Partition-based crossover operator applied to vehicle routing problem

Elcio Cezario Sanches Júnior[1], Gabriel de Abreu[1], Vinícius Macedo Noda[1], Ozeas Quevedo de Carvalho[1], Josimar da Silva Rocha[2], Renato Tinós[3], Danilo Sipoli Sanches[1]

[1]Departamento Acadêmico de Computação
Universidade Tecnológica Federal do Paraná (UTFPR)
86.300-000 – Cornélio Procópio, PR – Brazil

[2]Departamento Acadêmico de Matemática
Universidade Tecnológica Federal do Paraná (UTFPR)
86.300-000 – Cornélio Procópio, PR – Brazil

[3]Departamento de Computação e Matemática
Faculdade de Filosofia, Ciências e Letra de Ribeirão Preto
Universidade de São Paulo (USP)
14.040-901 – Ribeirão Preto, SP – Brazil

{elcioc, gabreu, viniciusnoda} @alunos.utfpr.edu.br,

ozeasx@gmail.com, rtinos@ffclrp.usp.br,

{jrocha, danilosanches}@utfpr.edu.br

Abstract. *This paper proposes the crossover operator Generation Partition Crossover 2 (GPX2) which is an adaptation of the Partition Crossover (PX) applied in the vehicle routing problem (VRP). The GPX2 operator uses the concept of Ghost nodes and a specific fitness function to add penalty to non-feasible VRP solutions. Also, GPX2 operator is compared with the Order Crossover (OX) operator, validating the proposed method in the VRP.*

Resumo. *Este artigo propõe uma modificação do operador de cruzamento* Generation Partition Crossover 2 *(GPX2) que é uma adaptação do operador de cruzamento* Partition Crossover *(PX). Os operadores GPX2 e PX foram inicialmente validados apenas no Problema do Caixeiro Viajante. Diante disto, a proposta deste trabalho consiste na adaptação do operador GPX2 ao* Problema de Roteamento de Veículos*. O GPX2 utiliza o conceito de* Ghost nodes *e uma função* fitness *que considera penalizações em soluções não factíveis do Problema de Roteamento de Veículos. O método é validado comparando o desempenho do* Generation Partition Crossover 2 *com o operador* Order Crossover *que é bastante utilizado em Problemas de Roteamento de Veículos.*

1. Introdução

Problemas de Roteamento de Veículos (PRV) são problemas de otimização que buscam minimizar o percurso total de uma frota de veículos que deve atender a demanda de

um conjunto de clientes e são bastante utilizados em logística, como podemos ver em [Rodrigues 2008].

Diversos operadores de cruzamento foram aplicados para a resolução do Problema de Roteamento de Veículos (PRV). Em [Nagata 2007] o *Edge Assembly Crossover* (EAX) é aplicado ao PRV dentro de um algoritmo evolutivo. No entanto, como o EAX é projetado originalmente para o Problema do Caixeiro Viajante (PCV), então é necessária uma heurística de reparação para a obtenção de resultados satisfatórios na resolução do PRV. Em [Puljic and Manger 2013] encontramos o operador *Order Crossover* (OX) que é um operador bastante utilizado na resolução do PRV e que geralmente é utilizado como base para comparação com outros operadores. Outro operador de cruzamento determinístico originalmente projetado para o Problema do Caixeiro Viajante (PCV) é o *Partition Crossover* (PX). O *Partition Crossover* (PX) é um operador que recombina duas soluções pais para gerar duas outras soluções e tem a vantagem de recombinar ótimos locais para produzir, com alta probabilidade, novos ótimos locais. A descrição do operador (PX) pode ser encontrada em [Tinós et al. 2019].

