

Mais recentemente, o trabalho Garcia-Najera e Bullinaria (2011) propõem um algoritmo denominado MOEA (*Multiobjective Evolutionary Algorithm*), baseado no NSGA II, substituindo a *crowding distance* pelo conceito de similaridade. Os objetivos minimizados foram a distância total e o número de veículos. A população inicial é gerada de maneira aleatória. O *crossover* utilizado copia um número de rotas aleatório do primeiro pai, posteriormente completando com as rotas e clientes restantes do segundo pai. Para mutação, foram utilizadas três operações chamadas *reallocation*, *exchange* e *reposition*. O principal conceito introduzido foi o de similaridade de soluções, substituindo a *crowding distance* utilizada no NSGA II. A similaridade serve para melhor medir a proximidade das soluções, e seu uso no algoritmo é incluído na seleção de soluções para o *crossover* e na sobrevivência das soluções para a próxima geração. Essa similaridade é calculada através da contagem do número de arestas iguais, ligando os mesmos vértices (clientes e depósito). O valor da similaridade é 1 quando as soluções são iguais, e 0 quando são completamente diferentes. Para realizar a comparação entre toda a população, a similaridade é calculada pela média da similaridade de solução com todas as outras soluções da população.

Banos et al. (2013) abordaram o PRVJT através de um algoritmo híbrido que utiliza Pareto, algoritmo genético e *simulated annealing* (Kirkpatrick, 1984) chamado MMOEASA. Os objetivos a serem minimizados foram a distância total das rotas e o desequilíbrio da distância entre as rotas e cargas, os mesmos

abordados neste trabalho. Seu estudo foi o PRVJT biobjetivo, tendo como meta principal a distância total. O algoritmo utilizou *crossover* idêntico ao proposto no MOEA e, como mutação, foram utilizadas 10 operações. Entre operações de melhoria de solução e operações de movimentos aleatórios, apenas uma é utilizada por vez. Após esses cálculos, a solução é aceita ou não, utilizando critérios de decisão do *simulated annealing*.

Chiang and Hsu (2014) apresentaram o estudo mais recente de PRVJT multiobjetivo. Nele os autores fazem um estudo das diversas técnicas de algoritmos genéticos multiobjetivo e aplicam o conhecimento adquirido nelas para construir um novo algoritmo. Para inicializar a população, recorre-se a duas heurísticas e, para o restante, são empregadas soluções aleatórias. São utilizadas a heurística gulosa de inserção orientada ao cliente mais próximo em relação ao tempo (Solomon, 1987) e a heurística I1 proposta por Solomon (1987). O *crossover* utilizado é uma versão melhorada do proposto por Tan et al. (2006a). Após o *crossover*, é aplicada uma operação de redução de rotas. Como operadores de mutação, dois métodos são utilizados: reinserção e divisão de rotas. No restante, o algoritmo se assemelha muito ao NSGA II e obteve bons resultados quando comparado aos algoritmos propostos pelo restante da literatura.

3.2. Problema de Roteamento de Veículos com Janela de Tempo

O problema de roteamento de veículos com janela de tempo (PRVJT) é constituído por um conjunto de veículos idênticos em um depósito. Esse depósito opera em uma janela de tempo e é utilizado para atender um conjunto de clientes, suprindo completamente a demanda de cada um deles. Cada cliente também possui uma janela de tempo, que determina o intervalo de tempo que o serviço pode inciar prestando auxílio ao cliente. O tempo máximo da rota de cada veículo não pode ultrapassar o tempo de funcionamento do depósito. Cada cliente deve ser servido por um único veículo. Além disso, a capacidade dos veículos não pode ser ultrapassada.

Neste trabalho, é abordada uma versão biobjetivo do PRVJT, mantendo como objetivo principal a distância total percorrida e variando os objetivos secundários. Os objetivos secundários abordados foram desequilíbrio das

distâncias das rotas e desequilíbrio das cargas, ambos descritos detalhadamente na Seção 3.2.4.

3.2.1. Formulação Matemática

É possível definir o PRVJT em um grafo completo orientado $G = (V,A)$. Temos que $V = \{0, ..., n\}$ é o conjunto de vértices, sendo n o número de clientes. $A = \{(i,j)|i,j \in V\}$ é o conjunto de arcos. Cada arco (i,j) é associado a um tempo t_{ij} e uma distância d_{ij}. Para simplificar o estudo, é considerada uma unidade de distância correspondente a uma unidade de tempo, ou seja, $d_{ij} = t_{ij}$.

Existe um conjunto de K veículos idênticos de capacidade Q e eles devem atender a demanda dos n clientes. O conjunto de vértices $Vc = \{1,..., n\}$ representa os clientes a serem atendidos e o vértice 0 representa o depósito, de onde todos os veículos devem partir e retornar ao final de suas rotas. O depósito não possui demanda, mas sua janela de tempo deve ser respeitada. Cada cliente i possui uma demanda qi que deve ser atendida por um único veículo. Além disso, todos os vértices possuem uma janela de tempo $[a_i,b_i]$, isto é, o serviço no cliente i deve ser iniciado dentro desse intervalo de tempo. É importante observar que chegada do veículo ao cliente i pode ocorrer antes do instante a_i. Caso isso ocorra, o veículo deverá esperar o instante ai para realizar o serviço. O veículo não poderá chegar após o instante b_i, pois isso violaria as restrições de tempo do problema, tornando essa solução inválida. Essa restrição é conhecida na literatura como janela de tempo rígida. O tempo necessário para suprir a demanda de cada cliente i é si.

A formulação matemática mono-objetivo do PRVJT foi adaptada de (Toth e Vigo, 2001) e é apresentada nas equações abaixo. O modelo utiliza uma variável binária x_{ijk} que assume valor 1 quando o veículo k utiliza o arco (i,j), e 0 quando ocorre o caso contrário. A variável ts_{ik} indica o tempo que o veículo k realiza o serviço no cliente i.

$$\text{Minimizar} \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} d_{ij}x_{ijk}$$

$$\text{Sujeito a} \sum_{k\in K}\sum_{i\in V} x_{ijk} = 1 \qquad j \in V_c \qquad (1)$$

$$\sum_{j\in V_c} x_{0jk} = 1 \qquad k \in K \qquad (2)$$

$$\sum_{i\in V_c} x_{ijk} - \sum_{i\in V_c} x_{jik} = 0 \qquad j \in V,\ k \in K \quad (3)$$

$$\sum_{i\in V_c} x_{i0k} = 1 \qquad k \in K \qquad (4)$$

$$\sum_{i\in V_c} q_i \sum_{j\in V_c} x_{ijk} \le Q \qquad k \in K \qquad (5)$$

$$s_i + ts_{ik} + t_{ij} - (1 - x_{ijk})M_{ij} \le ts_{jk} \quad i,j \in V,\ k \in K (6)$$

$$a_i \le ts_{ik} \le b_i \qquad i \in V,\ k \in K \quad (7)$$

$$x_{ijk} \in \{0,1\} \qquad i,j \in V,\ k \in K (8)$$

A primeira expressão representa o custo total (por exemplo, distância total) a ser minimizado. A expressão (1) garante que um veículo k atenda cada cliente j. As expressões (2-4) garantem que a continuidade do caminho percorrido pelo veículo k seja mantida: o veículo parte do depósito, visita os clientes e retorna para o depósito ao final da rota. A restrição imposta pela expressão (5) faz com que cada cliente k atenda somente um conjunto de clientes que não ultrapasse sua capacidade máxima Q. As expressões (6-7) garantem a viabilidade das rotas com relação às restrições de janela de tempo, em que ts_{ik} representa o tempo no qual o veículo k começa a atender o cliente i e M_{ij} equivale a constantes de valor suficientemente grande. Por fim, a expressão (8) define restrições de domínio das variáveis de decisão.

3.2.2. Conjunto de Problemas de *Solomon* para PRVJT

O conjunto de problemas de teste mais utilizado para o PRVJT é o conjunto de problemas de teste de Solomon (1987), que inclui 56 instâncias, todas considerando n = 100 clientes. Essas instâncias são classificadas em 4 tipos de acordo com a forma de agrupação dos clientes. Nas instâncias do tipo C1 (9 instâncias) e C2 (8 instâncias), os clientes são localizados em agrupamentos segundo critérios geográficos. Em instâncias do tipo R1 (12

instâncias) e R2 (11 instâncias), os clientes são distribuídos de maneira aleatória no espaço. Instâncias do tipo RC1 (8 instâncias) e RC2 (8 instâncias) são uma mistura de clientes agrupados e distribuídos de maneira aleatória. A Figura 6 mostra a distribuição geográfica dos clientes de cada categoria.

As instâncias C1, R1 e RC1 têm um horizonte de escalonamento curto, isto é, a constante de tempo age como restrição de capacidade, o que permite que apenas alguns clientes sejam atendidos por rota. Por outro lado, C2, R2, e RC2 têm um longo horizonte de escalonamento, isto é, veículos de capacidade maior, permitindo que vários clientes sejam atendidos pelo mesmo veículo (Solomon, 1987).

3.2.3. Complexidade do Problema

Encontrar uma solução ótima para o PRVJT resulta em resolver pelo menos dois problemas NP-difíceis, entre os quais temos: o Problema do Caixeiro Viajante (PCV) e o Problema da Mochila. Com isso, podemos concluir que encontrar uma solução para o PRVJT também é um problema NP-difícil. Encontrar uma solução para o PRVJT com um conjunto limitado de veículos é um problema NP-difícil no sentido forte (Kohl, 1995). Já para um número ilimitado de veículos, temos uma solução inicial trivial, que consiste em atender cada cliente com um veículo. Modificar o problema visando minimizar múltiplos objetivos o torna ainda mais difícil. Sua complexidade de resolução aumenta significativamente devido à necessidade de se avaliar múltiplas combinações de objetivos.

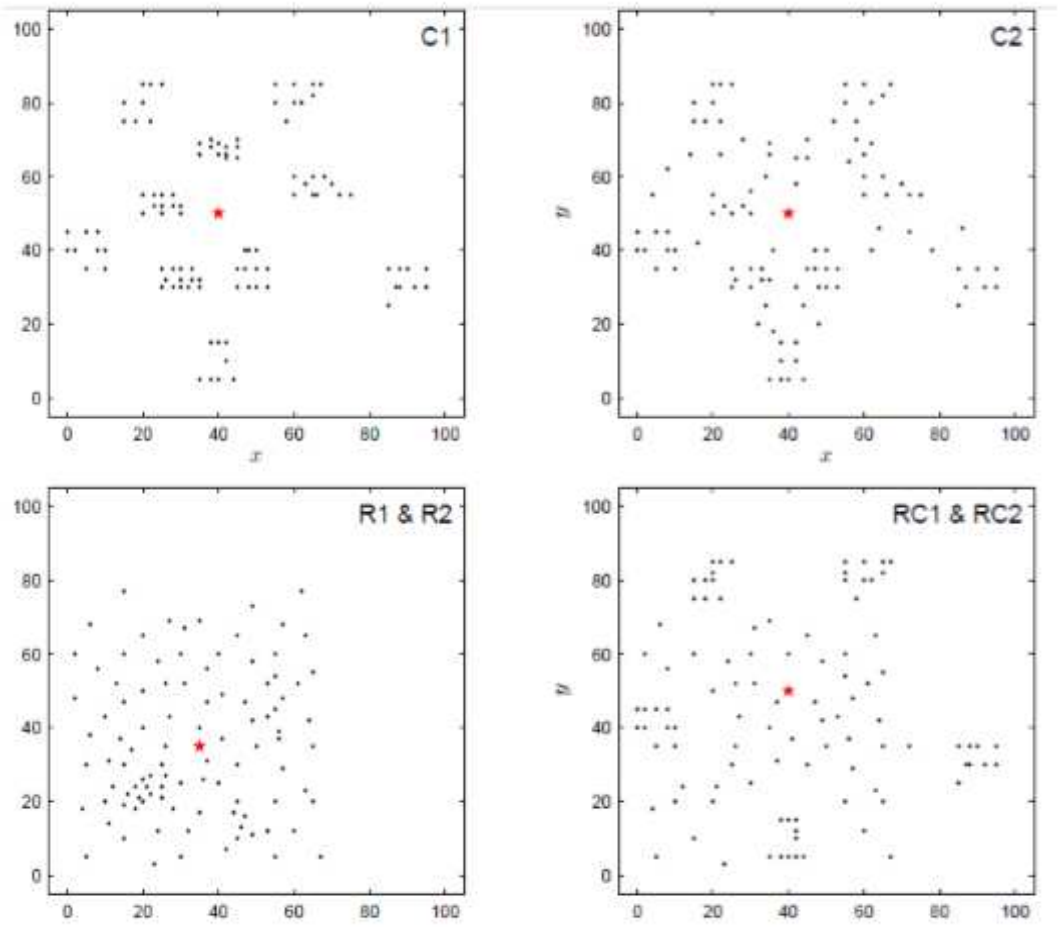

Figura 6 - Distribuição geográfica dos clientes em cada categoria dos problemas de teste de Solomon. A estrela representa o depósito.

3.2.4. PRVJT Multi-objetivo

O PRVJT busca obter rotas de atendimento aos clientes a fim de minimizar custos da prestação de serviço em geral. Para isso, é necessário definir algumas funções-objetivo para demonstrar quais tipos de reduções de custo serão feitas. Existem várias funções objetivo a serem abordadas neste problema. Algumas delas serão especificadas a seguir:

• Distância Total (f_1), que é o somatório do comprimento de todas as rotas. Considerando as variáveis do modelo definido para o PRVJT, temos que a distância total da solução é definida na Equação 3, na qual d_{ij} é a distância entre os vértices i e j.

$$f_1 : \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} x_{ijk} d_{ij} \quad (3)$$

• O Desequilíbrio das Distâncias das Rotas ($f2$) pode ser calculado pela Equação 4. Seu valor é calculado pela diferença entre a rota mais longa e a mais curta. Minimizar essa diferença melhora o desequilíbrio das distâncias das rotas.

$$f_2 : \max_{k \in K} \left(\sum_{i \in V} \sum_{j \in V} x_{ijk} d_{ij} \right) - \min_{k \in K} \left(\sum_{i \in V} \sum_{j \in V} x_{ijk} d_{ij} \right) \quad (4)$$

• O Desequilibro das Cargas ($f3$) é calculado através da Equação 5. Seu valor é calculado pela diferença entre o veículo com maior carga e o veículo com a menor carga. Minimizar essa diferença melhora o desequilíbrio das cargas.

$$f_2 : \max_{k \in K} \left(\sum_{i \in V} \sum_{j \in V} x_{ijk} d_{ij} \right) - \min_{k \in K} \left(\sum_{i \in V} \sum_{j \in V} x_{ijk} d_{ij} \right) \quad (5)$$

• Número de Veículos Necessários (f_4) é o número de rotas presentes na solução.

Este trabalho estuda dois balanços que podem ser feitos, o balanço das distâncias percorridas e o balanço das cargas. Ainda neste trabalho, foi abordado o estudo do PRVJT biobjetivo (PRVJT-bi), tomando como função-objetivo principal a distância total. Esse objetivo foi combinado com cada uma das demais funções-objetivo, gerando, assim, dois pares biobjetivo a serem estudados. Abaixo apresentamos estes pares:

• f_1 e f_2, (Distância Total e Desequilíbrio das Distâncias das Rotas).

• f_1 e f_3, (Distância Total e Desequilíbrio das Cargas).

4. Metodologia proposta

Neste trabalho, foram propostos dois algoritmos híbridos. Primeiramente, foi proposto um algoritmo Iterated Local Search, que utiliza o método Random Variable Neighborhood Descent e a combinação de soluções para resolução do problema. A segunda abordagem propõe um Algoritmo Genético com etapa de intensificação que utiliza o método Iterated Greedy. Ambos os métodos apresentaram desempenho superior nos testes realizados.

4.1. Representação da Solução

Uma solução do PRVJT é representada por k subsequências de clientes, em que k é o número de rotas da solução. Os clientes são representados por um número de 1 a n. Cada cliente deve estar em apenas uma subsequência. Em uma subsequência, o depósito é representado por 0. Os veículos percorrem os clientes seguindo a ordem da rota e voltando ao depósito no final da rota. Considere o exemplo de uma solução na qual 10 clientes são atendidos por 3 veículos. Podemos representar esta solução como [0,10,7,9,4,0][0,5,8,6,0][0,2,1,3,0].

A Figura 7 consiste na representação gráfica da solução acima, utilizando o PRVJT como entrega de encomendas. O depósito está simbolizado como uma carta e os clientes como casas. Cada cliente possui, ao lado da sua posição, o valor de sua janela de tempo. Podemos observar que as janelas de tempo interferem na ordem dos clientes na Rota 3. Desconsiderar a janela de tempo resultaria em uma menor distância total para atender o cliente 7 antes do 10, entretanto, a janela de tempo do cliente 10 começa muito cedo, impossibilitando que ele seja atendido em uma visita posterior a do cliente 7.

