Algoritmo Genético para o Problema de Roteamento em Arcos Capacitado e Aberto

Rafael Kendy Arakaki
Instituto de Computacao - UNICAMP
Av. Albert Einstein 1251, CEP 13083-852, Campinas/SP - Brasil
rafael.arakaki@ic.unicamp.br

Fábio Luiz Usberti
Instituto de Computação - UNICAMP
Av. Albert Einstein 1251, CEP 13083-852, Campinas/SP - Brasil
fusberti@ic.unicamp.br

RESUMO

Problemas de roteamento em arcos têm por objetivo determinar rotas de custo mínimo que visitam um subconjunto de arcos de um grafo, com uma ou mais restrições adicionais. A solução desses problemas remete à diminuição de custos logísticos, melhorando a competitividade das empresas. O Problema de Roteamento em Arcos Capacitado e Aberto (*OCARP - Open Capacitated Arc Routing Problem*) é um problema de otimização combinatória NP-difícil com aplicações práticas, como o problema de roteamento de leituristas. Este trabalho propõe uma nova metodologia de algoritmos genéticos para tratar o OCARP. São apresentados experimentos computacionais com instâncias da literatura.

PALAVRAS CHAVE. roteamento em arcos, roteamento de leituristas, algoritmo genético, otimização combinatória, programação linear inteira.

Otimização Combinatória, Metaheurística, Programação Matemática

ABSTRACT

Arc routing problems aim at determining the lowest cost routes visiting a subset of edges from a graph, with one or more additional constraints. The solution of these problems leads to lower logistics costs, improving business competitiveness. The Open Capacitated Arc Routing Problem (OCARP) is an NP-hard combinatorial optimization problem with practical applications, such as the meter reading routing problem. This work proposes a new genetic algorithm methodology to handle the OCARP. It is presented a computational experiment using instances from the literature.

KEYWORDS. arc routing, meter reading routing, genetic algorithm, combinatorial optimization, integer linear programming.

Combinatorial Optimization, Metaheuristics, Mathematical Programming

1. Introdução

Os problemas de roteamento podem ser subdivididos em duas grandes áreas: os problemas de roteamento em nós e os problemas de roteamento em arcos (*ARPs - Arc Routing Problems*). Os problemas de roteamento em nós, como o clássico Problema do Caixeiro Viajante (*TSP - Travelling Salesman Problem*), objetivam a criação de rota(s) que visita(m) um subconjunto de nós. Por outro lado, os *ARPs* objetivam a criação de rota(s) que visita(m) um subconjunto de arestas. Durante a última década, os problemas de roteamento em arcos têm sido uma área de pesquisa bastante ativa [Wøhlk, 2008].

O problema de roteamento em arcos capacitado e aberto (*OCARP - Open Capacitated Arc Routing Problem*) é um problema de otimização combinatória definido em um grafo conexo não-direcionado com custos e demandas não-negativas nas arestas. É disponibilizada uma frota de veículos com capacidade limitada para atender as demandas das arestas. O objetivo do OCARP é encontrar um conjunto de rotas que atenda às demandas das arestas e possua custo mínimo [Usberti et al., 2011].

O OCARP é um problema NP-difícil, portanto se $P \neq NP$ não existe um algoritmo de tempo polinomial para resolvê-lo [Usberti, 2012]. Trata-se de um problema de otimização combinatória que possui diversas aplicações práticas.

Uma das aplicações práticas do OCARP é um problema real de roteamento de leituristas. Empresas distribuidoras de energia elétrica, água e gás possuem colaboradores que registram o consumo de clientes dispersos geograficamente. O roteamento desses funcionários apresenta-se como um problema de difícil solução, o qual consiste em determinar as rotas que os leituristas devem percorrer para realizar as leituras de consumo dos clientes [Usberti et al., 2008]. Outra aplicação do OCARP é o problema de determinação do caminho de corte, o qual consiste em guiar o corte de chapas metálicas em peças de formas poligonais utilizadas na indústria metal-mecânica [da Silva, 2016].

O presente trabalho propõe uma metodologia de solução heurística baseada em algoritmos genéticos para o OCARP. O método desenvolvido melhorou substancialmente as soluções conhecidas para várias instâncias da literatura.

Este trabalho está dividido em seis seções. A Seção 2 apresenta a definição e a formulação para o problema de roteamento em arcos capacitado e aberto. Adicionalmente, são revisados os principais resultados da literatura relacionada ao OCARP. A Seção 3 descreve a metaheurística de algoritmos genéticos proposta. A Seção 4 apresenta os resultados de experimentos computacionais da heurística. Por fim, a Seção 5 apresenta as conclusões.

2. Problema de Roteamento em Arcos Capacitado e Aberto

O Problema de Roteamento em Arcos Capacitado e Aberto (OCARP), proposto por Usberti et al. [2011], é definido a seguir. Seja $G(V, E)$ um grafo conexo não-direcionado com custos não-negativos c_{ij} e demandas não-negativas d_{ij} associados a cada aresta. Todas as arestas que possuem demandas positivas ($d_{ij} > 0$) formam o conjunto de arestas requeridas $E_R \subseteq E$. Cada aresta requerida deve ser atendida apenas uma vez por um único veículo. Entretanto, as arestas podem ser visitadas múltiplas vezes, por um ou mais veículos. Uma frota de M veículos idênticos com capacidade limitada D é disponibilizada. Ao percorrer o grafo, um veículo pode escolher entre atender uma aresta ou apenas visitá-la. No caso de atendimento a sua capacidade é reduzida pela demanda da aresta atendida.