Em [Perez-Wohlfeil et al. 2018, Puljic and Manger 2013] encontramos dois trabalhos que buscam aplicar o PX ao PRV, partindo da premissa de que o PX pode produzir bons resultados se aplicado ao PRV, uma vez que o PRV pode ser visto como uma generalização do PCV. No entanto, nestes trabalhos são impostas certas restrições para que o PX seja empregado. Em [Tlili et al. 2015] o PX é empregado sem alterações, exigindo que as soluções sejam gêmeas. Soluções gêmeas do PRV são soluções que possuem as mesmas pétalas[1], mas com subrotas diferentes. Em estudo mais recente [Perez-Wohlfeil et al. 2018] leva em conta a relação entre o depósito e os clientes vizinhos ao depósito para efetuar o cruzamento. Nestes trabalhos, sempre é aplicado um mecanismo para reparar as soluções infactíveis criadas pelo PX.

Neste trabalho utilizaremos o PX para a resolução do PRV, reescrevendo a representação do problema para que as soluções apresentadas sejam circuitos hamiltonianos para serem suportadas pelo PX, não exigindo nenhum tipo de restrição adicional sobre as soluções do PRV, o que torna o método proposto mais genérico.

Quanto a organização deste trabalho, na seção 2 descreve-se o problema abordado e o operador de cruzamento GPX2 que será adotado para a obtenção de soluções para o PRV. Na seção 3 descreve-se o método proposto, o algoritmo genético, a configuração dos experimentos e as instâncias de teste. Os resultados obtidos são apresentados e discutidos na seção 4.

2. Descrição do Problema e do Operador de Cruzamento Proposto

2.1. O Problema de Roteamento de Veículos

O PRV utilizado neste trabalho é o *capacitated vehicle routing problem* (CVRP) que consiste em encontrar rotas que minimizam a soma das distâncias percorridas por um conjunto de k veículos para cobrir um conjunto de n clientes (ou cidades) partindo de um local de depósito e voltando para este local de depósito uma única vez, sendo que cada veículo $t \in \{1, 2, \cdots, k\}$ só pode transportar certa quantidade de mercadorias indicada

[1]Em soluções PRV, uma pétala é o conjunto de clientes atendidos pelo mesmo veículo

pela capacidade C e cada cliente $i \in \{1, \cdots, n\}$ terá uma demanda c_i de mercadorias e será atendido uma única vez por um único veículo, sendo $c_i \leq C$.

Matematicamente, o CVRP pode ser representado como em [Vigo and Toth 2014] por um grafo não direcionado $G = (V, A)$ formado por um conjunto de vértices $V = \{0, 1, \cdots, n\}$ e um conjunto de arestas $A = \{(i, j) \mid i < j\}$, onde

- o vértice 0 representa o depósito;
- cada cliente é representado por um vértice $i \in \{1, \cdots, n\}$;
- cada aresta (i, j) é associado a um valor d_{ij} ou d_{ji} que representa o custo da viagem entre i e j;
- cada vértice $i \in \{0, 1, \cdots, n\}$ é associado um valor real positivo q_i se $i \neq 0$ e $q_0 = 0$.

O CVRP consiste em minimizar

$$\sum_{t=1}^{k} \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_{ijt}$$

sujeito as seguintes condições:

$$\sum_{i=1}^{n} q_i x_{it} \leq C, \forall t \in \{1, 2, \cdots, k\}$$

$$\sum_{t=1}^{k} x_{it} = 1, \forall i \in \{1, 2, \cdots, n\}$$

$$x_{ijt} = \begin{cases} 1, & \text{se o veículo } t \text{ percorreu a aresta } (i, j) \\ 0, & \text{se o veículo } t \text{ não percorreu a aresta } (i, j) \end{cases}$$

$$x_{ik} = \begin{cases} 1, & \text{se o veículo } t \text{ visitou o vértice } i \\ 0, & \text{se o veículo } t \text{ não visitou o vértice } i \end{cases}$$

2.2. *Generalized Partition Crossover 2*

O GPX2 [Tinós et al. 2019] é a versão mais recente do PX, um operador de recombinação que tem como princípio de funcionamento preservar o que há de comum entre as soluções pais e otimizar as diferenças entre elas para formar as soluções filhas [Whitley et al. 2009]. As diversas versões do PX apresentam complexidade de tempo $O(n)$ e diferem entre si na forma como são encontradas e utilizadas as componentes recombinantes, e na variante do PCV tratado, se simétrico ou assimétrico.