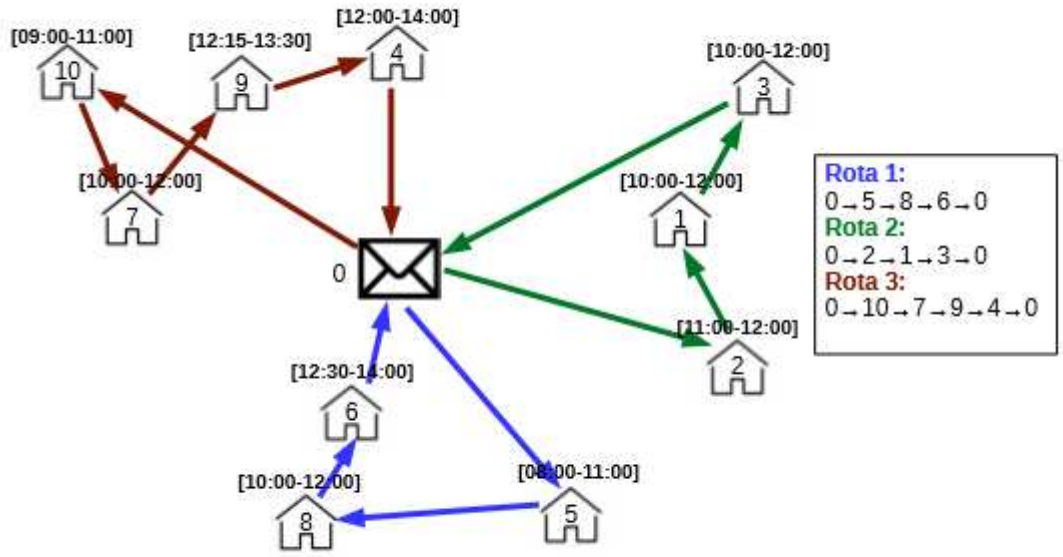

Figura 7 - Exemplo de solução para o PRVJT

4.2. Algoritmo Baseado em *Iterated Local Search*

O primeiro método proposto para resolver o PRVJT-bi foi uma heurística baseada em ILS (Iterated Local Search, Lourenço et al. (2003)), RVND (Variable Neighbourhood Descent with Random Neighbourhood Ordering, Mladenović e Hansen (1997)) e combinação de soluções não dominadas.

O ILS é uma meta-heurística simples que consiste em repetidamente aplicar uma busca local para modificar a solução corrente. Essa meta-heurística é uma poderosa técnica de otimização e demonstra bons resultados em vários problemas combinatórios mono-objetivo (Lourenço et al., 2003).

A versão do ILS utilizada nesse trabalho foi baseada nos trabalhos de Paquete et al. (2004) e Geiger (2007).

Algoritmo 1 H_MOILS(*MaxPert*, *pertInc*, *Ncomb*, *Restart*)

```
1: s = Gera_Solucao_Inicial( );
2: NDc = {s} \\ Conjunto atual de soluções não dominadas;
3: ND = ∅ \\ Conjunto Final de soluções não dominadas;
4: while (Critério_de_Parada = false) do
5:    RVND(s, NDc);
6:    Marca a solução s como explorada;
7:    Operador_de_Combinação(Ncomb, NDc);
8:    if (∃ uma solução não explorada em NDc ) then
9:       s = Seleciona uma solução não explorada aleatória de NDc;
10:      iteracoesSemMelhora = 0;
11:   else
12:      iteracoesSemMelhora++;
13:      s = Seleciona uma solução aleatória de NDc;
14:      nPert = min{(3+iteracoesSemMelhora/pertInc, MaxPert};
15:      s = Perturbacao(s, nPert);
16:      if (count > Restart) then
17:         ND = soluções não dominadas obtidas de (ND ∪ NDc);
18:         NDc = {s};
19:      end if
20:   end if
21: end while
22: ND = soluções não dominadas obtidas de (ND ∪ NDc) ;
23: Return ND;
```

1 - H_MOILS (MaxPert, pertInc, Ncomb, Restart)

A heurística proposta foi nomeada H_MOILS, e seu pseudocódigo geral é apresentado no Algoritmo 1. Na linha 1 do algoritmo, uma solução inicial s é gerada. Na linha 2, o conjunto de soluções não dominadas, NDc, é inicializado com a solução s. O conjunto de soluções não dominadas final, ND, é inicializado como vazio na linha 3. O *loop* principal do algoritmo H_MOILS está entre as linhas 4 e 21. O algoritmo executa os próximos passos até que o critério de parada seja atingido. Durante cada iteração, uma busca local baseada em RVND é aplicada usando uma solução não dominada s como solução inicial (linha 5). As novas soluções não dominadas encontradas pelo RVND são armazenadas no conjunto NDc. A solução s é marcada como explorada ao final do RVND (linha 6). Soluções marcadas não podem mais ser selecionadas como soluções iniciais do RVND nas próximas iterações. Na linha 7, o operador de combinação é aplicado, seu critério de parada é: $Ncomb$ soluções consecutivas geradas sem que o conjunto NDc seja atualizado. Na linha 8, é verificado se existe uma solução não explorada. Caso exista, uma nova solução não explorada s é selecionada de maneira aleatória (linha 9). Uma vez que todas as soluções de

NDc já foram exploradas, uma solução s é selecionada também aleatoriamente. No mínimo, serão feitas 3 operações de perturbação, e, no máximo, $M\ axP\ ert$, esse valor aumenta a cada $pertInc$ iterações, sem melhora, até atingir o máximo.

O algoritmo é reiniciado após um número de iterações (*Restart*) sem encontrar novas soluções não dominadas. Após o *Restart*, o conjunto NDc é reinicializado (linhas 16 a 19). A reinicialização consiste em três passos:

• Utilizar a solução perturbada s.

• Atribuir ao conjunto ND as soluções não dominadas de $ND \cup NDc$. Com esse passo, as não dominadas encontradas pelo algoritmo estarão salvas.

• Atribuir ao conjunto NDc o conjunto $\{s\}$, um conjunto contendo apenas a solução perturbada.

A reinicialização permite que soluções anteriormente descartadas tenham chance de entrar no conjunto NDc para serem melhoradas utilizando o RVND.

Para finalizar, o conjunto resposta ND é atualizado com as soluções do conjunto atual NDc. Assim, o algoritmo retorna ao conjunto ND contendo as melhores soluções encontradas durante a execução (linha 22).

Nas próximas subseções, cada etapa do H_MOILS é explicada em detalhes.

4.2.1. Solução Inicial

As soluções iniciais são geradas da seguinte maneira: primeiramente, uma ordem de inserção aleatória de clientes é gerada. Cada cliente é inserido seguindo essa ordem de inserção. Iniciando com uma rota vazia, os clientes sempre são inseridos no final da rota. Se a rota se tornar inválida após a inserção de um cliente i (i.e. a capacidade do veículo ou a janela de tempo for violada), a rota é finalizada.

Uma nova rota é criada e o cliente i é inserido nela. Essa nova rota se torna a rota atual. Esse processo é repetido até que todos os clientes tenham sido inseridos. O Algoritmo 2 representa o pseudocódigo para criação da solução

inicial.

Algoritmo 2 Solução inicial

```
1: O = Ordem de inserção aleatória de clientes;
2: S = Solução vazia, sem rotas;
3: r = Nova rota vazia;
4: while (O ≠ ∅) do
5:     c = Próximo cliente de O;
6:     InsereFimRota(r,c);
7:     if (VerificaViabilidadeRota(r) = false) then
8:         RemoveClienteRota(r,c);
9:         AdicionaRota(S,r);
10:        r = Nova rota vazia;
11:        InsereFimRota(r,c);
12:    end if
13:    O = O − {c};
14: end while
15:
16: AdicionaRota(S,r);
17: Return S;
```

2 - Solução Inicial

4.2.2. *Random Variable Neighbourhood Descent (RVND)*

A busca local é realizada pelo VND (Variable Neighbourhood Descent), utilizando uma ordem aleatória de vizinhanças (Random VND, Mladenović e Hansen (1997)). O pseudocódigo do RVND é apresentado no Algoritmo 3. No RVND, são utilizadas sete estruturas de vizinhanças inter-rota (vizinhanças que realizam movimentos de clientes entre rotas diferentes) e cinco vizinhanças intra-rota (vizinhanças que realizam movimentos de clientes dentro da mesma rota).

Primeiramente, o conjunto de vizinhanças (N = {N_1, ..., N_7}), contendo todas as vizinhanças inter-rota, é inicializado (linha 1). No *loop* principal (linhas 4 a 19), uma vizinhança N_i é escolhida de maneira aleatória do conjunto N (linha 5). N_i é removida de N (linha 6). As soluções vizinhas viáveis de s são geradas (linha 7), sendo s uma solução não dominada. As soluções não dominadas geradas pela vizinhança $N_i(s)$ são utilizadas para atualizar o conjunto local de soluções não dominadas da vizinhança, ND' (linha 8). Para cada solução não dominada encontrada em ND', uma busca local intra-rota é aplicada (linha 10). Nessa busca local, são realizados movimentos de clientes dentro da mesma rota. O conjunto de soluções não dominadas da vizinhança ND' é atualizado

novamente com as soluções não dominadas obtidas pela busca local intra-rota (linha 12).

Caso a solução s atual tenha sido melhorada, isto é, dominada por pelo menos uma solução $s^* \in ND'$, então a solução atual s é atualizada e N é reinicializado com todas as vizinhanças (linhas 13 a 16).

O pseudocódigo da busca local intra-Rota é apresentado no Algoritmo 4. Nessa busca local, são utilizadas cinco estruturas de vizinhança. Inicialmente, o conjunto N' de vizinhanças é inicializado e todas as cinco estruturas de vizinhança intra-rota são incluídas (linha 1). Enquanto N' não estiver vazio, o *loop* principal é executado (linhas 3 a 12). A vizinhança $Ni \in N'$ é selecionada de maneira aleatória (linha 4). Em seguida, a vizinhança Ni é removida do conjunto N' (linha 5). Depois, as soluções viáveis são geradas pela vizinhança Ni (linha 6). Se a solução atual s foi melhorada, isto é, se $\exists s^* \in ND$", tal que s^* domina s ($s^* \preceq s$), a população N' é reinicializada, todas as vizinhanças intra-rota são incluídas novamente e s passa a ser uma das soluções não dominadas (linhas 8 a 11). Ao final do método, a busca local intra-rota retorna um conjunto de soluções não dominadas ND" (linha 13).

Algoritmo 3 RVND (s, ND)

1: $N = \{N_1, N_2, ..., N_7\}$;
2: $ND = \emptyset$ \\ Conjunto local de soluções não dominadas;
3: $ND' = \emptyset$ \\ Conjunto local de soluções não dominadas da vizinhança atual;
4: **while** ($N \neq \emptyset$) **do**
5: N_i = Seleciona de maneira aleatória uma vizinhança de N;
6: $N = N - \{N_i\}$;
7: Gera os vizinhos usando $N_i(s)$ of s;
8: ND' = soluções não dominadas obtidas de ($N_i(s)$);
9: **for all** solução $s' \in ND'$ **do**
10: $ND'' =$ **IntraRota**(s');
11: **end for**
12: ND' = soluções não dominadas de ($ND' \cup ND''$);
13: **if** ($\exists s* \in ND'$ tal que $s* \preceq s$) **then**
14: $s = s*$;
15: $N = \{N_1, N_2, ..., N_7\}$;
16: **end if**
17: ND = soluções não dominadas obtidas de ($ND \cup ND'$);
18: $ND' = \emptyset$;
19: **end while**

3 - RVND (*s, ND*)

Algoritmo 4 IntraRota(s)

1: $N' = \{N_8, N_9, ..., N_{12}\}$;
2: $ND'' = \emptyset$;
3: **while** $(N' \neq \emptyset)$ **do**
4: N_i = Seleciona uma vizinhança de maneira aleatória de N';
5: $N' = N' - \{N_i\}$
6: Gera os vizinhos de s utilizando $N_i(s)$;
7: ND'' = Soluções não dominadas de $(ND'' \cup N_i(s))$;
8: **if** $(\exists s* \in ND'', s* \preceq s)$ **then**
9: $s = s*$, onde $s* \in ND''$ e $s* \preceq s$;
10: $N = \{N_8, N_9, ..., N_{12}\}$;
11: **end if**
12: **end while**
13: **Return** ND'';

4 - IntraRota (*s*)

As estruturas das vizinhanças inter-rota e intra-rota estão definidas nas próximas subseções.

Vizinhanças Inter-rota

Sete vizinhanças envolvendo movimento inter-rota propostas por Ribas et al. (2011) foram utilizadas. Esses movimentos modificam duas ou mais rotas. Somente são consideradas as soluções viáveis. Abaixo são definidas as vizinhanças:

• $N1$ - Shift(1,0): um cliente i é transferido de uma rota $r1$ para outra rota $r2$. A Figura 8 apresenta o funcionamento desse movimento.

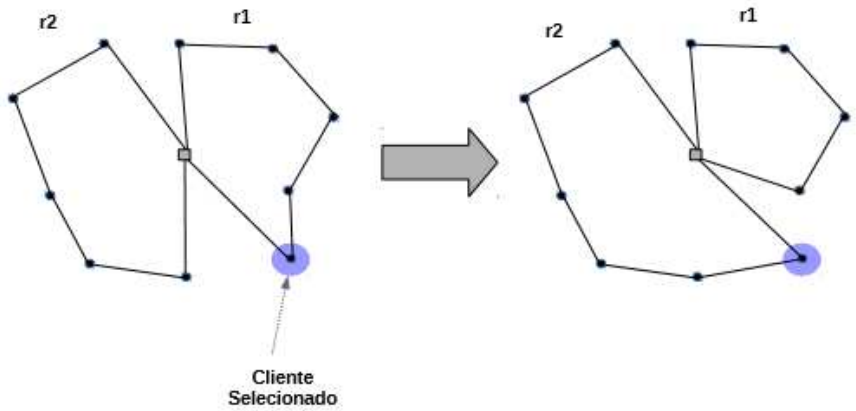

Figura 8 - Exemplo do movimento *Shift*(1,0).

• $N2$ - Swap(1,1): um cliente i da rota $r1$ é trocado com um cliente j da rota $r2$. A Figura 9 apresenta o funcionamento desse movimento.

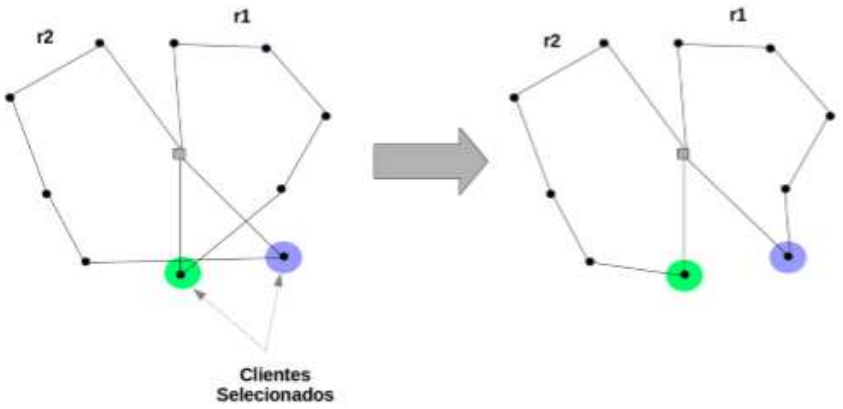

Figura 9 - Exemplo do movimento *Swap*(1,1).

• $N3$ - Shift(2,0): um par de clientes consecutivos (i, j) é transferido da rota $r1$ para a rota $r2$. A Figura 10 apresenta o funcionamento desse movimento.

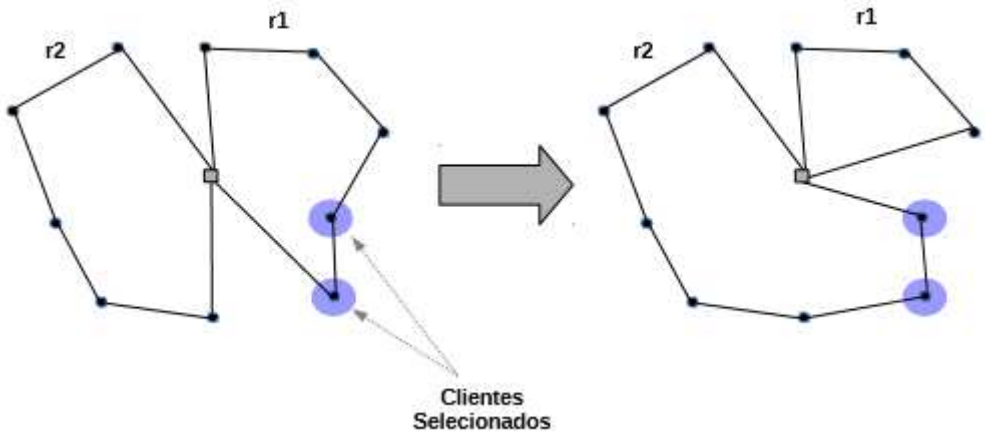

Figura 10 - Exemplo do movimento *Shift*(2,0)

• $N4$ - Swap(2,1): um par de clientes consecutivos (i, j) da rota $r1$ é trocado por um cliente k da rota $r2$. A Figura 11 apresenta o funcionamento desse movimento.