O OCARP pode ser considerado uma variante do CARP (*Capacitated Arc Routing Problem - CARP*), porém com a diferença de que o OCARP considera tanto rotas abertas como fechadas. Uma rota aberta possui nós distintos como nó inicial e nó final, enquanto uma rota fechada possui o mesmo nó inicial e final. Uma solução OCARP consiste em um conjunto de M rotas, uma para cada veículo, para atender todas as arestas requeridas sem exceder a capacidade de nenhum veículo (Figura 1). O objetivo do OCARP é minimizar o custo desse conjunto de rotas.

Do ponto de vista teórico, OCARP e CARP são ambos problemas *NP-difíceis*, sendo possível a redução em tempo polinomial de um problema para o outro como mostrado por Usberti [2012]. Apesar das semelhanças entre CARP e OCARP, existem diferenças importantes que dificultam a adaptação de algoritmos do CARP para se resolver o OCARP. A maioria das metodologias para o CARP considera o número de veículos M como uma variável a ser otimizada, entretanto no OCARP o número de veículos M é um parâmetro fixo. Essa diferença é importante porque torna a geração de uma solução viável para o OCARP mais difícil, uma vez que se o número de veículos e a capacidade dos veículos forem baixos o suficiente, em relação aos valores das demandas, implica-se em um problema *NP-difícil* de empacotamento (*BPP - Bin-Packing Problem*) para se obter qualquer solução viável.

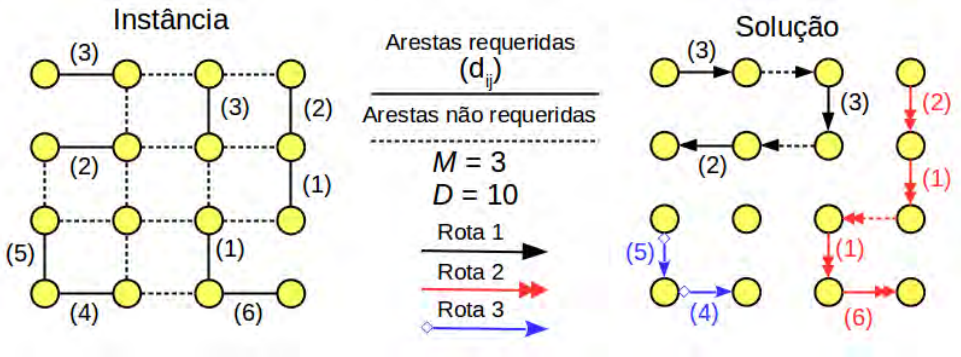

Figura 1: OCARP: uma instância e uma solução viável.

2.1. Modelo Matemático

Uma formulação para o OCARP proposta originalmente por Usberti et al. [2011] é apresentada a seguir.

Apesar do grafo G ser não-direcionado, é conveniente representar soluções OCARP com arcos direcionados para dar a orientação e ordem em que as arestas são visitadas. Desse modo são definidos quatro conjuntos de variáveis:

- $x_{ij}^k = 1$ se a rota k visita o arco (i, j), $x_{ij}^k = 0$ do contrário;
- $l_{ij}^k = 1$ se a rota k atende o arco (i, j), $l_{ij}^k = 0$ do contrário;
- $\alpha_i^k = 1$ se o vértice i é início da rota k, $\alpha_i^k = 0$ do contrário.
- u_S^k, v_S^k e w_S^k ($S \subseteq V$) são variáveis auxiliares utilizadas no conjunto de restrições (7) para eliminação de subciclos ilegais.

A função objetivo (1) minimiza o custo total, dado pela soma dos custos das arestas pertencentes a cada rota da solução. Seja $N(i)$ o conjunto de vértices vizinhos ao vértice i em G, a restrição (2) representa a continuidade das rotas e também impõe $\alpha_i^k = 1$ se o vértice i é início da rota k. A restrição (3) não permite que haja mais de um vértice como início de rota. A restrição (4) estabelece que arcos atendidos precisam ser visitados. A restrição (5) força que cada aresta requerida (representada por dois arcos) seja atendida uma única vez por uma única rota. A restrição (6) representa o limite de capacidade dos veículos.

O conjunto de restrições (7) não permite subciclos ilegais na solução. Para uma certa rota k, a rota é considerada ilegal se existe algum subconjunto de nós S tal que $v_S^k = u_S^k = w_S^k = 1$. A variável u_S^k identifica se há um ciclo em S, a variável v_S^k identifica se não há arestas de corte em $(S, \hat{S})$ e a variável w_S^k identifica se não há um vértice inicial em S. Um exemplo de subciclo ilegal

é apresentado na Figura 2. O conjunto de restrições (7) é de ordem exponencial, portanto na prática essas restrições são adicionadas ao modelo à medida em que são necessárias.