Uma componente recombinante é um componente conectado do grafo de recombinação que, quando recombinado com outros componentes recombinantes sempre produz soluções factíveis, i.e., ciclos Hamiltonianos. O grafo de recombinação é o grafo resultante da remoção das arestas comuns no grafo formado pela união entre as duas soluções pais. O operador, então, escolhe o melhor caminho em cada componente recombinante, construindo assim a melhor solução possível explorando, com complexidade de tempo O(n), um número de soluções possíveis que cresce exponencialmente

com o número de componentes encontrados. O PX, em todas as suas versões, possui duas propriedades consideradas importantes em operadores de cruzamento: É um operador respeitoso (*respectful*), pois utiliza apenas as arestas existentes nos pais para criar as soluções filhas e; transmite alelos comuns (*transmit alleles*), pois transmite para as soluções descendentes o que as soluções pais têm em comum [Sanches et al. 2017].

A Figura 1 contém as etapas executadas pelo GPX2 para encontrar componentes recombinantes. Em primeiro lugar, o grafo união das soluções pais é criado (Figura 1a). Para ampliar as possibilidades de particionamento, os vértices de grau 4 são duplicados em "vértices fantasmas", ou *ghost nodes* (Figura 1b). As componentes conectadas são encontradas retirando-se as arestas comuns da união dos grafos pais (Figura 1c) e são classificadas como válidas (i.e., componente recombinante) ou inválidas do seguinte modo. Uma componente conectada é válida (i.e., é uma componente recombinante) caso as soluções entrem e saiam da partição pelos mesmo vértices. A componente resultante da união de todas as componentes inválidas também é uma componente recombinante.