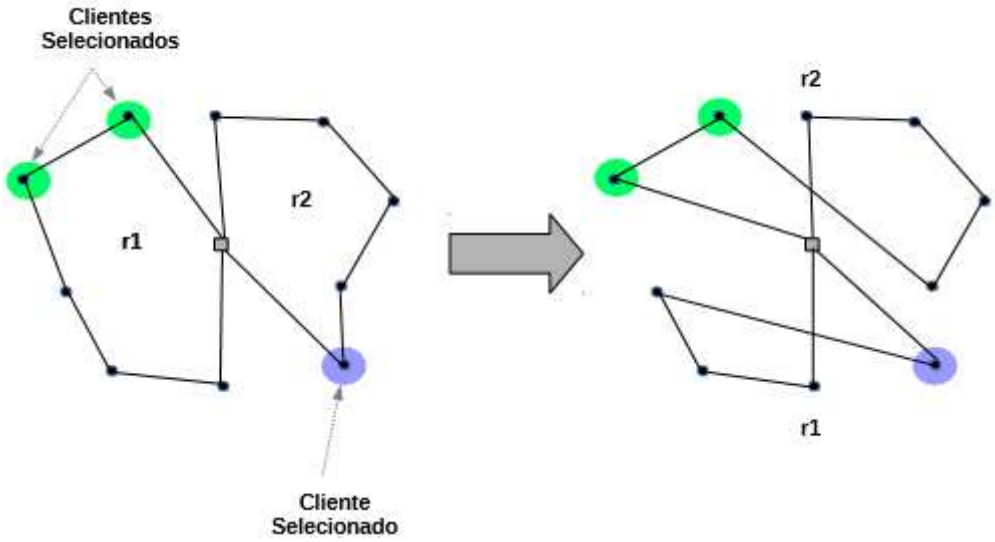

Figura 11 - Exemplo do movimento *Swap*(2,1)

• $N5$ - Swap(2,2): um par de clientes consecutivos (i, j) da rota $r1$ é trocado por um par de clientes consecutivos $(k,\ l)$ da rota $r2$. A Figura 12 apresenta o funcionamento desse movimento.

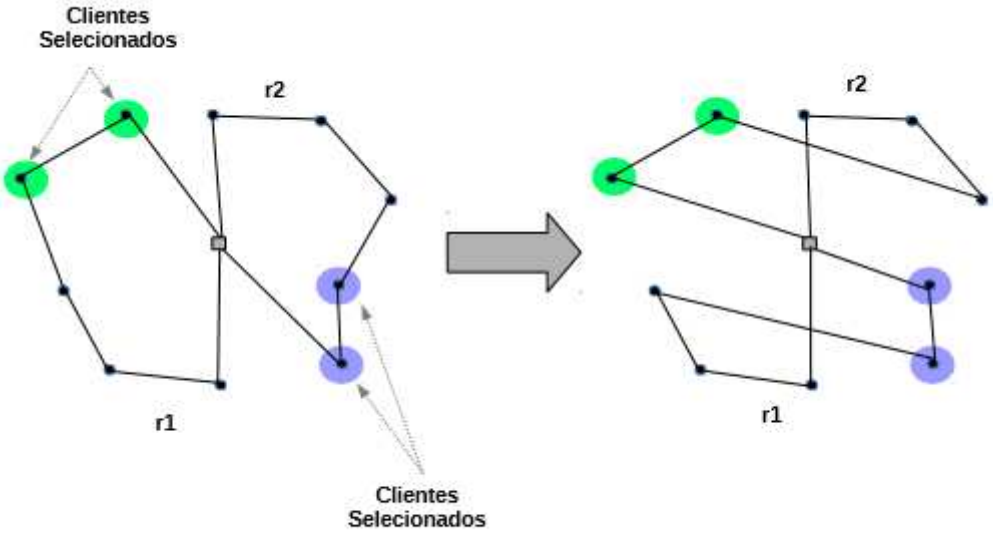

Figura 12 - Exemplo do movimento *Swap*(2,2).

• $N6$ - 2-opt inter-rota: o arco entre dois clientes consecutivos, i e j, pertencentes à rota $r1$, e o arco entre os clientes consecutivos, i' e j, da rota $r2$ são ambos removidos. Em seguida, um arco é inserido conectando i' e j, assim como outro arco conectando i e j'. A Figura 13 apresenta o funcionamento desse movimento.

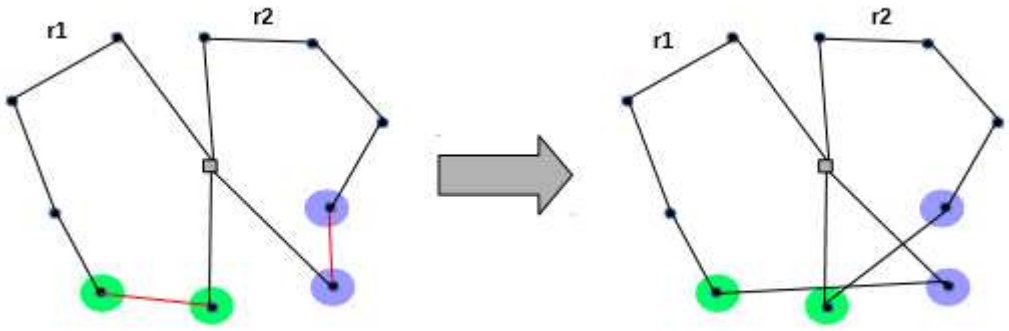

Figura 13 - Exemplo do movimento 2-opt inter rota.

• $N7$ - K-Shift: Um subconjunto k de clientes consecutivos é transferido da rota $r1$ para o final da rota $r2$ ($2 \leq K \leq |r1|$), tendo em vista que $r2$ pode ser uma rota vazia. A Figura 14 apresenta o funcionamento desse movimento.

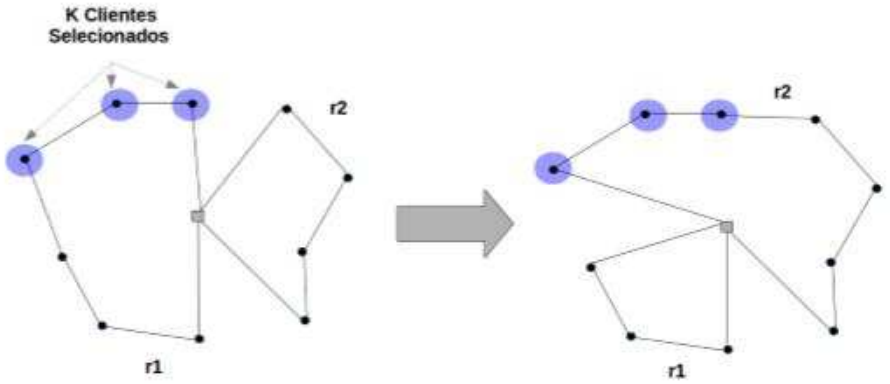

Figura 14 - Exemplo do movimento *K-Shift* inter rota.

Vizinhanças Intra-rota

Cinco vizinhanças que realizam movimentos intra-rota propostas por Ribas et al. (2011) foram utilizadas. Elas são definidas abaixo:

• $N8$ - Shift-intra-route(1,0): um cliente i da rota r é removido da sua posição original. Ele é reinserido em todas as posições possíveis da mesma rota. A Figura 15 apresenta o funcionamento desse movimento.

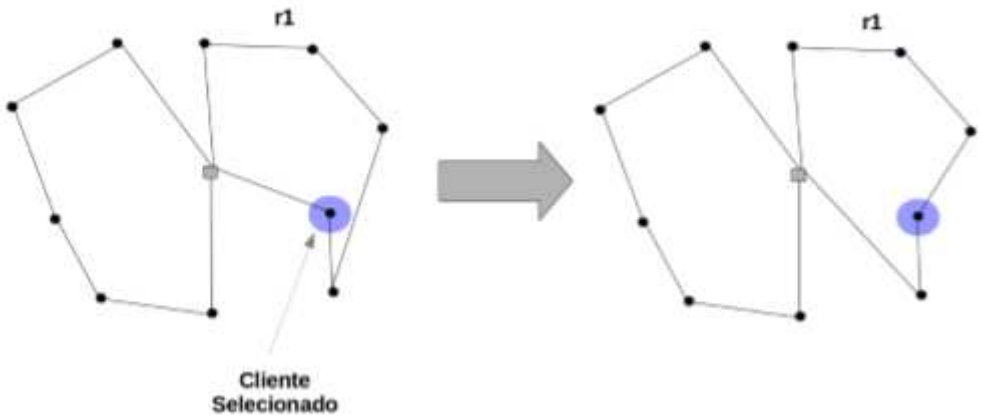

Figura 15 - Exemplo do movimento *Shift*-intra-rota(1,0).

• $N9$ - Shift-intra-route(2,0): um par de cliente consecutivos (i, j) da rota r é removido da sua posição original. Eles são reinseridos em todas as posições da mesma rota. A Figura 16 apresenta o funcionamento desse

movimento.

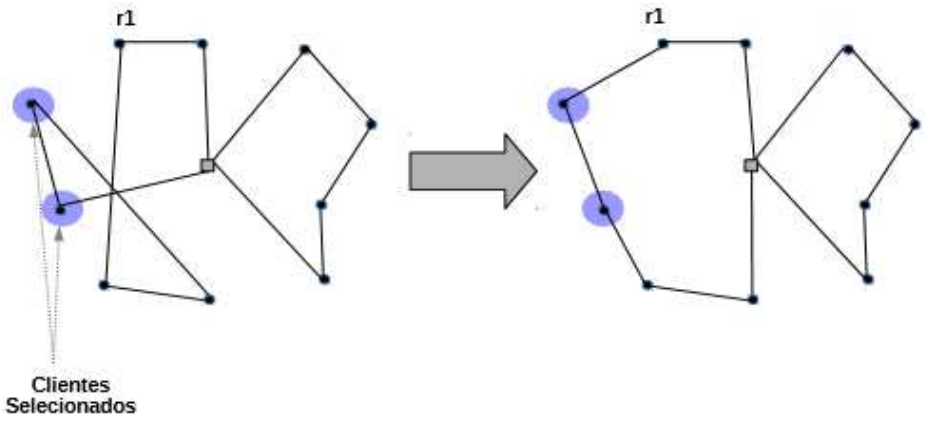

Figura 16 - Exemplo do movimento *Shift*-intra-rota(2,0).

• $N10$ - Shift-intra-route(3,0): um subconjunto de três clientes consecutivos (i, j, k) da rota r é removido de sua posição original. Eles são reinseridos em todas as posições da mesma rota. A Figura 17 apresenta o funcionamento desse movimento.

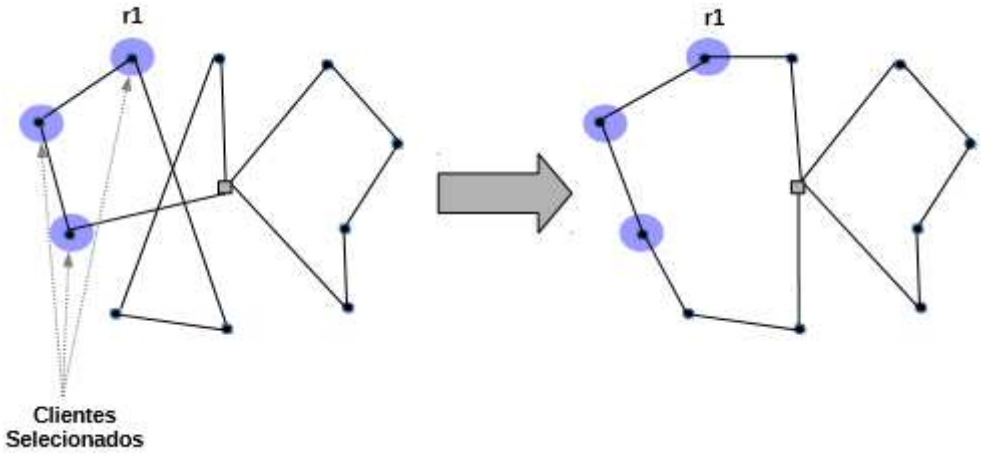

Figura 17 - Exemplo do movimento *Shift*-intra-rota(3,0).

• $N11$ - 2-Opt intra-rota: dois pares de arcos não adjacentes são removidos. Outros dois arcos são criados, de maneira que uma nova rota

é gerada. A Figura 18 apresenta o funcionamento desse movimento.

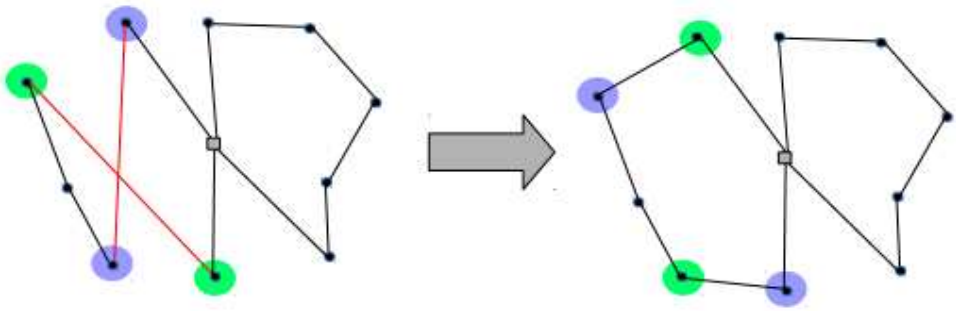

Figura 18 - Exemplo do movimento *2-opt* intra rota.

• $N12$ - Swap(1,1): dois clientes diferentes são trocados, i e j, em uma rota r. A Figura 19 apresenta o funcionamento desse movimento.

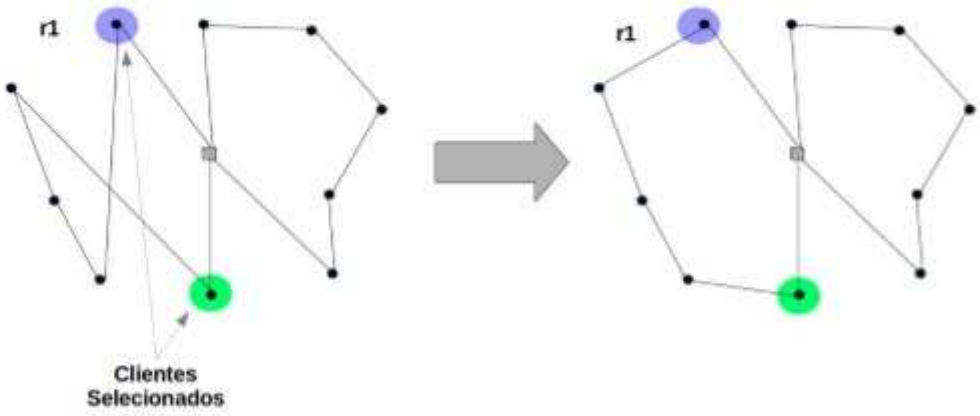

Figura 19 - Exemplo do movimento *Swap*(1,1) intra rota.

4.2.3. Operação de Combinação

No algoritmo H_MOILS, aplicamos um procedimento de intensificação. Esse procedimento combina soluções não dominadas. É aplicado um operador

de *crossover* proposto por Garcia-Najera e Bullinaria (2011) para realizar a combinação.

Nesse operador, duas soluções pais, $s1$ e $s2$, não dominadas são selecionadas de maneira aleatória do conjunto NDc. Uma nova solução filha s^* é gerada combinando $s1$ e $s2$. A combinação é descrita abaixo: um número de rotas n é selecionado de maneira aleatória da primeira solução $s1$. Essas rotas são copiadas para a solução filho s^*. Todas as rotas de $s2$ 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 $s2$. Esses clientes são inseridos em s^*, de maneira que o aumento da distância total seja o mínimo possível. Caso não seja possível fazer a inserção viável de um cliente em nenhuma rota de s^*, uma nova rota é criada contendo esse cliente. Na Figura 20, mostra-se um exemplo do funcionamento do *crossover*.

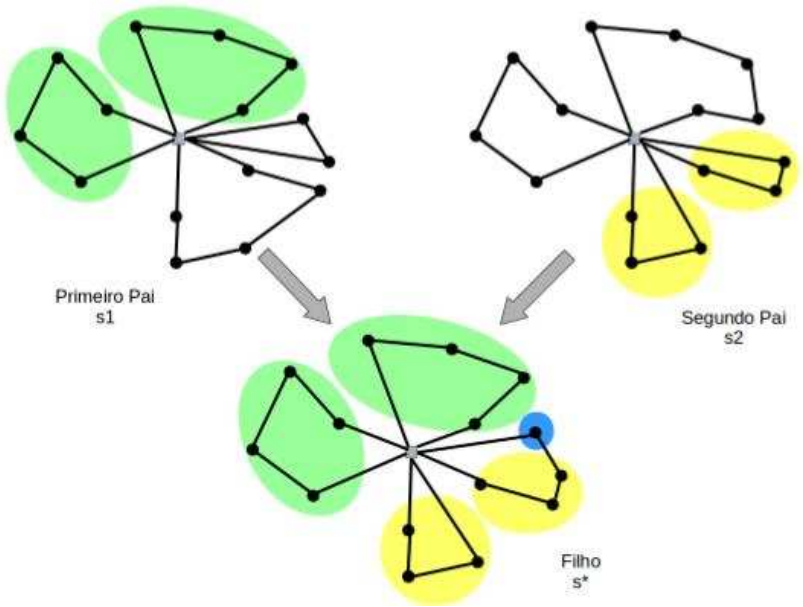

Figura 20 - Exemplo do funcionamento do operador crossover.

Esse procedimento tem um parâmetro $Ncomb$, usado para definir seu critério de parada. O procedimento de combinação é executado até que conjunto de soluções não dominadas, NDc, não seja melhorado por $Ncomb$ combinações consecutivas.

4.2.4. Mecanismos de perturbação

Dois mecanismos de perturbação foram adotados (veja o Algoritmo 5). Esses mecanismos foram baseados em movimentos aleatórios de Shift e Swap. Sempre que a perturbação é utilizada, um dos movimentos é selecionado ao acaso.