$(OCARP)$

$$MIN \quad \sum_{k=1}^{M} \sum_{(i,j)\in E} c_{ij} x_{ij}^k \tag{1}$$

s.a.

$$\sum_{j\in N(i)} x_{ij}^k - x_{ji}^k \leq \alpha_i^k \qquad (i \in V, k \in \{1,...,M\}) \tag{2}$$

$$\sum_{i\in V} \alpha_i^k \leq 1 \qquad (k \in \{1,...,M\}) \tag{3}$$

$$x_{ij}^k \geq l_{ij}^k \qquad ((i,j) \in E_R, k \in \{1,...,M\}) \tag{4}$$

$$\sum_{k=1}^{M} (l_{ij}^k + l_{ji}^k) = 1 \qquad ((i,j) \in E_R) \tag{5}$$

$$\sum_{(i,j)\in E_R} d_{ij} l_{ij}^k \leq D \qquad k \in \{1,...,M\}) \tag{6}$$

$$\left.\begin{aligned} \sum_{(i,j)\in(S,S)} x_{ij}^k - |S|^2 u_S^k \leq |S| - 1 \\ \sum_{(i,j)\in(S,\hat{S})} x_{ij}^k + v_S^k \geq 1 \\ \sum_{i\in S} \alpha_i^k + w_S^k \geq 1 \\ u_S^k + v_S^k + w_S^k \leq 2 \end{aligned}\right\} \qquad (S \subseteq V, \hat{S} = V \setminus S, k \in \{1,...,M\}) \tag{7}$$

$$x_{ij}^k \in \{0,1\}, \qquad ((i,j) \in E, k \in \{1,...,M\}) \tag{8}$$

$$l_{ij}^k \in \{0,1\}, \qquad ((i,j) \in E_R, k \in \{1,...,M\}) \tag{9}$$

$$\alpha_i^k \in \{0,1\}, \qquad (i \in V, k \in \{1,...,M\}) \tag{10}$$

$$u_S^k, v_S^k, w_S^k \in \{0,1\}, \qquad (k \in \{1,...,M\}, S \subseteq V) \tag{11}$$

Lema 2.1. *O conjunto de restrições (7) elimina soluções com rotas ilegais (subciclos ilegais) mas não elimina solução viáveis (rotas legais).*

Demonstração. Primeiro provaremos que o modelo proíbe uma rota ilegal. Suponha uma rota k ilegal e um grafo G' induzido pelas arestas que pertencem à rota k. Se a rota k é ilegal então o grafo G' não é conexo. Pela restrição (3) no máximo uma componente de G' terá um vértice de início de rota ($\alpha_i^k = 1$). Excluindo-se a componente que tem o vértice inicial, há pelo menos mais uma. Essa(s) outra(s) componente(s) de G' deverão formar subciclo(s) para manter o equilíbrio da restrição (2). Seja S' o conjunto de vértices de uma dessas componentes que não contém o vértice inicial. Então é possível verificar que:

- $u_{S'}^k = 1$, pois S' corresponde a um subciclo.
- $v_{S'}^k = 1$, porque S' é uma componente em G' (não há arestas de corte).
- $w_{S'}^k = 1$, não há nó inicial em S'.

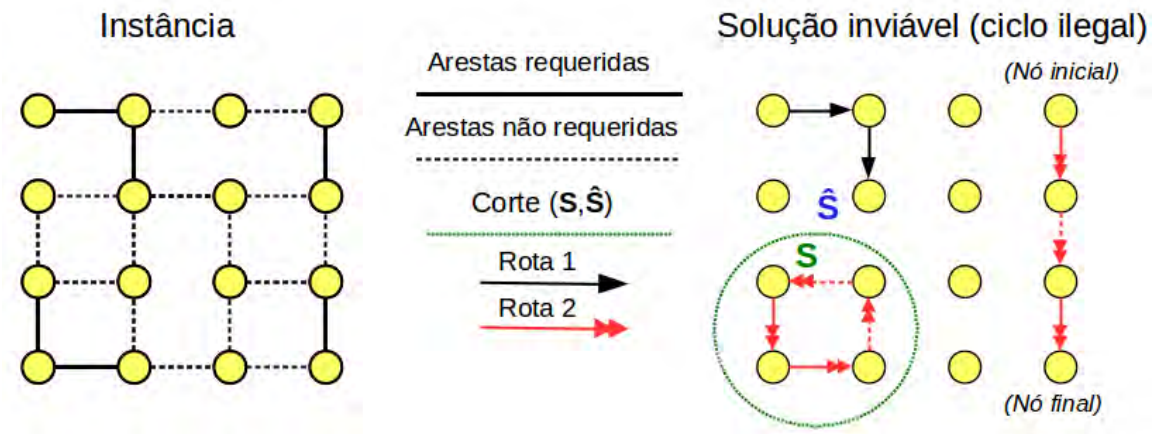

Figura 2: Uma instância OCARP (à esquerda) e uma solução inviável por causa do subciclo ilegal formado na rota 2 (à direita).

Temos portanto que qualquer rota k que gere mais de uma componente em um grafo G' induzido por suas arestas é ilegal no modelo (subciclo ilegal). Para completar a prova, mostraremos que o conjunto de restrições (7) não proíbe rotas legais, ou seja, rotas que gerem somente uma componente em um grafo G' induzido pelas suas arestas.

Suponha uma rota legal k e o conjunto R formado pelos vértices que são visitados por esta rota (vértices de G'). Temos então três casos possíveis para um subconjunto de vértices $S \subseteq V$ em relação a R:

1. Se $R \subseteq S$, um dos vértices da rota deve ser o inicial (em caso de rota fechada qualquer vértice pode ser inicial), portanto $w_S^k = 0$.

2. Se $R \not\subseteq S$ e $R \cap S \neq \varnothing$, então há pelo menos uma aresta de corte, que liga um elemento de $R \cap S$ com um elemento de $R \setminus S$, portanto $v_S^k = 0$.