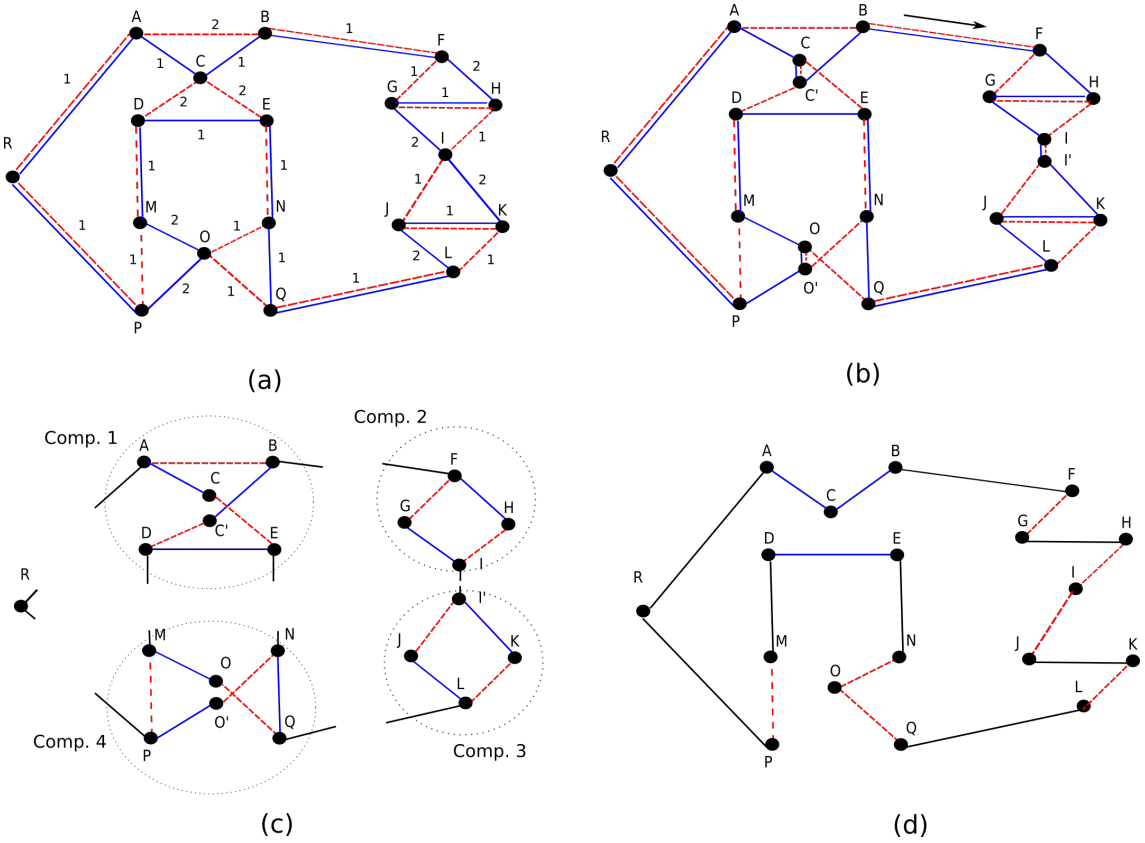

Figura 1. Exemplo de recombinação por GPX2. (a) Gráfico de união composto pelos pais 1 (linha sólida azul) e 2 (linha tracejada vermelha); (b) Nós fantasmas são inseridos considerando-se uma dada direção para o fluxo; (c) Depois de remover as arestas comuns, GPX2 identifica 4 componentes recombinantes; (d) O descendente é gerado escolhendo-se o menor caminho (de um ou outro pai) para cada componente. O descendente tem custo 18, enquanto que os pais tem custo 24 (pai 1) e 21 (pai 2).

Outro mecanismo utilizado pelo GPX2 é a fusão de componentes inválidas. O operador testa se a união de duas componentes inválidas forma uma componente válida.

Esse procedimento é repetido até que não hajam mais combinações ou até um limite predeterminado, mantendo a complexidade de tempo do operador.

Encontradas e classificadas as componentes, as arestas comuns entre os pais são copiadas para as soluções descendentes. Então, o operador escolhe o melhor percurso em cada componente válida (Figura 1d).

Devido a forma como o operador aplica sua heurística de construção, garante-se que, dada duas soluções de entrada, a solução de saída tenha *fitness* melhor ou igual ao *fitness* das soluções de entrada.

3. Materiais e Métodos

3.1. Método proposto

Para aplicar o GPX2 ao PRV é necessária uma adaptação do grafo gerado pelo problema para um formato aceito pelo operador GPX2, garantindo a restrição de ciclo hamiltoniano, necessária para seu funcionamento.

O método é dividido em duas etapas. A principal contribuição deste trabalho está no mecanismo da primeira etapa, que é a sucessiva duplicação do depósito, de modo similar ao que é feito para criar os *ghost nodes*. A Figura 2a contém a sobreposição de duas soluções do PRV. Percorrendo as rotas, a cada retorno, o depósito é duplicado em um depósito fantasma de diferente identificação e mesma posição (Figura 2b). Desse modo, a solução se torna um circuito Hamiltoniano, o tamanho do percurso total é preservado e não há vértices com mesma identificação. Assim, as soluções podem ser recombinadas pelo GPX2 e não há nenhuma restrição adicional sobre elas, como acontece nos trabalhos encontrados na literatura.

A segunda etapa, semelhante ao que é proposto em [Perez-Wohlfeil et al. 2018], é a reordenação das rotas a partir da similaridade entre elas, criando mais correlações entre as sub-rotas do grafos pais dentro do GPX2, o que possibilita que mais partições sejam encontradas (Figura 2c).

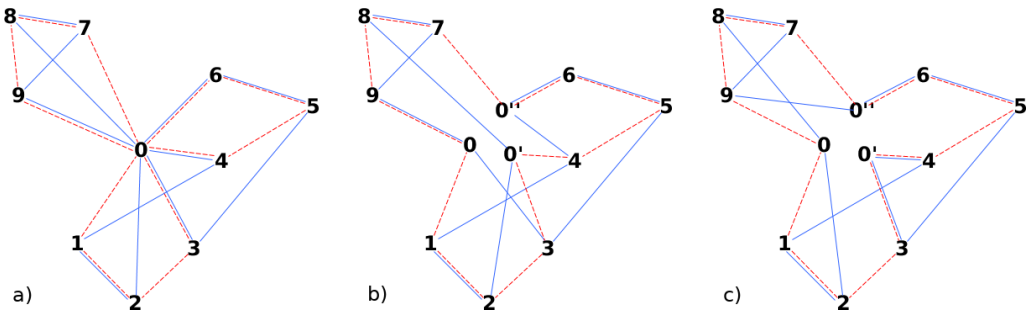

Figura 2. Etapas do método proposto: a) Sobreposição das soluções; b) Duplicação dos depósitos; c) Reordenação.