O *Shift* aleatório consiste em: selecionar de maneira aleatória uma rota $r1$ e um cliente i dessa rota. Esse cliente é reinserido em uma posição aleatória de uma rota $r2$, também selecionada de maneira aleatória. Observe que $r1$ e $r2$ podem ser iguais.

O *Swap* aleatório consiste em trocar dois clientes, i e j, pertencentes às rotas $r1$ e $r2$, respectivamente. Os clientes e as rotas são selecionados de maneira aleatória, e as rotas podem ser iguais.

Os movimentos de *Shift* e *Swap* são repetidos *nPert* vezes, isto é, *nPert* movimentos são feitos em cada perturbação. Esse nível de perturbação *nPert* é incrementado por um valor $pertIncr$ quando todas as soluções pertencentes a *NDc* já foram exploradas. Quando uma nova solução é encontrada em *NDc*, *nPert* é reiniciado para 1. O valor máximo que $nP\ ert$ pode assumir é $MaxPer$t.

Algoritmo 5 Perturbação(s, $nPert$)

```
1: n = random number (1, 2);
2: if (n == 1) then
3:     s* = RandomShift(s, nPert);
4: else
5:     s* = RandomSwap(s, nPert);
6: end if
7: Return s*;
```

5 - Perturbação (*s, nPert*)

4.3. Algoritmo Baseado em Algoritmo Genético

Além do H_MOILS, foi proposto um Algoritmo Memético (Moscato e Cotta, 2003) para resolver o PRVJT biobjetivo. Ele é baseado no Algoritmo Genético (GA), utilizando um procedimento de melhoria baseado na heurística Iterated Greedy (IG) e proposto por Ruiz e 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 não dominadas que procura soluções melhores utilizando uma estratégia gulosa.

Algoritmo 6 GAIG($Psize$, $prob_s$, $prob_c$, $prob_m$)

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

6 - GAIG (*Psize, porbs, probc, probm*)

O algoritmo proposto é denominado GAIG. As etapas principais do método são apresentadas no Algoritmo 6. Inicialmente, o algoritmo cria uma população P com Psize soluções iniciais (Linha 1). Em seguida, o conjunto de soluções não dominadas ND da população atual é determinado (Linha 2). O laço principal do algoritmo começa com o procedimento de Intensificação-IG aplicado nas soluções não dominadas do conjunto ND (Linha 4). Como resultado do IG, temos um conjunto M de soluções. O procedimento de Seleção (Linha 5) escolhe as soluções pais que serão utilizadas no *crossover*. Esse procedimento seleciona soluções não dominadas de $P \cup M$ de acordo com uma probabilidade $probs$. 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 $probm$, a cada solução filha pertencente ao conjunto Q. Em seguida, é feita uma classificação das soluções do conjunto $P \cup M \cup Q$ (Linha 8). Essa categorização primeiramente identifica as soluções não dominadas de $P \cup M \cup Q$, e as soluções dominadas são ordenadas de acordo com sua proximidade às soluções não dominadas. 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). Nessa nova

população, estarão (sobreviverão) as soluções não dominadas do conjunto R e algumas soluções que estiverem próximas às não dominadas, completando, assim, Psize soluções. A proximidade de soluções é medida através da distância euclidiana no espaço objetivo. Finalmente, o conjunto de soluções não dominadas ND é atualizado (Linha 10). Os procedimentos do GAIG são executados até que o critério de parada seja satisfeito. A Figura 21 ilustra o funcionamento do Algoritmo GAIG.

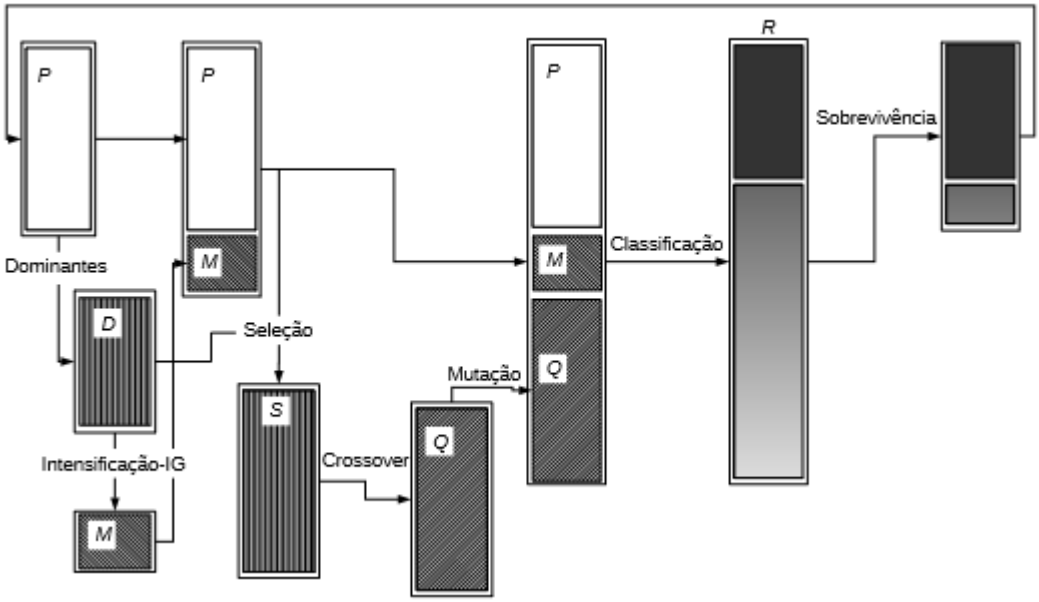

Figura 21 - Etapas do algoritmo GAIG.

4.3.1. Solução Inicial

A solução inicial é a mesma utilizada no H_MOILS. As soluções iniciais viáveis são geradas de maneira aleatória. A descrição completa dessas soluções é feita na Seção

4.3.2. Intensificação

O procedimento de intensificação é baseado 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, de modo a obter uma solução parcial. Na Reconstrução, cada elemento removido é reinserido, de

forma gulosa, na solução parcial. Dessa forma, o IG retorna a melhor solução completa obtida.

O procedimento de Intensificação-IG proposto neste trabalho é apresentado no Algoritmo 7. Neste algoritmo, o IG é aplicado de duas maneiras: IG mono-objetivo e IG biobjetivo. O IG mono-objetivo é aplicado às duas soluções não dominadas, $s1$ e $s2$ ($s1$, $s2 \in ND$), que possuem o menor valor de cada objetivo.

Na fase de Destruição do IG mono-objetivo, remove-se de maneira aleatória $d1$ clientes da solução si. Ou seja, estes clientes são retirados de rotas também selecionadas de maneira aleatória. 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 ele ocupe a melhor posição. A melhor posição é determinada de acordo com o valor da função-objetivo ($f1$ ou $f2$). Note que, na reinserção de um cliente i, é gerado um número nsi de soluções parciais (para cada posição possível do cliente, obtém-se uma solução parcial). Entre as nsi soluções parciais, seleciona-se a melhor (suponha sm). Por conseguinte, o próximo cliente removido deve ser reinserido na melhor posição de sm. O IG mono-objetivo finaliza quando o último cliente removido for reinserido na solução, pois, assim é obtida a melhor solução completa s'.

O IG biobjetivo é aplicado em uma porcentagem $porcig$ das soluções do conjunto ND. Na fase de Destruição, remove-se de maneira aleatória $d2$ clientes da solução s. Na fase de Reconstrução, também, cada cliente removido é reinserido na melhor solução possível da solução parcial. Com base nas nsi soluções parciais obtidas na reinserção de um cliente, determina-se o conjunto das soluções parciais não dominadas (NDi). O próximo cliente removido será reinserido na melhor posição de cada solução parcial do conjunto NDi. Então, um novo conjunto de soluções parciais não dominadas será obtido. O procedimento finaliza-se quando é obtido um conjunto NDi de soluções não dominadas completas, ou seja, com todos os clientes já reinseridos. No final do procedimento de Intensificação-IG, tem-se o conjunto M das soluções não

dominadas obtidas no IG mono- objetivo e biobjetivo.

Algoritmo 7 Intensificação-IG(ND)

1: $M = \emptyset$;
2: **for all** Função objetivo f_i, $i \in 1, ..., m$ **do**
3: s_i = EncontraSoluçãoComMenorObjetivo(f_i, ND);
4: s' = IG-mono-objetivo(s_i, f_i);
5: M = Não_Dominados($M \cup \{s'\}$);
6: **end for**
7: **for all** solução dominante s de $ND * porc_{ig}$ **do**
8: ND_i = IG-bi-objetivo(s);
9: M = Não_Dominandos($M \cup ND_i$);
10: **end for**
11: retorna M;

7 - Intensificação-IG (*ND*)

4.3.3. Seleção

O procedimento de Seleção consiste em selecionar um conjunto S de 2*P $size$ soluções pais para serem utilizadas pelo operador de *crossover*. Esse procedimento tem como entrada os conjuntos ND (soluções não dominadas), $P \cup M$, e uma probabilidade de seleção $probs$. 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 $r \leq probs$, seleciona-se, de maneira aleatória, uma solução não dominada do conjunto ND. Caso contrário, seleciona-se de maneira aleatória uma solução do conjunto $P \cup M$, ou seja, a probabilidade para escolher uma solução dominante é $probs$, e a probabilidade para escolher uma solução de $P \cup M$ é $1 - probs$.

4.3.4. Crossover

O operador *crossover* utilizado neste trabalho é uma versão melhorada do *crossover* proposto por Tan et al. (2006a). Essa versão melhorada foi proposta por Chiang e Hsu (2014). Inicialmente, o *crossover* copia as duas soluções pais para as duas soluções filhas. Em seguida, a melhor rota de cada filho é escolhida. A melhor rota é definida como a rota que possui menor valor da Equação 4.1. A melhor rota de uma solução filha é inserida na outra solução filha, removendo os clientes repetidos das rotas antigas. Como a rota adicionada é viável, a solução não deixará de ser viável nessa operação.

$$\frac{Distância\_total}{N^o\_de\_clientes}$$

Esse *crossover* possui uma fraqueza: geralmente, esse tipo de operação aumenta o número de rotas que originalmente os pais possuíam. Para melhorar esse aspecto, Chiang e Hsu (2014) introduziram uma tentativa de redução de rota após o *crossover*. A pior rota, rota com maior valor obtido na Equação 4.1, e as rotas muito curtas (rotas com um ou dois clientes) são removidas. Todos os clientes removidos são inseridos novamente na solução em uma rota onde a distância total obtida após a inserção seja a menor possível. Esse valor de aumento é calculado através da formula (dik + dkj + dij), em que e k é o cliente a ser inserido. A Figura 22 apresenta um exemplo do funcionamento do *crossover*. As rotas grifadas em azul e verde são as melhores rotas do filho 1 e 2, respectivamente. Na Figura 23, temos o exemplo de como é feita a redução de rotas aplicada após o *crossover*. A pior rota e as rotas com um ou dois clientes são removidas e seus clientes, reinseridos na solução.

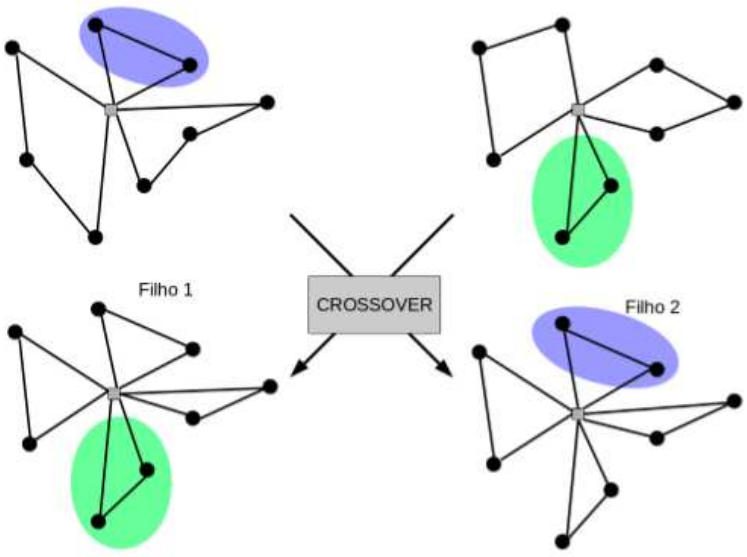

Figura 22 - Exemplo do *crossover* da inserção da melhor rota.

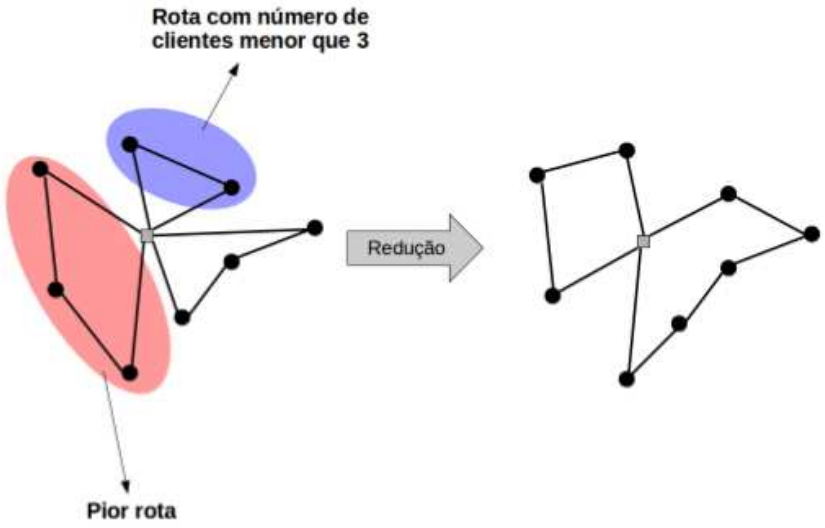

Figura 23 - Exemplo da redução de rota aplicada após o *crossover*.

4.3.5. Mutação

No GAIG, é utilizada a mutação proposta por Garcia-Najera e Bullinaria (2011). Aplica-se uma mutação a cada filho gerado pelo *crossover* com uma probabilidade de $probm$. São aplicados três operadores de mutação. Essas mutações utilizam as seguintes operações básicas:

• *SelecionaRota*: seleciona uma rota de maneira aleatória com maior probabilidade para rotas longas com poucos clientes.

• *SelecionaClientes*: seleciona de maneira aleatória um cliente de uma rota específica. A seleção aleatória tem maior probabilidade de escolher clientes com maior valor da média da soma de seus arcos de entrada e saída. Observe que o primeiro cliente e o último cliente de uma rota devem considerar apenas o arco de saída e entrada, respectivamente.

• *InsereClientes*: essa operação tenta inserir um conjunto de clientes, um de cada vez, em uma rota específica onde seja obtida a menor distância total. 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 pelo uso da

operação $SelecionaClientes$. Esses clientes são removidos da rota, incluindo aqueles localizados entre os dois. Por fim, a operação $InsereClientes$ tenta inserir os clientes removidos em todas as possíveis rotas existentes. Essa operação está representada na Figura 24.

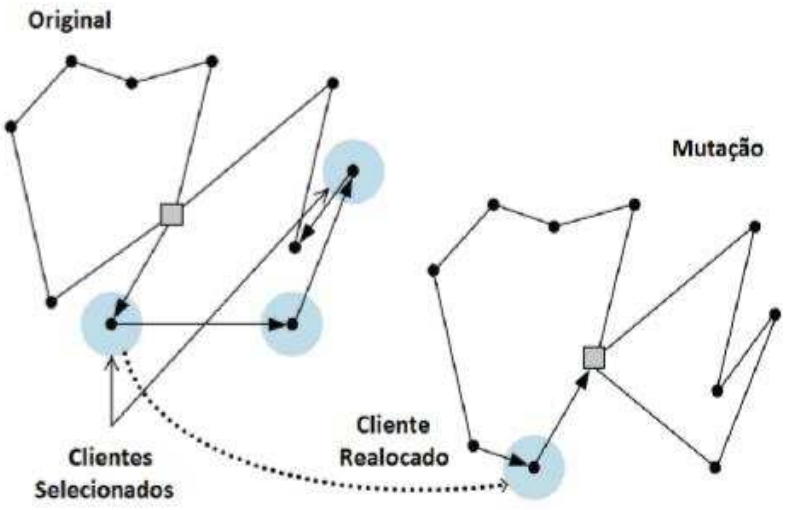

Figura 24 - Exemplo da operação de *Realocação* utilizada na mutação

• *Troca*: esse 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 e a operação $InsereClientes$ tenta inserir essa sequência de clientes na outra rota. Se um ou mais clientes não puderem ser inseridos, a troca é cancelada e as rotas originais são mantidas. Essa operação está representada na Figura 25.