3. Se $R \cap S = \varnothing$, não há arestas $(i, j) \in (S, S)$ na rota, portanto $u_S^k = 0$.

Como conclusão temos que o conjunto de restrições (7) proíbe exatamente as rotas que não são conexas (rotas ilegais) e em conjunto com as outras restrições do modelo define um conjunto de rotas factíveis para o OCARP. □

2.2. Revisão da Literatura

Os experimentos com a formulação original (Seção 2.1) reportaram soluções ótimas para várias instâncias do conjunto *ogdb* (7-27 nós, 11-55 arestas), porém não alcançou bom desempenho para os conjuntos de instâncias maiores: *oval* (24-50 nós, 34-97 arestas) e *oegl* (77-140 nós, 98-190 arestas). Segundo os autores a principal deficiência nesse método apresentou-se ser os limitantes duais fracos para as maiores instâncias [Usberti et al., 2011].

Mais tarde outro método de solução exata para o OCARP foi descrito por Usberti et al. [2012]. Trata-se de um algoritmo *branch-and-bound* que se vale da relação entre o OCARP e outro problema chamado *Capacity and Degree Constrained Minimum Spanning Forest* (CDCMSF). Esse método aprimorou os limitantes duais conhecidos, porém para o conjunto de instâncias mais difícil *oegl* ainda foi reportada uma média de desvio de otimalidade substancial (acima de 15%), evidenciando assim a necessidade por métodos mais eficientes.

3. Algoritmo Genético

Muitos problemas combinatórios NP-difíceis são resolvidos através de métodos heurísticos, uma vez que a solução exata desses problemas requer tempos exponenciais, exceto se $P = NP$ [Garey e Johnson, 1978]. Uma das metaheurísticas mais utilizadas para a solução de problemas de otimização são algoritmos genéticos [Goldberg, 1988].

Algoritmos genéticos são algoritmos inspirados por elementos da evolução das espécies como seleção, herança, mutação e cruzamento [Holland, 1991]. Nesta seção é descrita uma metaheurística de algoritmos genéticos proposta para a solução do OCARP.

3.1. Descrição da Metaheurística

A heurística apresentada nesta seção trata-se de um algoritmo genético híbrido, além das operações clássicas (cruzamento, seleção, população inicial) considera também os seguintes componentes: (i) rotina de factibilização, (ii) busca local, (iii) rotina de reinício de população. O Algoritmo 1 mostra o pseudo-código do algoritmo genético.

A rotina de factibilização é utilizada para se obter uma solução factível a partir da codificação de um indivíduo. A rotina de busca local é utilizada para aprimorar as soluções geradas ao longo do processo. Por fim, a rotina de reinício de população justifica-se para evitar a convergência prematura da população, uma vez que não é considerado nenhum operador de mutação.

Algorithm 1: Algoritmo Genético

```
início
    Inicialização da matriz de distâncias.
    Gerar a população inicial.
    enquanto limite de tempo não for excedido faça
        Selecionar pais para cruzamento.
        para um par de pais selecionados (pai1,pai2) faça
            c = cruzamento(pai1,pai2)
            s = factibilização(c)
            s' = busca_local(s)
            Adiciona a solução s' à população se factível.
        Selecionar nova população.
        Reinício de população a cada k gerações.
fim
```

3.2. Inicialização da Matriz de Distâncias

Essa etapa computa uma matriz de distâncias que é utilizada em diversas componentes da heurística. Seja E_R o conjunto de arestas requeridas e seja A_R o conjunto de arcos requeridos, onde cada aresta requerida $\{i, j\} \in E_R$ possui dois arcos requeridos correspondentes $(i, j), (j, i) \in A_R$. É gerada uma matriz D de dimensão $|A_R| \times |A_R|$. Cada entrada $D[e, f]$ dessa matriz é definida como o custo do caminho mínimo a partir do vértice fim do arco $e \in A_R$ até o vértice início do arco $f \in A_R$. Desse modo a distância entre arcos requeridos pode ser recuperada em tempo $O(1)$.

3.3. Codificação

O algoritmo genético utiliza como codificação de uma solução uma sequência de $|E_R|$ arcos requeridos. Desse modo, mantém-se na codificação a ordem em que os arcos requeridos são atendidos, sem a necessidade de explicitar as arestas não requeridas uma vez que estas estão inseridas implicitamente nos caminhos mínimos entre cada par de arcos requeridos.

A codificação não possui nenhum delimitador entre as rotas, de modo que a codificação pode ser vista como uma única rota não-capacitada, como mostra a Figura 3. A solução OCARP correspondente à codificação somente é obtida após a codificação passar pela etapa de factibilização (Seção 3.8).

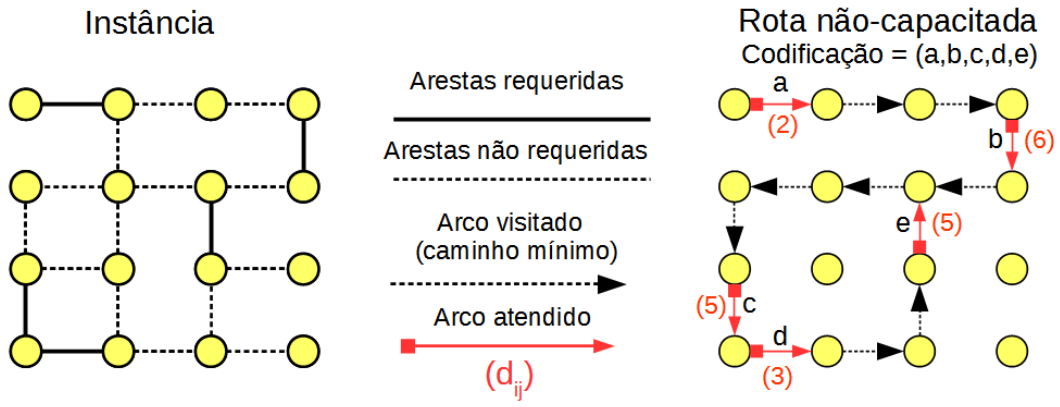

Figura 3: Uma instância e uma codificação com sua rota não-capacitada.