Ao final, as arestas ligadas aos depósitos fantasmas são religadas ao depósito original, fazendo com que a solução volte a ser uma solução do PRV.

3.2. Algoritmo Genético

Para testar a eficiência do método proposto, implementamos o GPX2 modificado em um algoritmo genético tradicional. As seções a seguir contém os detalhes de configuração do

algoritmo genético implementado.

3.2.1. K-*means*

Para a geração da população inicial, 3 estratégias foram testadas (ver Seção 3.4). Em uma delas, adotou-se o método proposto em [Geetha et al. 2009]. Esse método implementa uma busca local no K-*means* para tornar possível sua aplicação ao PRV. Após o K-*means* determinar os centroides, cada *cluster* formado é usado para construir uma rota. Busca local é então aplicada e tem como objetivo verificar se a carga total da rota não viola a restrição de capacidade. Caso isso ocorra, escolhe-se o centroide mais próximo com espaço disponível para receber aquele cliente.

3.2.2. Função *Fitness*

Na abordagem proposta, o problema de roteamento de veículos com limitação de capacidade utilizará a função *fitness* que penaliza as soluções com rotas que não respeitam a restrição de capacidade. Desta forma, se x é uma rota escolhida para os veículos, a função *fitness* adotada será

$$F(x) = f(x) + \Delta(x) \tag{1}$$

onde:

- $f(x) = \sum_{t=1}^{k} \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_{ijt}$ é a distância total percorrida pelos veículos;
- $\Delta(x) = (n+k) \left(\sum_{t=1}^{k} \delta_t(x) \right) d_{max}$ é uma função para penalização dos percursos;
- $d_{max} = \max_{0 \leq i < j \leq n} d_{ij}$ é a distância máxima entre dois pontos do circuito;
- $\delta_t(x) = \left\lfloor \frac{p_t(x)}{C+1} \right\rfloor$ é o maior inteiro menor ou igual a $\frac{p_t(x)}{1+C}$ é chamada de função de penalização no percurso do caminhão t com $p_t(x) = \sum_{i=1}^{n} c_i x_{it}, t \in \{1, ..., k\}$;
- $x_{it}, x_{ijt} \in \{0, 1\}, \forall i, j \in \{0, 1, \cdots, n\}, t \in \{1, 2, \cdots, k\}$ são variáveis binárias que indicam os trajetos percorridos pelos veículos.

Observe que no caso de $C \geq \sum_{i=1}^{n} c_i$, teremos que $\Delta(X) = 0$ e o problema se restringe ao problema de roteamento de veículos sem limitação de capacidade. Neste caso, quando $k = 1$, o problema se restringe ao problema do caixeiro viajante.

3.2.3. Mutação

Como em [Prins 2004], a mutação utilizada aplica uma busca local de 6 movimentos e em seguida uma busca 2-opt. Os movimentos realizados envolvem trocas e reinserções entre os elementos presentes no cromossomo. Para evitar que a restrição de capacidade seja violada, os clientes têm suas posições alteradas. Os depósitos são mantidos sempre nas mesmas posições, tanto nos movimentos realizados, quanto no 2-opt. Testes preliminares demonstraram a eficiência da mutação adotada.