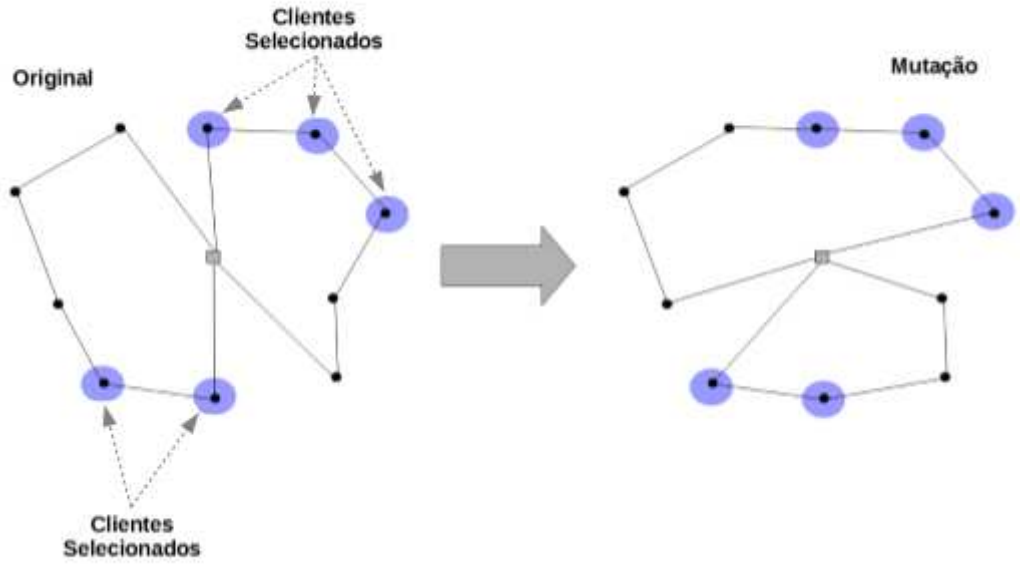

Figura 25 - Exemplo da operação de Troca utilizada na mutação

• *Reposicionamento*: utiliza a operação $SelecionaClientes$ para selecionar um cliente de uma rota. Em seguida esse cliente é inserido na mesma rota utilizando a operação $InsereClientes$. Essa operação está representada na Figura 26.

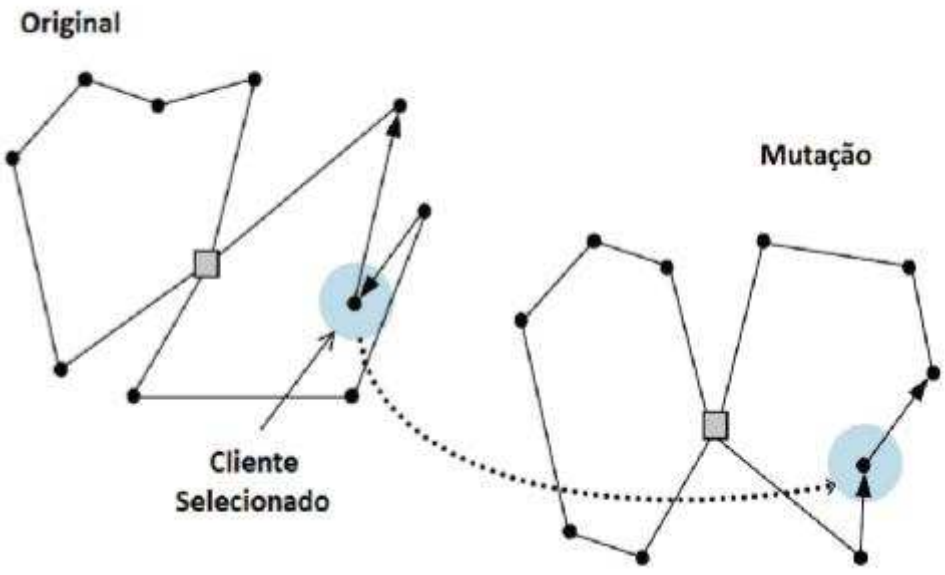

Figura 26 - Exemplo da operação de *Reposicionar* utilizada na mutação

A mutação funciona seguindo o procedimento a seguir: duas rotas são selecionadas utilizando as regras $SelecionaRota$. Se elas forem iguais, a operação de Realocação é aplicada; caso contrário a operação de Troca é aplicada. Em seguida, a operação $SelecionaRota$ escolhe uma terceira rota e aplica-se a operação *Reposicionar*.

5. Resultados computacionais

Os algoritmos propostos, GAIG e H_MOILS, foram desenvolvidos na linguagem de programação C++. Os testes foram efetuados em um computador com microprocessador Intel Core i5 2500 3.3 Ghz, com 8 Gb de memória RAM, utilizando sistema operacional Windows 8.1 PRO. A comparação dos algoritmos propostos foi feita analisando os resultados obtidos nesse computador e comparando-os com dois algoritmos da literatura. Os algoritmos da literatura utilizados na comparação foram o MMOEASA (Banos et al., 2013), um algoritmo híbrido que combina o Algoritmo Genético com o *simulated annealing*, e o NSGA II (Deb et al., 2002), que foi implementado utilizando os operadores de *crossover* e mutação utilizados por Garcia-Najera e Bullinaria (2011). Como não foram encontrados todos os resultados para comparação em seus respectivos artigos, reimplementamos ambos os métodos seguindo a descrição em cada um dos artigos.

Para prover estatísticas de maior confiança, todos os algoritmos foram executados 30 vezes em cada instância do Solomon (descrito na Seção 3.2.2), usando uma semente de número aleatório diferente em cada iteração. As aproximações do Pareto-ótimo (conjunto de soluções não dominadas obtidas) foram salvas em cada iteração para análise posterior. Para cada instância, um algoritmo foi executado por 40 segundos, ou seja, todos os algoritmos foram executados com a mesma condição de parada.

Dois pares de objetivos foram abordados no PRVJT-bi. No primeiro par de objetivos, foram utilizadas as funções-objetivo Distância Total ($f1$) e Desequilíbrio das Distâncias das rotas ($f2$). No segundo par de objetivos, foram utilizadas as funções- objetivo, Distância Total ($f1$) e Desequilíbrio das Cargas ($f3$). A seguir, apresentamos a definição dos parâmetros dos algoritmos implementados e a análise dos resultados para os dois pares de objetivos.

5.1. Parâmetros dos Algoritmos

Os parâmetros utilizados para o algoritmo NSGA II foram: tamanho da população = 100, seleção de torneio = 2, probabilidade de *crossover* = 100% e probabilidade de mutação = 10%, de acordo com os parâmetros do artigo

(Garcia-Najera e Bullinaria, 2011). No MMOEASA os parâmetros foram utilizados também seguindo os preceitos definidos no artigo de Banos et al. (2013), sendo eles: tamanho da população = 40, probabilidade de *crossover* = 25% e probabilidade de mutação = 25%.

Os parâmetros dos algoritmos propostos foram calibrados executando cada configuração 5 vezes para cada problema. Com a conclusão desse procedimento, aplicamos um teste estatístico de Análise de Variância (ANOVA, Montgomery (2008)) para validar os resultados. A calibração do algoritmo H_MOILS foi dividida: cada parte independente do algoritmo foi calibrada separadamente. A calibração foi decomposta em: perturbação, reinício e combinação. As comparações de calibração do H_MOILS foram feitas considerando apenas a métrica do *hypervolume*. Na perturbação, foram testados valores para $pertIncr$ e $MaxPert$. O parâmetro $pertIncr$ controla o aumento do número de operações da perturbação. A cada $pertIncr$ perturbações consecutivas, o número de operações de perturbação é incrementado em uma unidade. Já o $MaxPert$ define o número máximo de operações de perturbação que podem ser aplicadas. Foram testadas as combinações dos valores abaixo:

- $pertIncr$ = {0, 1, 5, 10, 20}
- $MaxPert$ = {3, 6, 10, 15}

Na Figura 27, são apresentados os resultados da calibração da perturbação. Pode-se observar que muitas configurações não apresentaram diferença significativa (os intervalos se sobrepuseram), então foi escolhida a que apresentou menor média, que é a configuração inc_1_$maxpert$_10, na qual $pertIncr$ = 1 e $MaxPert$ = 10. No reinício, testamos valores para o parâmetro $Restart$. Esse parâmetro define após quantas perturbações consecutivas o algoritmo reinicia o conjunto NDc. As soluções já encontradas são armazenadas e o algoritmo se reinicia com NDc contendo apenas uma solução perturbada. Foram testados os valores abaixo:

- $Restart$ = {25, 50, 75, 100, 125, 150, 175}

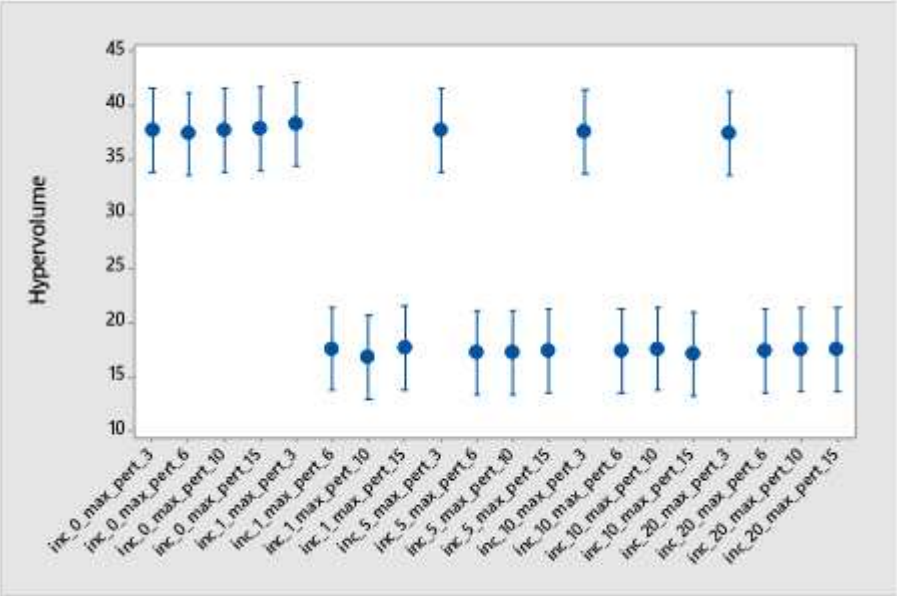

Figura 27 - Calibração da perturbação do algoritmo H_MOILS utilizando a métrica de *hypervolume*.

Nos resultados da calibração do reinício na Figura 28, novamente, as configurações ficaram muito próximas, sem demonstrar diferenças significativas, portanto foi escolhido o valor de $Restart$ = 50. Essa escolha foi feita visando dar ao algoritmo maior chance de melhorar as soluções perturbadas.

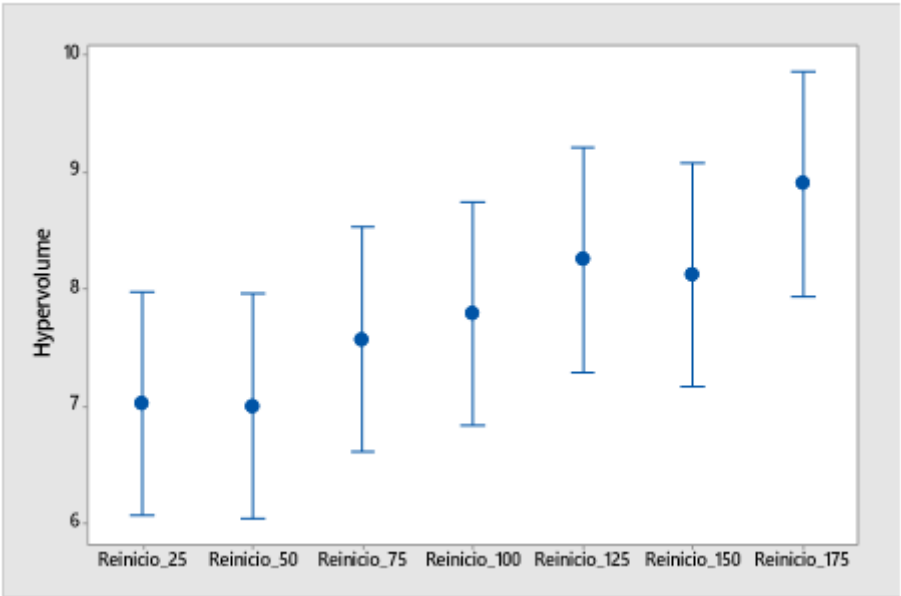

Figura 28 - Calibração do reinício do algoritmo H_MOILS utilizando a métrica de hypervolume.

Na combinação, calibramos o parâmetro $Ncomb$. O parâmetro $Ncomb$ indica a condição de parada para a combinação de soluções que ocorre após $Ncomb$ iterações, sem a adição de novas soluções a NDc. Foram testados os valores abaixo para $Ncomb$:

• $NComb$ = {0, 100, 200, 300, 400, 500, 600}

A Figura 29 apresenta os testes feitos sobre critério de parada da combinação, o número máximo de combinações sem melhora ($Ncomb$). Alguns valores foram testados e boa parte deles não apresentou diferença significativa. Assim, foi escolhido $Ncomb$ = 200, objetivando que mais soluções fossem geradas pela combinação.

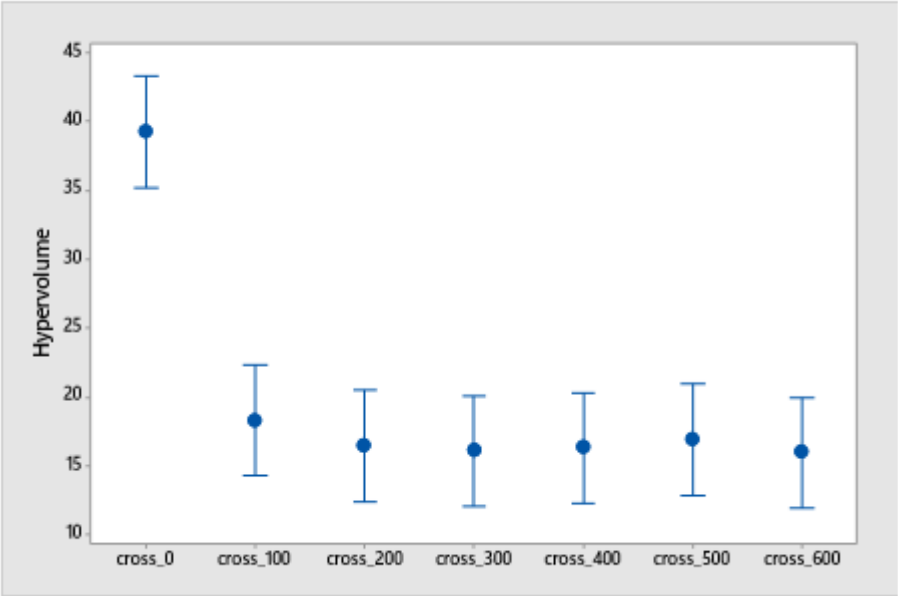

Figura 29 - Calibração da combinação do algoritmo H_MOILS utilizando a métrica de *hypervolume*.

Na calibração do GAIG, utilizamos a combinação de cada valor para cada parâmetro a ser calibrado do algoritmo. Cada uma dessas combinações de parâmetros gerou uma configuração. No total, foram testadas 486 configurações diferentes. Ambas as métricas foram utilizadas na comparação. As variáveis a serem calibradas e seus valores estão descritos abaixo:

• $P\ size$ = {100, 150, 200}

• $probc$(Probabilidade de *Crossover*) = {0.8, 1}

• $probm$(Probabilidade de Mutação) = {0.1, 0.2, 0.3}

• $probs$(Probabilidade de Seleção) = {0.3, 0.5, 0.8}

• $porcig$(Percentagem de Iterated Greedy) = {0.1, 0.2, 0.5}

• $d1$(Número de removidos no Iterated Greedy Mono-objetivo) = {5, 10, 15}

• $d1$(Número de removidos no Iterated Greedy Multiobjetivo) = {3, 5, 8}

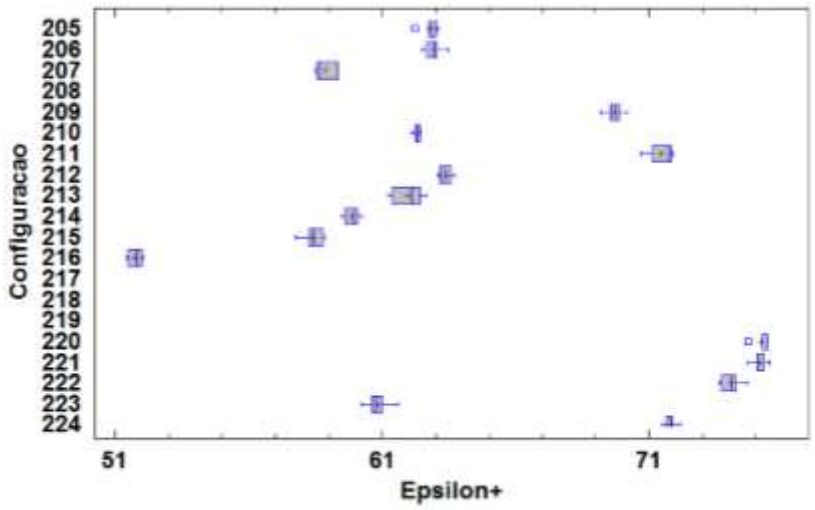

Figura 30 - Calibração do algoritmo GAIG utilizando a métrica de *Epsilon+*.

Tentou-se aplicar o ANOVA para a calibração do GAIG, porém o teste de normalidade falhou. Esse fato nos forçou a utilizar um teste não paramétrico.

Foi utilizado o teste de Kruskal-Wallis (Theodorsson-Norheim, 1986). Nas Figuras 30 e 31, podemos observar o *box-and-whisker plot* de parte dos resultados das configurações de ambas as métricas. Avaliando os resultados dos testes, foi constatado que a configuração 216 obteve menores médias e está em um grupo homogêneo diferente das demais configurações, sendo significativamente diferente das demais. Os seus parâmetros da configuração 216 são: $P\ size$, tamanho da população = 100, $probc$ = 0,8, $probm$ = 0,3, $probs$ = 0,3, $porcig$ = 0,5, $d1$ = 15 e $d2$ = 8. Essa calibração do GAIG com tantos

parâmetros só foi possível devido à utilização do *cluster* da Universidade Federal de Viçosa. Devido ao grande volume de configurações, repetições e pares de objetivos, o tempo para realizar a calibração em um computador normal seria de 90 dias, período que foi reduzido para apenas 2 dias utilizando o *cluster*.