3.4. Fitness

A função de *fitness* de uma solução s de custo $c(s)$ é definida como $1/(c(s) - lb_0)$, onde lb_0 é um limitante dual trivial para o problema: a soma do custo de todas as arestas requeridas. Se for encontrada uma solução s onde $c(s) = lb_0$, então o algoritmo é encerrado pois foi encontrada uma solução ótima.

3.5. Geração da População Inicial

A inicialização da população é dividida em duas etapas. Na primeira etapa resolve-se o problema de roteamento não-capacitado e na segunda etapa procura-se particionar a rota não-capacitada em sub-rotas capacitadas, gerando assim soluções factíveis para o OCARP.

Na primeira etapa a rota não-capacitada é construída por via da heurística *Nearest Neighbour*, onde a rota é construída de maneira gulosa e iterativa até inserir todos os elementos requeridos. Por fim, a rota é otimizada pelo método de busca local 2-opt [Croes, 1957].

A segunda etapa é composta pela rotina de factibilização (Seção 3.8). Caso o indivíduo gerado seja infactível, outro indivíduo é gerado até que toda a população seja preenchida com indivíduos factíveis.

3.6. Seleção

São realizados $P - 1$ cruzamentos a cada geração, tal que P é o tamanho da população e cada cruzamento requer a seleção de dois indivíduos.

A seleção dos indivíduos é realizada através da técnica *Stochastic Universal Selection* de Baker [1986]. Nessa técnica, os indivíduos são selecionados em quantidade proporcional à razão de seu *fitness* sobre a soma de *fitness* de toda a população.

A nova geração é formada pelo melhor indivíduo e pelos $P - 1$ indivíduos filhos da geração corrente. Se houver C cruzamentos que geraram indivíduos filhos infactíveis, então são escolhidos aleatoriamente C indivíduos da geração atual para serem inseridos na nova geração.

3.7. Cruzamentos

Foi adotado o operador de cruzamento *C1* de Reeves [1996], onde um gene g é escolhido aleatoriamente e copia-se para o filho os genes do início até g do primeiro pai. Em seguida são copiados para o filho os genes que faltam na ordem em que aparecem no segundo pai (Figura 4).

3.8. Factibilização

O objetivo da rotina de factibilização é particionar a rota não-capacitada em M sub-rotas capacitadas (solução OCARP). Para isso é utilizado o algoritmo *Split* de Ulusoy [1985]. O algoritmo gera M sub-rotas que não excedem a capacidade dos veículos e que atendem os arcos requeridos na mesma ordem definida pela codificação (Figura 5).

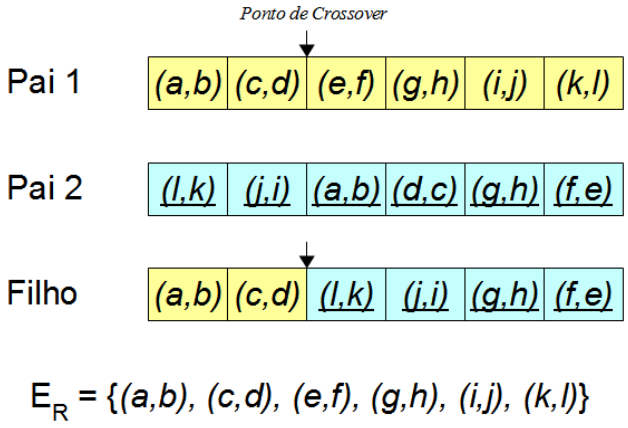

Figura 4: Operador de cruzamento $C1$.

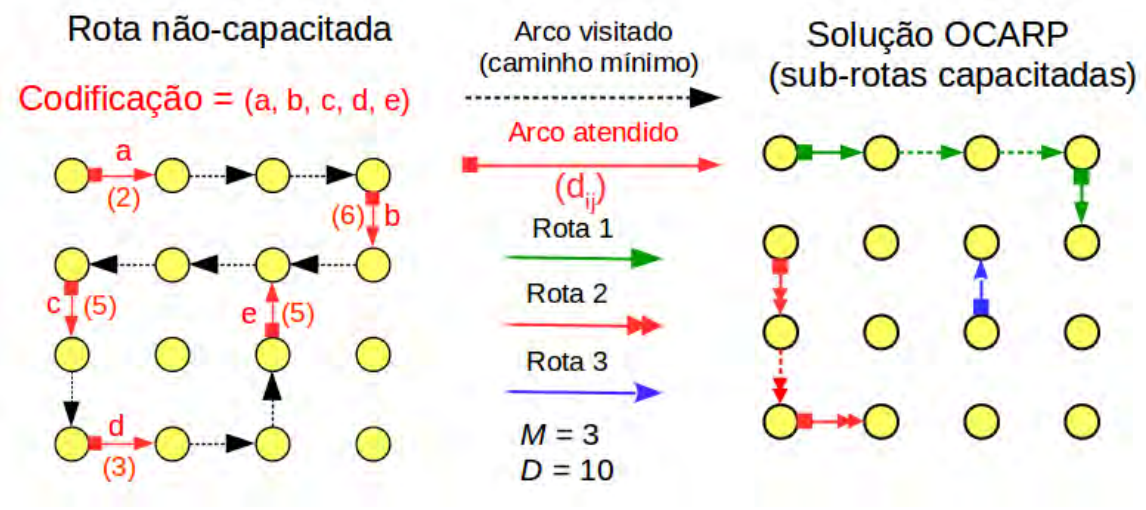

Figura 5: Uma codificação e uma solução correspondente dada pelo algoritmo *Split*.