3.3. Reparação

O procedimento de reparação adotado nesse estudo é baseado em um algoritmo chamado *Taburoute*, descrito em [Cordeau et al. 2002]. A reparação tem como objetivo melhorar a qualidade da solução, que geralmente acaba sendo prejudicada devido aos operadores genéticos clássicos não possuírem heurísticas para o PRV. Dessa forma, é possível gerar novas soluções válidas contendo arestas presentes no ótimo global. O procedimento adotado para realizar a reparação das soluções consiste em realizar uma varredura na solução, verificando quais rotas violam a restrição de carga. A partir dessa verificação, os clientes com maior demanda são removidos dessas rotas até que a restrição seja atendida. Após essa etapa, os clientes removidos são realocados em outras rotas, de forma que seja possível eliminar ou reduzir o quanto a restrição não é atendida.

3.4. Configuração dos Experimentos

Por ser um operador de recombinação clássico, escolheu-se o OX para a comparação com o GPX2. Os experimentos foram realizados com três configurações de população inicial diferentes, descritas a seguir:

- **Configuração 1** - Geração por operador de mutação: a população é gerada a partir de uma disposição aleatória na qual ocorre então a aplicação de seis iterações do operador de mutação;
- **Configuração 2** - Geração por K-*means* e operador de mutação: A população inicial é obtida a partir de uma clusterização realizada com o algoritmo K-*means*, após a clusterização, seis iterações do operador de mutação são aplicadas na disposição, para melhoria da distância obtida pela clusterização.
- **Configuração 3** - Geração aleatória padrão: nesse método a população é obtida a partir de uma geração aleatória na qual é aplicado somente o método de reparação de soluções.

3.5. Instâncias de teste

As instâncias de teste utilizadas no estudo estão detalhadas na Tabela 1. São instâncias de diferentes autores e podem ser encontradas no repositório CVRPLIB [Uchoa et al. 2017].

Tabela 1. Instâncias de teste

Instância	Clientes	Veículos	Ótimo global conhecido
E-n51-k5	51	5	521
A-n65-k9	65	9	1174
F-n72-k4	72	4	237
F-n135-k7	135	7	1162

4. Resultados

As tabelas 2, 3, 4 e 5 contém os resultados dos experimentos para cada uma das instâncias de teste escolhidas. A segunda coluna indica a configuração do experimento realizado, como descrito na seção 3.4. A terceira coluna contém a média e desvio padrão da melhor solução encontrada nas execuções. A última coluna apresenta, em percentagem, quanto

Tabela 2. Resultados - E-n51-k5

Crossover	Configuração	Solução Final Média ± Desvio Padrão	Ótimo conhecido (%) Proximidade	Acertos
GPX2	1	526.667 ± 6.342	98.912	60.0
	2	526.000 ± 5.754	99.04	60.0
	3	533.111 ± 10.796	97.675	30.0
OX	1	653.100 ± 37.209	74.645	0.0
	2	549.111 ± 18.663	94.604	10.0
	3	625.556 ± 40.250	79.932	0.0

Tabela 3. Resultados - A-n65-k5

Crossover	Configuração	Solução Final Média ± Desvio Padrão	Ótimo conhecido (%) Proximidade	Acertos
GPX2	1	1326.500 ± 366.284	87.01	10.0
	2	1230.600 ± 27.645	95.179	0.0
	3	1226.300 ± 25.076	95.545	0.0
OX	1	1724.300 ± 87.421	53.126	0.0
	2	2538.500 ± 1932.495	0	0.0
	3	1775.400 ± 102.772	48.773	0.0

Tabela 4. Resultados - F-n72-k4

Crossover	Configuração	Solução Final Média ± Desvio Padrão	Ótimo conhecido (%) Proximidade	Acertos
GPX2	1	242.700 ± 3.035	97.595	10.0
	2	244.000 ± 1.612	97.046	0.0
	3	244.500 ± 1.500	96.835	0.0
OX	1	304.600 ± 15.259	71.477	0.0
	2	249.800 ± 7.652	94.599	0.0
	3	303.900 ± 17.265	71.772	0.0

Tabela 5. Resultados - F-n135-k7

Crossover	Configuração	Solução Final Média ± Desvio Padrão	Ótimo conhecido (%) Proximidade	Acertos
GPX2	1	1202.375 ± 27.395	96.525	0.0
	2	1225.250 ± 22.426	94.557	0.0
	3	1226.714 ± 49.882	94.431	0.0
OX	1	1951.375 ± 110.059	32.068	0.0
	2	1275.125 ± 48.103	90.265	0.0
	3	2003.250 ± 134.420	27.603	0.0

a média das soluções encontradas se aproxima da melhor solução conhecida, bem como em quantas execuções (”Acertos”) se alcançou a melhor solução conhecida.