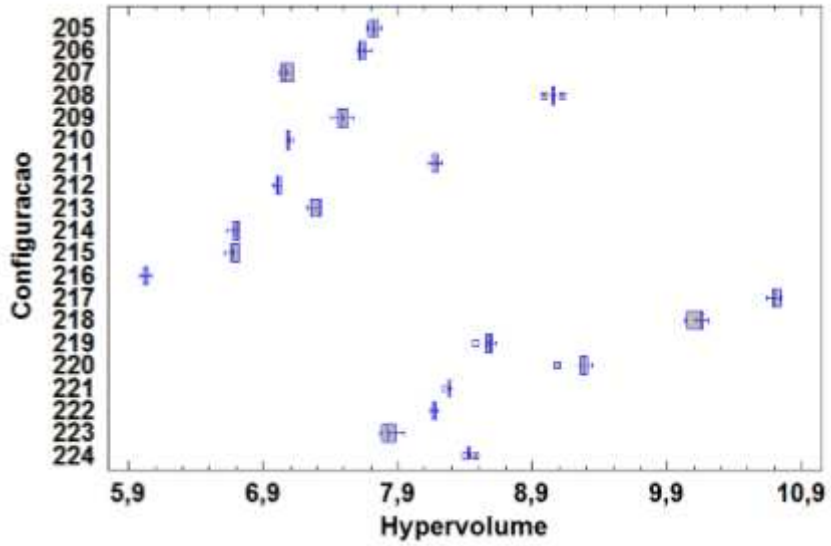

Figura 31 - Calibração do algoritmo GAIG utilizando a métrica de *Hypervolume*+.

5.2. Experimentos com Distância Total ($f1$) e Desequilíbrio das Distâncias das rotas ($f2$)

Nessa seção, são apresentados os resultados obtidos pelos algoritmos pela minimização do par de objetivos Distância Total ($f1$) e Desequilíbrio das Distâncias das rotas ($f2$). As comparações são feitas através das métricas de performance apresentadas anteriormente, *hypervolume* e Epsilon+.

Nas Tabelas 1, 2 e 3, são apresentadas as médias das métricas obtidas por cada algoritmo em cada instância. Podemos observar que os algoritmos propostos conseguiram melhores resultados que os algoritmos da literatura na maior parte dos problemas. Nos problemas da classe C1, problemas nos quais os objetivos não são conflitantes e possuem um curto horizonte de roteamento, os algoritmos propostos H_MOILS e GAIG obtiveram performances bem similares, considerando o *hypervolume*. Nos problemas das demais categorias, C2, R1, R2, RC1, RC2, o algoritmo GAIG obteve melhores médias em quase

todos os problemas.

Tabela 1 - Estão apresentadas a média das métricas de performance, considerando as 30 iterações para os objetivos Distância total ($f1$) e Desequilíbrio das distâncias das rotas ($f2$). Os problemas apresentados são da classe C. O valor em negrito indica o melhor valor para o problema.

	Hypervolume				Epsilon+			
Problemas	MMOEASA	NSGAII	GAIG	H_MOILS	MMOEASA	NSGAII	GAIG	H_MOILS
C101	30,15	19,57	32,36	**11,6**	63,63	19,19	28,19	**16,39**
C102	25,04	**10,92**	11,27	8,419	128	26,93	**14,64**	53,79
C103	25,96	14,25	**9,629**	11,92	174,4	150,1	**19,37**	158,2
C104	17,29	13,29	**6,909**	10,6	280,6	197,2	**30,69**	236,2
C105	25,89	12,01	14,85	**5,67**	192,1	18,00	**16,88**	24,77
C106	32,35	16,18	26,22	**9,517**	288,3	27,85	24,12	**21,91**
C107	23,97	9,672	15,04	**5,01**	410,3	19,86	**15,69**	52,75
C108	25,92	13,82	14,17	**7,311**	450,6	69,1	**15,98**	118,7
C109	35,28	13,92	8,827	**8,122**	669,4	183,9	**12,92**	164,7
C201	9,951	1,717	2,79	**0,775**	173,4	8,672	**6,142**	9,675
C202	20,81	1,558	**0,7652**	2,548	327,3	22,15	**9,669**	44,57
C203	29,35	5,814	**0,5768**	7,079	450,3	83,17	**10,14**	116,1
C204	39,95	10,08	**0,8256**	11,26	826,5	198,8	**19,07**	237,7
C205	31,4	2,748	**0,2195**	3,062	575	43,23	**0,9013**	59,51
C206	37,77	2,776	**0,4548**	3,31	738,6	42,94	**0,9481**	77,28
C207	32,84	3,292	**0,5503**	5,194	625,3	48,85	**2,935**	111,4
C208	45,78	3,165	**0,2315**	7,111	948,9	57,74	**1,633**	151,8

A Figura 32 mostra o gráfico de médias em todas as instâncias resultantes do teste Tukey da Diferença Honestamente Significativa (HSD), cujo nível de confiança é de 95% para os algoritmos testados, considerando a métrica *hypervolume*. Nessa figura, o intervalo HSD de Turkey do algoritmo GAIG não se sobrepõe a nenhum outro intervalo, o que significa que a diferença desse algoritmo para os demais é significativa. Como verificamos nas Tabelas 1, 2, 3 e agora comprovamos utilizando testes estatísticos, o GAIG tem desempenho superior em relação aos demais algoritmos que utilizam a métrica H*ypervolume* para os objetivos *f1* e *f2*.

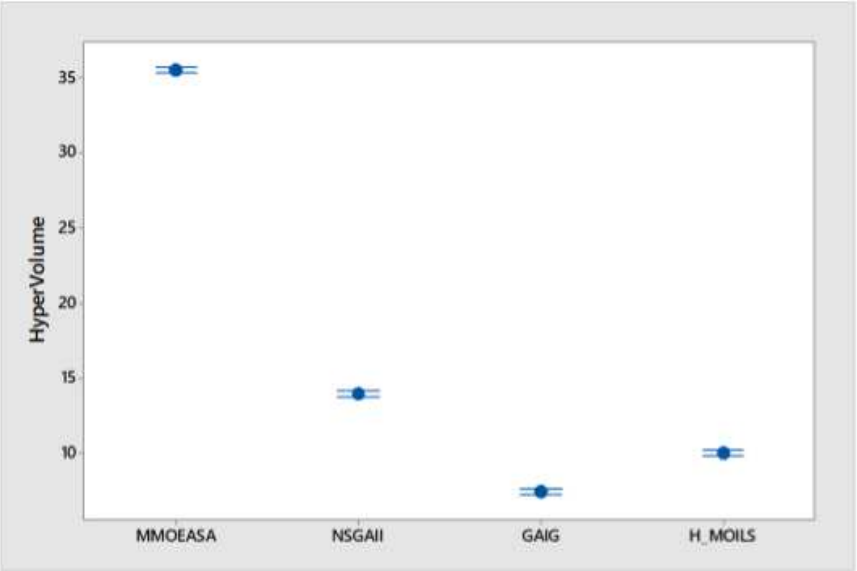

Figura 32 - Médias e Intervalos HSD de Turkey com nível de confiança de 95% *Hypervolume*.

Tabela 2 - Estão apresentadas a média das métricas de performance, considerando as 30 iterações para os objetivos Distância total (f1) e Desequilíbrio das distâncias das rotas (f2). Os problemas apresentados são da classe R. O valor em negrito indica o melhor valor para o problema.

	Hypervolume				Epsilon+			
Problemas	MMOEASA	NSGAII	GAIG	H_MOILS	MMOEASA	NSGAII	GAIG	H_MOILS
R101	28,19	14,48	13,29	**11,29**	134,4	85,63	**37,22**	71,53
R102	28,14	14,24	**11,57**	13,67	127,6	87,18	**43,34**	67,71
R103	30,94	17,51	**9,908**	10,12	145,3	93,08	**27,21**	57,69
R104	28,4	11,7	**5,28**	8,577	241,8	83,05	**27,58**	68,38
R105	30,64	19,55	**13,32**	14,02	152,7	99,66	**38,53**	72,93
R106	28,67	18,2	**8,774**	14,44	146,9	97,26	**25,00**	82,97
R107	27,12	13,17	**7,639**	9,688	172,5	72,57	**30,64**	56,36
R108	24,89	10,46	**5,799**	6,373	297,6	85,03	**29,29**	67,09
R109	38,05	18,55	9,986	**9,587**	249,4	116,9	**38,65**	66,18
R110	30,23	16,14	**6,243**	8,833	270,2	124,1	**25,85**	75,7
R111	29,96	14,1	**5,617**	9,625	266,8	101	**23,79**	79,53
R112	40,38	14,41	**5,732**	8,167	464,9	119,2	**27,97**	86,91
R201	42,69	20,61	**6,383**	15,23	457,4	211,5	**59,91**	161,3
R202	38,24	18,45	**4,783**	17,65	426,4	203,7	**53,83**	198,1
R203	38,16	16,32	**8,42**	12,58	497,9	215,6	**113,8**	169,7
R204	50,93	13,72	**4,233**	10,32	600,00	159,5	**53,11**	122,6
R205	50,28	15,25	**5,575**	9,32	744,7	219,2	**81,80**	137,1
R206	56,48	15,83	**4,97**	13,33	769,5	212,6	**69,05**	180,6
R207	46,38	13,72	**4,175**	11,67	721,4	211,7	**66,80**	182,4
R208	60,92	14,93	**3,235**	9,685	749,6	180,5	**43,76**	117,5
R209	52,81	16,83	**5,974**	10,89	781,3	242	**88,58**	160,7
R210	45,99	16,2	**4,844**	10,07	601,3	206,6	**63,69**	131,9
R211	57,3	17,01	**3,815**	10,12	922,4	269,3	**62,02**	164,5

Tabela 3 - Estão apresentadas a média das métricas de performance, considerando as 30 iterações para os objetivos Distância total (f1) e Desequilíbrio das distâncias das rotas (f2). Os problemas apresentados são da classe RC. O valor em negrito indica o melhor valor para o problema

	Hypervolume				Epsilon+			
Problemas	MMOEASA	NSGAII	GAIG	H_MOILS	MMOEASA	NSGAII	GAIG	H_MOILS
RC101	28,05	11,42	**8,893**	11,25	191,2	75,85	**46,18**	79,26
RC102	29,01	15,55	**8,556**	11,86	172,5	87,1	**28,56**	74,01
RC103	29,25	18,34	**10,81**	12,36	192,7	105,8	**54,81**	87,26
RC104	26,26	12,53	**7,891**	8,608	259,5	81,98	**25,31**	77
RC105	26,71	13,99	**9,337**	10,99	190,2	87,89	**37,02**	74,36
RC106	24,16	14,38	**7,555**	8,731	222,4	110,9	**31,2**	80,72
RC107	31,96	16,7	**8,714**	11,8	314,1	143,7	**58,18**	122,9
RC108	28,85	11,91	**7,118**	8,123	385,3	106,4	**42,84**	100,2
RC201	34,86	18,07	**7,118**	13,6	535,4	272,5	**104,5**	211,4
RC202	34,69	19,23	**5,796**	14,01	557	305,4	**96,38**	224,8
RC203	42,91	18,63	**5,014**	13,16	654,6	278,1	**76,65**	198,1
RC204	39,14	12,06	**3,742**	7,872	702,8	213	**69,92**	142,8
RC205	34,09	20,51	**6,259**	16,26	540	323,3	**95,84**	261,5
RC206	49,67	16,53	**3,266**	11,58	877,1	281,9	**58,55**	202,9
RC207	41,51	14,95	**2,099**	9,834	710,6	251,7	**38,72**	173,1
RC208	57,24	18,23	**3,013**	8,572	1104	343,8	**59,05**	163,7

A Figura 33 mostra o gráfico de médias resultantes do teste Tukey da Diferença Honestamente Significativa (HSD) com nível de confiança de 95% para as os algoritmos testados, considerando a métrica *epsilon+*. Novamente, o algoritmo GAIG não se sobrepõe a nenhum outro intervalo, mostrando ser significativamente diferente dos outros algoritmos. Comprovando também os resultados indicados nas Tabelas 1 e 2, observamos o GAIG com melhores resultados ao utilizar a métrica de comparação *epsilon+* nos objetivos f1 e f2. No artigo de Banos et al. (2013), são apresentadas as aproximações de fronteiras de Pareto obtidas pelos algoritmos MMOEASA, SPEA2 e NSGAII para as duas instâncias R108 e RC203. Nas Figuras 34 e 35, comparamos as fronteiras obtidas pelos algoritmos propostos com as fronteiras apresentadas por Banos et al. (2013). Os objetivos abordados nas fronteiras apresentadas nessas figuras são a Distância Total (f1) e Desequilíbrio das Distâncias das rotas (f2). Podemos observar que as soluções encontradas pelo GAIG e pelo H_MOILS foram superiores às publicadas por Banos et al. (2013). Podemos também observar que a diferença no problema R108 é pequena entre o GAIG e o H_MOILS. Já no problema RC203, o GAIG é claramente superior no objetivo f1.

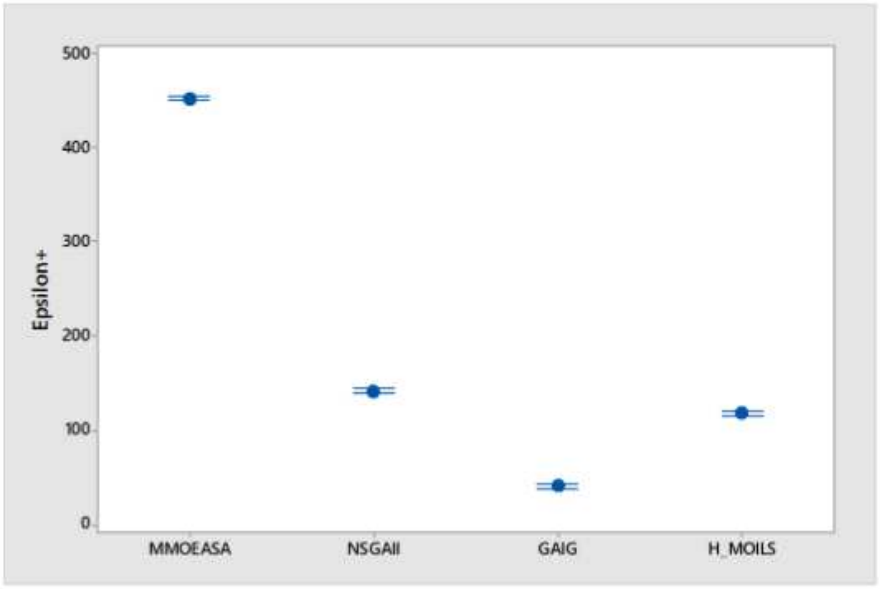

Figura 33 - Médias e Intervalos HSD de Turkey com nível de confiança de 95% *Epsilon+*.