Devido ao número fixo de veículos e das capacidades limitadas, não é possível garantir que a solução gerada pelo algoritmo de Ulusoy seja factível. Nesse caso é adotada uma estratégia para tentar tornar a solução factível. Dentre as rotas geradas pelo algoritmo, determina-se aquela com a menor capacidade residual adicionando-a a uma nova solução parcial. Em seguida, as arestas requeridas faltantes são adicionadas à solução parcial pelo método construtivo utilizado na inicialização da população (Seção 3.5). Finalmente, chama-se recursivamente a rotina de factibilização com essa nova solução. Se após M recursões não for encontrada uma solução factível, o indivíduo é declarado inviável.

3.9. Busca Local

Algoritmos de busca local realizam pequenas modificações em uma solução em busca de soluções similares (vizinhas) de melhor custo. As soluções geradas no cruzamento possuem uma probabilidade p de serem submetidas à rotina de busca local.

A busca local utiliza uma estratégia de desconstrução/reconstrução de pares de rotas. A reconstrução utiliza uma heurística construtiva gulosa *Nearest Neighbour* (Seção 3.5) para gerar uma única rota que atende os arcos do par de rotas desconstruído. Em seguida, essa rota única

é particionada em duas rotas pela rotina de factibilização (Seção 3.8). As alterações nas rotas da solução somente são efetivadas se diminuírem o custo da solução.

3.10. Reinício de População

A rotina de reinício de população é executada repetidamente ao longo da heurística em um intervalo de k gerações. Essa rotina tem o objetivo de evitar a convergência prematura da população.

A cada execução da rotina, metade da população atual é substituída por novas soluções. Os indivíduos substituídos são escolhidos aleatoriamente, entretanto a melhor solução nunca é substituída (seleção elitista). As novas soluções são geradas pelo mesmo processo utilizado pela rotina de inicialização da população.

4. Experimentos Computacionais

Esta seção apresenta o experimento computacional que tem por objetivo comparar os custos das soluções obtidos pelo algoritmo genético e pela heurística RPS_ER apresentada por Usberti et al. [2011].

A máquina utilizada no experimento compõe-se de processador Intel Xeon X3430 2.4 GHz, 8 GB de memória RAM e sistema operacional Linux 64-bit. O algoritmo genético foi executado uma única vez com limitação de uma hora de execução para cada instância. Os parâmetros definidos para o algoritmo genético são apresentados na Tabela 1.

Os resultados são apresentados nas Tabelas 2 e 3 para os respectivos conjuntos de instâncias: *oval* (24-50 nós, 34-97 arestas) e *oegl* (77-140 nós, 98-190 arestas) [Benavent et al., 1991; Li e Eglese, 1995]. Para cada instância, foram considerados três valores distintos para o parâmetro M (número de veículos): M^*, M^*+1 e M^*+2, onde M^* é o número mínimo de veículos necessários para se obter uma solução factível.

O valor de LP_1 corresponde à melhor solução obtida pela heurística RPS_ER (reportada pelos autores) e LP_2 à melhor solução obtida pelo algoritmo genético. Dados os valores de LP_1 e LP_2 é calculado um valor $\Delta LP(\%)$ que corresponde à diferença proporcional entre os dois limitantes. O tempo de processamento que o algoritmo genético levou para obter o limitante LP_2 é dado por CPU.

Tabela 1: Parâmetros do algoritmo genético

Parâmetro	Descrição	Valor
P	Tamanho da população	20
p	Probabilidade de busca local	0.1
k	Intervalo de reinício da população (gerações)	250

A partir das Tabelas 2 e 3 é possível observar que o algoritmo genético obteve resultados expressivamente melhores do que a metodologia da literatura. Pela Tabela 2 observa-se que o algoritmo genético se mostrou eficiente e robusto para encontrar boas soluções com pouco tempo de processamento. A Tabela 3 apresenta os resultados para as instâncias mais difíceis onde pode-se observar que o algoritmo genético obteve resultados expressivamente melhores do que os reportados na literatura, sendo esses, até onde sabemos, atualmente os melhores limitantes conhecidos para as instâncias em questão. Em muitas instâncias a diminuição do custo das soluções superou 15%, com a maior diminuição sendo de 44,45% na instância *egl-s4-C* que é a maior instância considerada nos experimentos.

Um fator que se mostrou de grande importância para a solução de uma instância OCARP é o valor do parâmetro M que limita o número de veículos disponíveis. Nas Tabelas 2 e 3 é possível verificar que nos casos onde $M = M^* + 1$ ou $M = M^* + 2$ a heurística tomou um tempo de processamento consideravelmente menor do que quando $M = M^*$.

O bom desempenho do algoritmo genético, em comparação ao algoritmo da literatura, pode ser atribuído à eficiência da rotina de factibilização que permite ao algoritmo encontrar soluções factíveis de bom custo mesmo se o número de veículos M é limitado.

Tabela 2: Resultados da heurística para as instâncias *oval*.