A partir da verificação dos resultados, observa-se que o método proposto consegue, no geral, se sobressair ao OX, principalmente quando a clusterização com K-*means* não é aplicada (Configurações 1 e 3). É possível observar também que o GPX2 possui maior vantagem em instâncias maiores, visto que a menor diferença de performance ocorre na instância E-n51-k5 apresentada na Tabela 2 e a maior ocorre na instância F-n135-k7, como pode ser visto na Tabela 5. Com a análise das demais tabelas, também é possível verificar que a diferença de performance aumenta de modo proporcional ao tamanho da instância.

Uma hipótese que pode explicar porque o uso do K-*means* gera mais impacto no OX do que no GPX2 é que o GPX2 é projetado para o PCV, enquanto o OX é um operador de propósito geral.

5. Conclusão

De acordo com os resultados, o operador GPX2 conseguiu alcançar um bom desempenho para a resolução do problema. Embora o operador em si não tenha alcançando estatisticamente uma grande variabilidade genética, a média das soluções se manteve próxima do ótimo global em quase todos os experimentos com GPX2.

Outra constatação importante é que, embora o operador mantenha certa consistência, é um operador guloso que rapidamente encontra ótimos locais, prejudicando a variabilidade genética. Uma forma de contornar esse problema seria adicionar outro método para melhorar a diversidade da população.

Em trabalhos futuros, a utilização de heurísticas de busca focadas no PRV poderia gerar soluções boas para otimização direta pelo operador, aproveitando suas qualidades. Além disso, a utilização do K-*means*, ou outro operador de clusterização, como uma forma de melhorar as soluções finais, poderia ser considerada.

Agradecimentos

Os autores agradecem cordialmente à UTFPR, FAPESP (auxílio - 2013/07375-0 e 2015/06462-1) e CNPq (auxílio 305755/2018-8) pelo apoio financeiro.

Referências

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5):512–522.

Geetha, S., Poonthalir, G., and Vanathi, P. (2009). Improved k-means algorithm for capacitated clustering problem. *INFOCOMP*, 8(4).

Nagata, Y. (2007). Edge assembly crossover for the capacitated vehicle routing problem. In *Proceedings of the 7th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'07, pages 142–153, Berlin, Heidelberg. Springer-Verlag.

Perez-Wohlfeil, E., Chicano, F., and Alba, E. (2018). An Intelligent Data Analysis of the Structure of NP Problems for Efficient Solution: The Vehicle Routing Case. pages 368–378.

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

Puljic, K. and Manger, R. (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications*, 18(2):359–375.

Rodrigues, P. R. A. (2008). *Introdução aos sistemas de transporte no Brasil e à logística internacional*. Edições Aduaneiras.

Sanches, D., Whitley, D., and Tinós, R. (2017). Building a better heuristic for the traveling salesman problem. In *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '17*, pages 329–336, New York, New York, USA. ACM Press.

Tinós, R., Whitley, D., and Ochoa, G. (2019). A New Generalized Partition Crossover for the Traveling Salesman Problem: Tunneling Between Local Optima. *Evolutionary Computation*, pages 1–31.

Tlili, T., Chicano, F., Krichen, S., and Alba, E. (2015). Evolutionary Algorithm Based on Partition Crossover (EAPX) for the Vehicle Routing Problem. In *2015 International Conference on Intelligent Networking and Collaborative Systems*, pages 169–175. IEEE.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., and Subramanian, A. (2017). New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*, 257(3):845–858.

Vigo, D. and Toth, P. (2014). Vehicle Routing; Problems, Methods, and Applications. *MOS-SIAM Series on Optimization*, page 467.

Whitley, D., Hains, D., and Howe, A. (2009). Tunneling between optima. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*, page 915, New York, New York, USA. ACM Press.