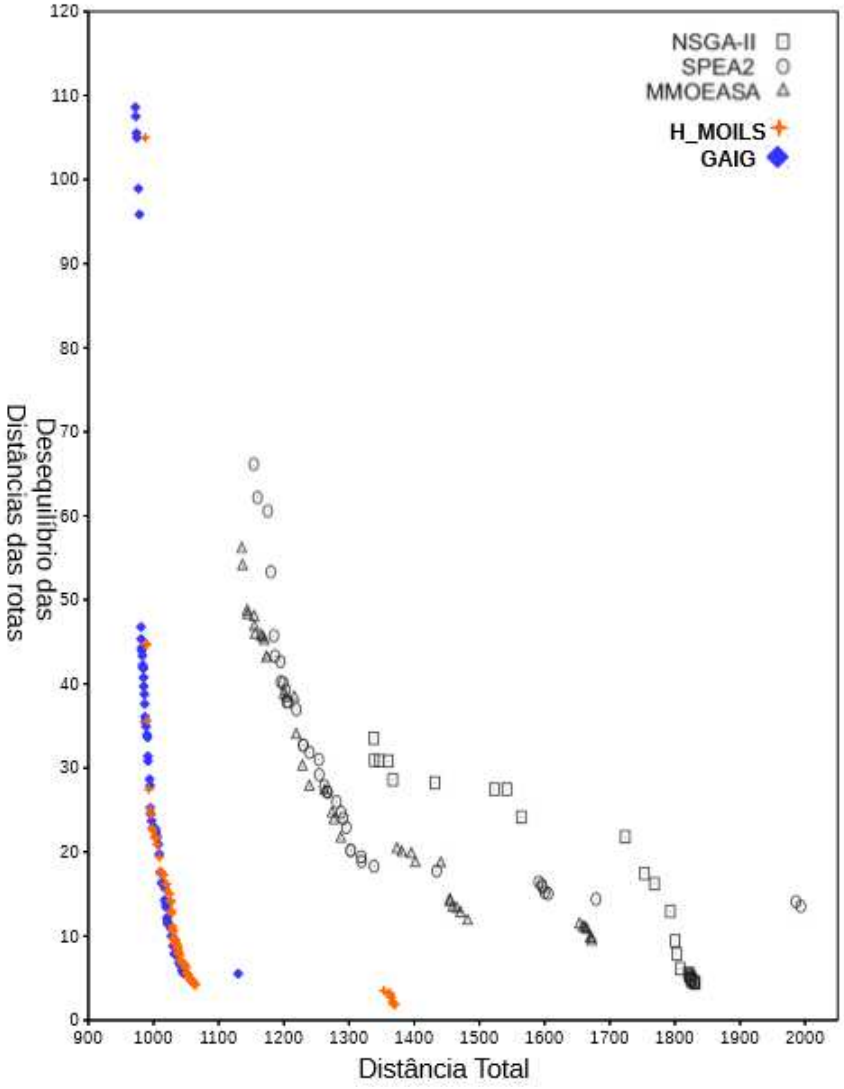

Figura 34 - Aproximação do Pareto-ótimo, para o problema R108 e objetivos Distância total(f1) e Desequilíbrio das distâncias das rotas(f2). Adaptado do trabalho do Banos et al. (2013)

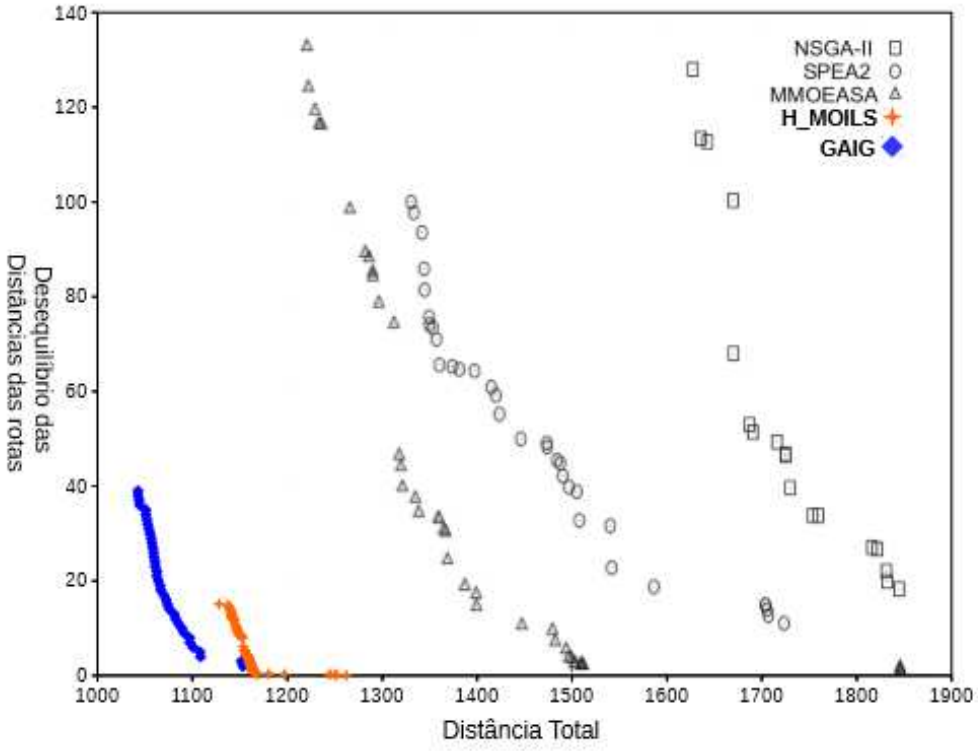

Figura 35 - Aproximação do Pareto-ótimo, para o problema RC203 e objetivos Distância total (f1) e Desequilíbrio das distâncias da srotas (f2). Adaptado do trabalho do Banos et al. (2013)

5.3. Experimentos com Distância Total (*f1*) e Desequilíbrio das Cargas (*f3*)

Nessa seção, são apresentados os resultados obtidos pelos algoritmos minimizando o par de objetivos: Distância Total (f1) e Desequilíbrio das Cargas (f3). As comparações são feitas através das métricas de performance apresentadas anteriormente, *hypervolume* e *epsilon+*.

Nas Tabelas 4, 5 e 6, são apresentadas as médias das métricas obtidas por cada algoritmo em cada instância. Podemos observar que, apesar de, em algumas instâncias, o desempenho do GAIG ser pior que o dos demais algoritmos, na maior parte dos problemas, O GAIG apresenta desempenho superior. Além disso, considerando o *hypervolume*, o algoritmo H_MOILS apresenta um bom desempenho nos problemas da classe C1, problemas com horizonte curto de roteamento, com objetivos não conflitantes.

A Figura 36 mostra o gráfico de médias resultantes do teste Tukey da Diferença Honestamente Significativa (HSD), cujo nível de confiança é de 95% para os algoritmos testados, considerando a métrica *hypervolume*. Nessa figura, o intervalo HSD de Turkey do algoritmo GAIG não se sobrepõe a nenhum outro intervalo, o que significa que a diferença desse algoritmo para os demais é significativa. Como verificamos nas Tabelas 4, 5, 6 e, agora, nos testes estatísticos, o GAIG tem desempenho superior em relação aos demais algoritmos que utilizam a métrica *hypervolume* para os objetivos f1 e f3.

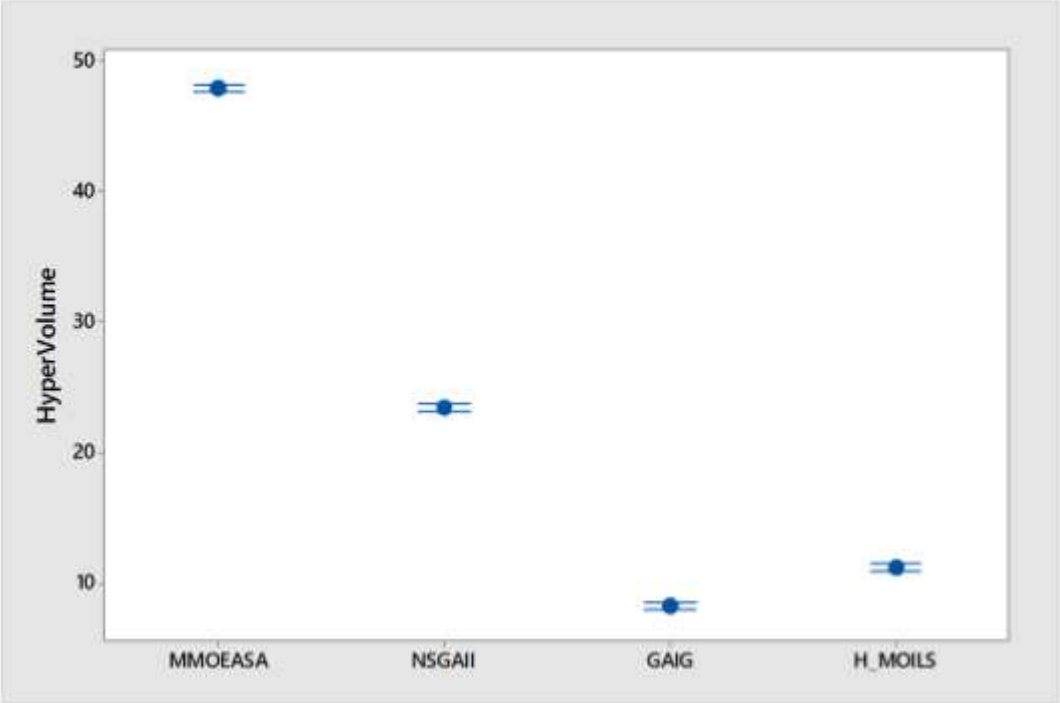

Figura 36 - Médias e Intervalos HSD de Turkey com nível de confiança de 95% *Hypervolume*.

A Figura 37 mostra o gráfico de médias resultantes do teste Tukey da Diferença Honestamente Significativa (HSD) com nível de confiança de 95% para os algoritmos testados, considerando a métrica *epsilon+*. Novamente, o algoritmo GAIG não se sobrepõe a nenhum outro intervalo, mostrando ser significativamente diferente dos outros algoritmos. Comprovando também os resultados indicados nas Tabelas 4, 5 e 6, observamos que o GAIG apresenta melhores resultados utilizando a métrica de comparação *epsilon+* nos objetivos *f1* e *f3*.

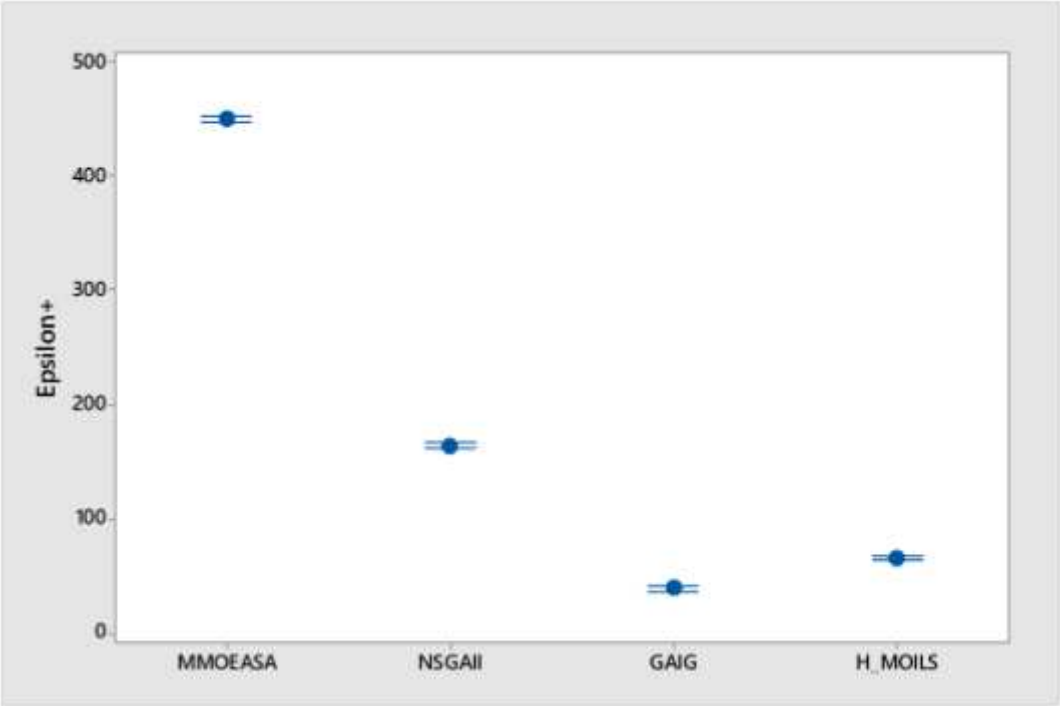

Figura 37 - Médias e Intervalos HSD de Turkey com nível de confiança de 95% Epsilon+.

Tabela 4 - Estão apresentadas a média das métricas de performance, considerando as 30 iterações para os objetivos Distância total (f1) e Desequilíbrio das Cargas (f3). Os problemas apresentados são da classe C. O valor em negrito indica o melhor valor para o problema.

	Hypervolume				Epsilon+			
Problemas	MMOEASA	NSGAII	GAIG	H_MOILS	MMOEASA	NSGAII	GAIG	H_MOILS
C101	**33,38**	58,17	70,28	47,29	358,6	30,61	30	**23,02**
C102	26,48	35,11	26,2	**24,18**	155,3	58,35	**15**	22,81
C103	28,37	36,21	**7,622**	17,74	126,7	187,8	**23,83**	52,98
C104	37,35	52,59	**8,801**	18,73	186,3	300,2	**32,78**	69,83
C105	36,31	40,15	43,67	**26,2**	292,1	44	**24**	16,06
C106	23,03	9,474	12,94	**7,687**	432,4	59,77	**22,33**	17,25
C107	31,27	18,24	28,03	**12,42**	424,1	59,04	**22**	14,23
C108	25,95	22,62	19,87	**10,6**	465,8	119,7	**15,67**	23,57
C109	45,85	37,06	**7,197**	18,4	458,4	295,9	**10,39**	73,15
C201	70,36	1,445	**0,1921**	1,628	481,8	8,458	**5,9**	13,44
C202	68,54	6,756	**0,5382**	5,194	447,4	46,95	**3,335**	34,11
C203	69,54	16,82	**1,015**	9,072	419,8	101,1	**6,341**	56,56
C204	79,05	29,07	**0,8901**	13,35	562,4	204,4	**8,466**	94,32
C205	76,82	8,821	**0,002167**	4,477	618,2	56,4	**0,192**	35,43
C206	76,07	8,748	**0,02226**	4,067	635,4	70,13	**0,3349**	35,84
C207	75,92	10,24	**0,06167**	4,217	611,1	81,01	**0,7421**	36,52
C208	78,08	9,228	**0,01941**	5,127	684,2	80,46	**0,3963**	45,5

Banos et al. (2013) também apresentaram as aproximações de fronteiras de Pareto (obtidas pelos algoritmos MMOEASA, SPEA2 e NSGAII) para os objetivos Distância Total (f1) e Desequilíbrio das Cargas (f3), nas duas instâncias R108 e RC203. Podemos observar, nas Figuras 38 e 39, que as soluções encontradas pelo GAIG e pelo H_MOILS foram superiores às publicadas por Banos et al. (2013). Podemos também observar que a diferença no problema R108 é menor entre o GAIG e o H_MOILS. Já no problema RC203, a diferença é bem grande.

Tabela 5 - Estão apresentadas a média das métricas de performance, considerando as 30 iterações para os objetivos Distância total (f1) e Desequilíbrio das Cargas (f3). Os problemas apresentados são da classe R. O valor em negrito indica o melhor valor para o problema.

	Hypervolume				Epsilon+			
Problemas	MMOEASA	NSGAII	GAIG	H_MOILS	MMOEASA	NSGAII	GAIG	H_MOILS
R101	24,45	18,71	**6,897**	13,75	240,7	80,61	**23,48**	52,67
R102	26,05	20,77	**6,536**	11,5	123,6	82,12	**22,61**	35,94
R103	28,92	22,87	**4,808**	10,37	175,1	120,5	**22,02**	46,15
R104	33,9	23,39	**5,213**	9,883	234,4	136,8	**30,06**	56,8
R105	32,19	19,15	**7,452**	11,88	261,7	102,5	**45,27**	61,83
R106	34,87	25,78	**4,963**	14,76	203,1	131,5	**25,14**	76,08
R107	40,54	27,15	**8,358**	14,49	225,1	123,1	**38,99**	66,1
R108	35,4	22,38	**5,056**	8,645	258,4	126,5	**30,33**	47,97
R109	44,08	26,48	**8,769**	15,26	278,7	139,1	**44,2**	70,91
R110	38,29	28,13	**4,718**	12,86	285,9	162,5	**25,08**	62,93
R111	32,2	23,89	**4,115**	10,46	276,1	170,3	**30,02**	80,69
R112	46,86	24,65	**4,307**	9,865	444,1	178,5	**31,56**	67,48
R201	48,89	19,83	**6,507**	6,921	516,1	192,2	**67,78**	68,59
R202	49,85	22,33	**6,637**	8,207	463,7	202,7	**63,17**	76,72
R203	50,27	24,48	**6,494**	9,143	467,2	221,8	**63,87**	84,6
R204	63,04	27,25	**6,955**	10,55	422,9	180,2	**50,39**	72,69
R205	74,47	27,85	**7,038**	8,529	628	227,4	**62,74**	71,96
R206	65,18	27,93	**8,937**	9,677	504,6	212,8	**70,88**	74,11
R207	69,82	33,52	**10,51**	13,16	505,9	239	**76,8**	95,07
R208	67,81	29,94	**5,386**	9,131	402,1	176,1	**35,23**	55,81
R209	73,9	27,74	9,855	**8,662**	620,6	228,6	86,22	**75,12**
R210	62,15	24,67	**5,815**	9,562	544,9	211,6	**52,55**	82,64
R211	61,22	30,05	**4,578**	8,808	561,3	265	**43,34**	80,38

Tabela 6 - Estão apresentadas a média das métricas de performance, considerando as 30 iterações para os objetivos Distância total (f1) e Desequilíbrio das Cargas (f3). Os problemas apresentados são da classe RC. O valor em negrito indica o melhor valor para o problema.