		*M**				*M*+1*				*M*+2*			
Instância	M^*	LP_1	LP_2	$\Delta LP(\%)$	CPU	LP_1	LP_2	$\Delta LP(\%)$	CPU	LP_1	LP_2	$\Delta LP(\%)$	CPU
val1A	2	154	154	0.00	0	154	151	**1.95**	1	154	149	**3.25**	1
val1B	3	154	151	**1.95**	1	149	149	0.00	1	149	147	**1.34**	1
val1C	8	173	148	**14.45**	0	149	146	**2.01**	0	146	146	0.00	0
val2A	2	195	195	0.00	0	195	192	**1.54**	1	195	189	**3.08**	1
val2B	3	192	192	0.00	0	192	189	**1.56**	1	192	186	**3.13**	0
val2C	8	197	185	**6.09**	0	186	185	**0.54**	0	185	185	0.00	0
val3A	2	71	71	0.00	0	71	69	**2.82**	1	71	67	**5.63**	1
val3B	3	69	69	0.00	0	68	67	**1.47**	1	68	66	**2.94**	1
val3C	7	68	66	**2.94**	0	65	65	0.00	0	65	65	0.00	0
val4A	3	361	358	**0.83**	4	365	354	**3.01**	4	365	350	**4.11**	4
val4B	4	363	354	**2.48**	6	363	350	**3.58**	3	363	347	**4.41**	2
val4C	5	364	350	**3.85**	457	355	347	**2.25**	2	355	345	**2.82**	2
val4D	9	359	343	**4.46**	326	354	343	**3.11**	1	359	343	**4.46**	1
val5A	3	387	383	**1.03**	3	383	378	**1.31**	4	383	374	**2.35**	3
val5B	4	385	378	**1.82**	3	386	374	**3.11**	2	386	371	**3.89**	2
val5C	5	381	374	**1.84**	2	378	371	**1.85**	2	378	368	**2.65**	1
val5D	9	377	367	**2.65**	1	378	367	**2.91**	1	378	367	**2.91**	0
val6A	3	196	195	**0.51**	1	196	193	**1.53**	2	196	192	**2.04**	1
val6B	4	197	194	**1.52**	1	194	192	**1.03**	1	194	191	**1.55**	1
val6C	10	195	190	**2.56**	0	192	190	**1.04**	0	192	190	**1.04**	0
val7A	3	267	263	**1.50**	3	267	259	**3.00**	4	267	256	**4.12**	3
val7B	4	259	259	0.00	2	262	256	**2.29**	3	262	253	**3.44**	2
val7C	9	262	250	**4.58**	2	261	249	**4.60**	1	256	249	**2.73**	0
val8A	3	366	364	**0.55**	16	366	359	**1.91**	3	366	354	**3.28**	3
val8B	4	364	359	**1.37**	3	360	354	**1.67**	2	360	351	**2.50**	2
val8C	9	361	347	**3.88**	11	353	347	**1.70**	0	355	347	**2.25**	0
val9A	3	303	298	**1.65**	16	303	294	**2.97**	11	303	292	**3.63**	9
val9B	4	304	294	**3.29**	10	301	292	**2.99**	7	301	290	**3.65**	6
val9C	5	304	292	**3.95**	6	297	290	**2.36**	5	297	288	**3.03**	4
val9D	10	300	282	**6.00**	152	294	281	**4.42**	19	294	280	**4.76**	1
val10A	3	409	402	**1.71**	12	409	396	**3.18**	13	409	391	**4.40**	11
val10B	4	406	396	**2.46**	11	406	391	**3.69**	9	406	388	**4.43**	7
val10C	5	406	391	**3.69**	7	404	388	**3.96**	6	404	385	**4.70**	5
val10D	10	398	379	**4.77**	10	396	378	**4.55**	10	396	377	**4.80**	2
média				**2.60**	31.36			**2.35**	3.38			**3.04**	2.23

LP_1: limitante primal obtido pela heurística RPS_ER de Usberti et al. [2011].
LP_2: limitante primal obtido pelo algoritmo genético.
$\Delta LP(\%) = 100(LP_1 - LP_2)/LP_1$: taxa de melhoria de LP_1 para LP_2.
CPU: tempo em segundos para o algoritmo genético obter LP_2.

Tabela 3: Resultados da heurística para as instâncias *oegl*.

		*M**				*M*+1*				*M*+2*			
Instância	M^*	LP_1	LP_2	$\Delta LP(\%)$	CPU	LP_1	LP_2	$\Delta LP(\%)$	CPU	LP_1	LP_2	$\Delta LP(\%)$	CPU
egl-e1-A	5	1959	1775	**9.39**	13	1813	1708	**5.79**	11	1813	1659	**8.49**	29
egl-e1-B	7	1961	1749	**10.81**	5	1809	1639	**9.40**	6	1818	1589	**12.60**	568
egl-e1-C	10	1833	1652	**9.87**	65	1709	1576	**7.78**	60	1750	1542	**11.89**	1
egl-e2-A	7	2450	2173	**11.31**	343	2352	2072	**11.90**	56	2352	2043	**13.14**	5
egl-e2-B	10	2423	2079	**14.20**	118	2283	1997	**12.53**	1	2275	1971	**13.36**	2
egl-e2-C	14	2700	2084	**22.81**	255	2344	1996	**14.85**	1685	2238	1964	**12.24**	107
egl-e3-A	8	3042	2526	**16.96**	301	2720	2410	**11.40**	364	2720	2366	**13.01**	381
egl-e3-B	12	2861	2409	**15.80**	563	2856	2351	**17.68**	14	2739	2321	**15.26**	158
egl-e3-C	17	3045	2357	**22.59**	925	2673	2286	**14.48**	1606	2720	2265	**16.73**	307
egl-e4-A	9	3270	2631	**19.54**	730	3118	2582	**17.19**	39	2988	2556	**14.46**	559
egl-e4-B	14	3293	2696	**18.13**	88	3119	2552	**18.18**	2986	3015	2517	**16.52**	328
egl-e4-C	19	4243	2812	**33.73**	3004	3183	2515	**20.99**	354	3003	2497	**16.85**	72
egl-s1-A	7	2075	1799	**13.30**	29	1946	1683	**13.51**	39	1946	1604	**17.57**	2
egl-s1-B	10	1963	1745	**11.11**	2	1878	1660	**11.61**	1	1828	1579	**13.62**	1173
egl-s1-C	14	2252	1874	**16.79**	1686	1963	1633	**16.81**	2	1849	1512	**18.23**	1
egl-s2-A	14	4383	3689	**15.83**	68	4249	3621	**14.78**	242	4159	3561	**14.38**	463
egl-s2-B	20	5442	3790	**30.36**	755	4243	3492	**17.70**	3107	4128	3427	**16.98**	276
egl-s2-C	27	5179	3732	**27.94**	1338	4310	3395	**21.23**	1486	4033	3336	**17.28**	1173
egl-s3-A	15	4428	3808	**14.00**	2330	4305	3761	**12.64**	2653	4356	3703	**14.99**	3229
egl-s3-B	22	4672	3670	**21.45**	2554	4354	3582	**17.73**	854	4179	3559	**14.84**	580
egl-s3-C	29	4828	3742	**22.49**	3233	4410	3564	**19.18**	2651	4265	3492	**18.12**	2613
egl-s4-A	19	5253	4476	**14.79**	3377	5168	4421	**14.45**	193	5086	4399	**13.51**	184
egl-s4-B	27	5653	4412	**21.95**	3488	5239	4333	**17.29**	1838	4981	4311	**13.45**	942
egl-s4-C	35	9032	5017	**44.45**	3100	5852	4379	**25.17**	1865	5237	4284	**18.20**	2750
média				**19.15**	1182			**15.17**	921			**14.82**	662

LP_1: limitante primal obtido pela heurística RPS_ER de Usberti et al. [2011].
LP_2: limitante primal obtido pelo algoritmo genético.
$\Delta LP(\%) = 100(LP_1 - LP_2)/LP_1$: taxa de melhoria de LP_1 para LP_2.
CPU: tempo em segundos para o algoritmo genético obter LP_2.

5. Conclusão

Este trabalho apresentou a definição e uma formulação de programação linear inteira para o OCARP. Foi proposto um algoritmo genético para tratar o OCARP. Os principais pontos-chaves do algoritmo apresentado foram:

- Codificação de uma solução ocorre através de uma representação não-capacitada.
- Algoritmo *Split* para decompor uma rota não-capacitada em sub-rotas capacitadas (solução).
- Rotina de Factibilização com estratégias para guiar o algoritmo *Split* e obter soluções viáveis.
- Busca Local de Desconstrução e Reconstrução para aprimorar as soluções.

Os resultados obtidos pelo algoritmo genético são favoráveis e mostram que o método conseguiu superar as metodologias existentes na literatura. Entretanto para se provar a otimalidade dos resultados obtidos pelo algoritmo genético ou pelo menos comprovar desvios de otimalidade baixos são necessários modelos matemáticos mais eficientes na prática, portanto esse pode ser um tópico de interesse para pesquisas futuras.

Referências

Baker, E. J. (1986). Reducing bias and inefficiency in the selection algorithm. *Proceedings of the second international conference on genetic algorithms*, p. 14–21.

Benavent, E., Campos, V., Corberán, A., e Mota, E. (1991). The capacitated arc routing problem: lower bounds. *Networks*, 22:669–690. ISSN 1097-0037.

Croes, A. G. (1957). A method for solving traveling-salesman problems. *Operations research*, 6: 791–812. ISSN 0030-364X.

da Silva, E. F. (2016). Modelos matemáticos para um problema de caminho de corte. Master's thesis, Universidade de São Paulo.

Garey, R. M. e Johnson, S. D. (1978). Computers and intractability: a guide to the theory of np-completeness. 1979. *San Francisco, LA: Freeman*.

Goldberg, D. (1988). Genetic algorithms in optimization, search and machine learning. *Addison Wesley*, 905:205–211.

Holland, H. J. (1991). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press. ISBN 0262581116.

Li, Y. L. e Eglese, W. R. (1995). An interactive algorithm for vehicle routeing for winter-gritting. *Journal of the Operational Research Society*, p. 217–228. ISSN 0160-5682.

Reeves, R. C. (1996). Feature article-genetic algorithms for the operations researcher. *INFORMS journal on computing*, 9:231–250. ISSN 1091-9856.

Ulusoy, G. (1985). The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337.

Usberti, F. L., França, P. M., e França, A. L. M. (2008). *Roteamento de Leituristas: Um Problema NP-Difícil. Em: XL SBPO Brazilian Symposium of Operational Research*. Annals XL SBPO, João Pessoa.

Usberti, F. L. (2012). *Métodos heurísticos e exatos para o problema de roteamento em arcos capacitado e aberto*. PhD thesis, Universidade Estadual de Campinas.

Usberti, F. L., França, P. M., e França, A. L. M. (2012). *Branch-and-bound algorithm for an arc routing problem*. Annals XLIV SBPO, Rio de Janeiro.

Usberti, F. L., França, P. M., e França, A. L. M. (2011). The open capacitated arc routing problem. *Computers and Operations Research*, 38(11):1543 – 1555. ISSN 0305-0548.

Wøhlk, S. (2008). A decade of capacitated arc routing. In *The vehicle routing problem: latest advances and new challenges*, p. 29–48. Springer.