	Hypervolume				Epsilon+			
Problemas	MMOEASA	NSGAII	GAIG	H_MOILS	MMOEASA	NSGAII	GAIG	H_MOILS
RC101	29,01	13,14	**5,708**	9,951	396,8	78,85	**46,53**	60,01
RC102	24,55	15,24	**4,044**	9,208	295,2	88,05	**32,05**	59,82
RC103	36,93	23,36	**6,181**	14,36	318,4	136,2	**40,36**	76,6
RC104	39,03	22,1	**8,035**	12,78	325,4	139,6	**53,79**	86,9
RC105	25,45	14,63	**4,672**	11,48	347	113,6	**50,13**	85,64
RC106	24,08	14,09	**3,445**	8,599	409,5	106,5	**28,63**	78,28
RC107	27,86	16,93	**3,854**	8,559	436,8	147,9	**38,13**	88,74
RC108	39,78	19,39	**5,546**	9,814	388	127,3	**37,63**	62,28
RC201	46,31	18,97	7,208	**6,826**	732,5	289,5	114,1	**104,7**
RC202	44,2	21,5	**4,885**	6,846	697,3	330,2	**77,44**	107,1
RC203	48,18	21,49	**5,115**	7,846	685,3	290,7	**73,35**	109,8
RC204	41,23	19,65	**5,216**	7,085	506,3	232,3	**64,83**	88,02
RC205	51,67	23,64	**5,123**	8,419	727,8	320,9	**73,87**	116,6
RC206	57,39	18,81	**2,692**	5,518	840,2	266,4	**44,49**	82,74
RC207	51,2	19,61	**3,282**	6,436	764,8	280,2	**50,99**	95,37
RC208	57,88	26,05	**5,684**	6,122	855,9	377,2	**83,71**	90,86

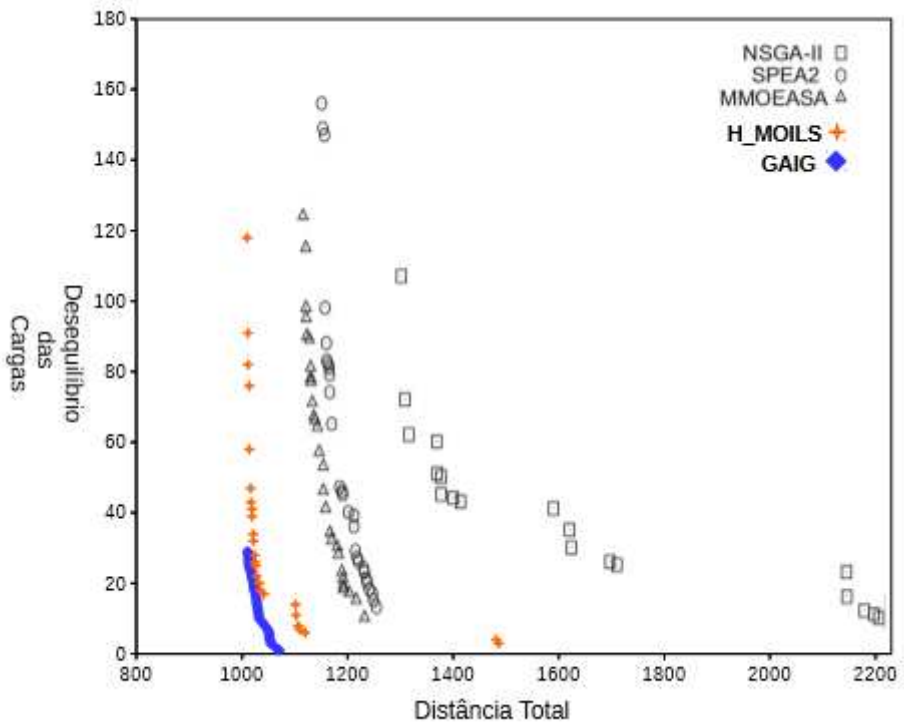

Figura 38 - Aproximação do Pareto-ótimo, para o problem R108 e objetivos Distância total (f1) e Desequilíbrio das cargas (f3). Adaptado do trabalho do Banos et al. (2013)

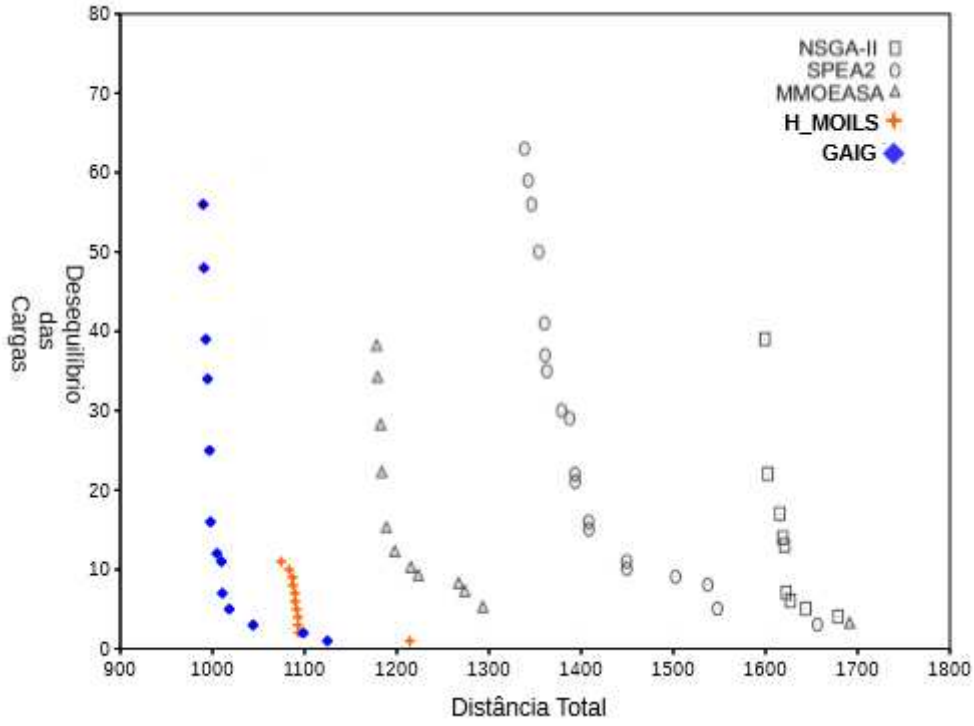

Figura 39 - Aproximação do Pareto-ótimo, para o problem RC203 e objetivos Distância total (f1) e Desequilíbrio das cargas (f3). Adaptado do trabalho do Banos et al. (2013)

6. Conclusões e trabalhos futuros

Este trabalho apresenta dois algoritmos híbridos para o Problema de Roteamento de Veículos com Janela de Tempo biobjetivo. O primeiro algoritmo H_MOILS combina a meta-heurística Iterated Local Search, os métodos Variable Neighborhood Descent e combinação de soluções através do uso do *crossover*. O segundo algoritmo GAIG combina o Algoritmo Genético com a etapa de intensificação utilizando o método Iterated Greedy.

Como o problema estudado é da classe NP-Difícil, abordagem exatas não conseguem encontrar a solução ótima em tempo polinomial. A fim de estudar métodos heurísticos, foi escolhida a implementação de métodos heurísticos para solucionar o problema.

Os dois algoritmos H_MOILS e GAIG foram testados utilizando os 56 problemas-teste de Solomon. Os resultados obtidos para os objetivos estudados superaram os encontrados na literatura. Nos experimentos, pudemos observar que ambas as heurísticas propostas são competitivas, porém a heurística GAIG apresentou melhor desempenho. Observamos que, para a métrica de *hypervolume* nos objetivos Distância Total e Desequilíbrio das Distâncias das rotas (f1 e f2), o algoritmo GAIG obteve melhor resultado em 46 dos problemas, enquanto o H_MOILS obteve melhor resultado em apenas 10. Considerando os objetivos Distância Total e Desequilíbrio das Cargas (f1 e f3), obtivemos resultados similares, nos quais o GAIG conseguiu melhor resultados em 48 problemas, enquanto o H_MOILS conseguiu em 7 instâncias. Considerando a métrica *epsilon+* nos objetivos (f1, f2), observamos que o GAIG obteve melhores resultados em 54 das instâncias, e o H_MOILS, em somente duas. Nos objetivos (f1, f3), o GAIG obteve melhor resultado em 50 instâncias, enquanto o H_MOILS obteve em 6. Durante o desenvolvimento do trabalho, algumas decisões foram tomadas, no projeto, para execução dos testes. A calibração do algoritmo GAIG foi feita com o auxílio de um *cluster*.

Os algoritmos propostos foram comparados com a reimplementação dos algoritmos da literatura, MMOEASA e NSGAII. No Capítulo 5, foram

apresentados os resultados da comparação. Pode-se observar que os algoritmos propostos obtiveram melhores resultados em quase todas as instâncias. Com base nisso, podemos afirmar que o desempenho dos algoritmos propostos, GAIG e H_MOILS, foi superior aos algoritmos da literatura nas condições do experimento. Também foi possível compará-los com alguns conjuntos de solução disponibilizados na literatura, nos quais é possível observar a superioridade dos algoritmos propostos. Através dos testes estatísticos, pudemos validar a boa performance dos algoritmos propostos.

Além disso, foi possível comparar os dois algoritmos propostos, GAIG e H_MOILS. Comparando-os pudemos observar que o algoritmo GAIG apresentou desempenho superior. Uma análise mais detalhada de possíveis vizinhanças poderia impactar positivamente nos resultados do H_MOILS. Por fim, tipos de movimentos diferentes de perturbação poderiam encontrar melhores soluções.

Como trabalho futuro, sugere-se a adaptação dos algoritmos propostos para outros conjuntos de objetivos, e também a aplicação deles em problemas com três objetivos. Também esperamos que bons resultados sejam obtidos através da aplicação para outras variantes do Problema de Roteamento de Veículos.

7. Referências Bibliográficas

AGUIRRE, H. E.; TANAKA, K. Working principles, behavior, and performance of moeasonmnk-landscapes. *European Journal of Operational Research*, 181(3):1670–1690.

ALVANGUERA, G. B.; MATEUS, G. R.; DE TOMI, G. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34(6):1561–1584.

BALDACCI, R; BALTARRA, M.; VIGO, D. Routing a heterogeneous fleet of vehicles. *In the vehicle routing problem*: latest advances and new challenges, pages 3–27. Springer.

BALDACCI, R.; TOTH, P.; VIGO, D. *Recent advances in vehicle routing exact algorithms*. 4OR, 5(4):269–298.

BALDACCI, R.; TOTH, P; VIGO, D. Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research,* 175(1):213–245.

BANOS, R.; ORTEGA, J.; GIL, C., MARQUEZ, A. L.; DE TORO, F. (2013). *A hybrid meta-heuristic formulti-objective vehicle routing problems with time windows. Computers & Industrial Engineering*, 65(2):286–296.

CHIANG, T.C; HSU, W.-H. Aknowledge-base devolutionary algorithm for the multiobjective vehicle routing problem with time windows. *Computers & Operations Research*, 45:25–37.

COELLO, C. A. C.; ZACATENCO, C. S. P. 20 years of evolutionary multiobjective optimization: what has been done and what remains to be done. *Computational Intelligence*: Principles and Practice, pages 73–88.

CORBERÁN, A.; FERNÁNDEZ, E.: LAGUNA, M.; MARTI, R.et al. (2002). Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the operational research society*, 53(4):427–435.

DANTZIG, G. B.; RAMSER, J. H.. The truck-dispatching problem. *Management science*, 6(1):80–91.

DE BACKER, B.; FURNON, V.; PROSSER, P. KILBY, P.; SHAW, P. Local search in constraint programming: Application to the vehicle routing problem. In: *Proc. CP-97 Workshop Indust. Constraint-Directed Scheduling*, pages 1–15. Citeseer.

DE JONG, K. A. *Analysis of the behavior of a class of genetic adaptive systems*.

DEB, K. Current trends in evolutionary multi-objective optimization. *International Journal for Simulation and Multidisciplinary Design Optimization*, 1(1):1–8.

DEB, K.; JAIN, S. (2002). Running performance metrics for evolutionary multi-objective optimizations. In: *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning* (SEAL'02), (Singapore), pages 13–20. Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02), (Singapore).

DEB, K., MOHAN, M.; MISHRA, S. Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary computation*, 13(4):501–525.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. In: *Evolutionary Computation*, IEEE Transactions on, 6(2):182–197.

DEB, K; TIWARI, S. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185(3):1062–1087.

DESROCHERS, M.; DESROSIERS, J.; SOLOMON, M. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354.

EDGEWORTH, F. Y. *Mathematical psychics*: An essay on the application of mathematics to the moral sciences. Number 10. CK Paul.

FISHER, M. L.; JAIKUMAR, R.. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124.

FONSECA, C. M.; FLEMING, P. J., et al. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: *ICGA*, volume 93, pages 416–423. Citeseer.

GARCIA-NAJERA, A.; 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.

GEIGER, M.J. The interactive pareto iterated local search (ipils) metaheuristic and its application to the biobjective portfolio optimization problem. In: Computational Intelligence; In: *Multicriteria Decision Making*, IEEE Symposium on, pages 193–199. IEEE.

GENDREAU, M.; LAPORTE, G.; POTVIN, J.-Y. Metaheuristics for the capacitated vrp. In: *The vehicle routing problem*, pages 129–154. Society for Industrial and Applied Mathematics.

GENDREAU, M.; LAPORTE, G.; SÉGUIN, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. In: *Transportation science*, 29(2):143–155.

GHOSEIRI, K.; GHANNADPOUR, S. F. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. In: *Applied Soft Computing*, 10(4):1096–1107.

GOLBERG, D. E. Genetic algorithms in search, optimization, and machine learning. Addion wesley, 1989.

JOSEFOWIEZ, N.; SEMET, F.; TALBI, E.-G. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2):293–309.

KALLEHAUGE, B. Formulations and exact algorithms for the vehicle routing problem with time windows. In: *Computers & Operations Research*, 35(7):2307– 2330.

KHARE, V., YAO, X.; DEB, K. Performance scaling of multi-objective evolutionary algorithms. In: *Evolutionary Multi-Criterion Optimization*, pages 376–390. Springer.

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

KNOWLES, J.; THIELE, L.; ZITZLER, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Tik report, 214:327–332.

KOHL, N. Exact methods for time constrained routing and related scheduling problems. PhD thesis, IMM.

KOLLAT, J. B.; REED, P. M.. Comparing state-of-the-art evolutionary multi-objective algorithms for long-term groundwater monitoring design. In: *Advances in Water Resources*, 29(6):792–807.

KONAK, A.; COIT, D. W.; SMITH, A. E. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety, 91(9):992–1007.

LAPORTE, G. (2009). Fifty years of vehicle routing. In: *Transportation Science,* 43(4):408–416.

LAPORTE, G.; GENDREAU, M.; POTVIN, J.-Y.: SEMET, F. (2000). Classical and modern heuristics for the vehicle routing problem. In: *International transactions in operational research*, 7(4-5):285–300.

LAUMANNS, M.; THIELE, L.; DEB, K..; ZITZLER, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. In: *Evolutionary computation*, 10(3):263–282.

LENSTRA, J. K.; KAN, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

LOURENÇO, H. R.; STUTZLE, T. Iterated local search. F. Glover Eds. Kluwer Academic.

LOURENÇO, H. R., MARTIN, O. C.; STUTZLE, T. (2003). Iterated local search. *Springer*.

MINSKY, M. (1961). Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. In: *Computers & Operations Research*, 24(11):1097–1100.

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

MOSCATO, P.; COTTA, C. (2003). A gentle introduction to memetic algorithms. In: *Handbook of metaheuristics*, pages 105–144. Springer.

OMBUKI, B.; ROSSs, B. J.; and HANSHAR, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. In: *Applied Intelligence*, 24(1):17–30.

OSMAN, I. H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. In: *Annals of operations research*, 41(4):421–451.

PACHECO, J.; MARTÍ, R. Tabu search for a multi-objective routing problem. *Journal of the Operational Research Society*, 57(1):29–37.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

PAQUETE, L., CHIARANDINI, M.; STUTZLE, T. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: *Metaheuristics for Multiobjective Optimisation*, pages 177–199. Springer.

POTVIN, J.-Y.; BENGIO, S. (1996). The vehicle routing problem with time windows part II: genetic search. In: *INFORMS journal on Computing*, 8(2):165–172.

PRINS, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002.

RAHOUAL, M., KITOUN, B., MABED, M.-H.; BACHELET, V.; Benameur, F. Multicriteria genetic algorithms for the vehicle routing problem with time windows. In: *4th Metaheuristics International Conference,* pages 527–532.

RIBAS, S. SUBRAMANIAN, A.; COELHO, I.; OCHI, L., SOUZA, M., DA PÁTRIA, R. P.; BLOCO, E. (. An algorithm based on iterated local search and set

partitioning for the vehicle routing problem with time windows. Welcome Note, page 145.

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

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

SRINIVAS, N. and DEB, K. (1994). Muiltiobjective optimization using non-dominated sorting in genetic algorithms. In: *Evolutionary computation*, 2(3):221–248.

TAILLARD, É: BADEAU, P.; GENDREAU, M.; GUERTIN, F., POTVIN, J.-Y. A taboo search heuristic for the vehicle routing problem with soft time windows. In: *Transportation science*, 31(2):170–186.

TAN, K., CHEW, Y.; LEE, L. .A hybrid multiobjective evolutionary algorithm for solving vehicle routing problems with time windows. In: *Computational Optimization and Applications*, 34(1):115–151.

TAN, K. C., CHEONG, C. Y.: GOH, C. K. Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. *European Journal of operational research*, 177(2):813–839.

TAN, K. C.; GOH, C. K., YANG, Y.; LEE, T. H. Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, 171(2):463–495.

THEODORSSON-NORHEIM, E. Kruskal-wallis test: Basic computer program to perform nonparametric one-way analysis of variance and multiple comparisons on ranks of several independent samples. In: *Computer methods and programs in biomedicine*, 23(1):57–62.

TOTH, P.; VIGO, D. An exact algorithm for the vehicle routing problem with backhauls. In: *Transportation science*, 31(4):372–385.

TOTH, P.; Vigo, D. (1998). Exact solution of the vehicle routing problem. In: *Fleet management and logistics*, pages 1–31. Springer.

TOTH, P.; VIGO, D. *The vehicle routing problem*. Siam.

TOTH, P.; VIGO, D. . Models, relaxations and exact approaches for the capacitated vehicle routing problem. In: *Discrete Applied Mathematics*, 123(1):487– 512.

ZITZLER, E.; LAUMANNS, M.; BLEUER, S. (2004). A tutorial on evolutionary multiobjective optimization. In: *Metaheuristics for multiobjective optimization*, pages 3–37. Springer.

ZITZLER, E.; LAUMANNS, M.; THIELE, L.. Spea2: *Improving the strength Pareto evolutionary algorithm*.

ZITZLER, E.; THIELE. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In: *Parallel problem solving from nature*—PPSN V, Springer, p. 292–301.

ZITZLER, E.; THIELE, L.; LAUMANNS, M.; FONSECA, C. M.,; DA FONSECA, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation,* IEEE Transactions on, 7(2):117–132.