XI Congreso Galego de Estatística e Investigación de Operacións
A Coruña, 24-25-26 de outubro de 2013

COMPARAÇÃO DE DOIS ALGORITMOS GENÉTICOS APLICADOS AO TOP

José A. Oliveira[1], João Ferreira[1], Manuel Figueiredo[1], Luis Dias[1] e Guilherme Pereira

[1] Centro Algoritmi - Universidade do Minho, Guimarães, Portugal

RESUMO

A recolha seletiva de resíduos sólidos urbanos para reciclagem é um processo dispendioso, especialmente quando realizado em grande escala. Um problema importante neste processo reside na gestão de uma frota, uma vez que atualmente as estratégias utilizadas geralmente têm baixa eficiência.

O processo de recolha seletiva de resíduos sólidos urbanos pode ser modelado como um problema de encaminhamento de veículos, em particular como um Problema de Orientação de Equipas (TOP - *Team Orienteering Problem*). No TOP uma frota de veículos é incumbida de visitar um conjunto selecionado de vértices, de modo a maximizar o luvro total. O objetivo deste trabalho é o de otimizar o processo de recolha selectiva de resíduos sólidos urbanos ao abordar as questões relacionadas com a gestão de uma frota. Isso deve ser alcançado através do desenvolvimento de uma ferramenta de software que implementa um algoritmo genético para resolver o modelo desenvolvido.

Neste artigo apresentamos e comparamos dois algoritmos genéticos através de experiências computacionais realizadas com instâncias de teste conhecidas da literatura. O uso de algoritmos genéticos para resolver o TOP mostra ser uma escolha acertada, pois o método é eficiente produzindo bons resultados num tempo aceitável.

Palabras e frases chave: Recolha Seletiva de Resíduos Sólidos Urbanos, Logística Inversa, Problemas de Roteamento de Veículos, Team Orienteering Problem, Algoritmos Genéticos, Heurísticas.

1. INTRODUCIÓN

Na última década, a importância do ato de reciclar tem aumentado consideravelmente, e a reciclagem tornou-se uma questão de educação, a fim de perspectivar um futuro melhor e mais limpo. Como o processo de reciclagem se tornou realmente importante, também se transformou num problema de gestão de recursos interessante, em particular quando se refere à recolha seletiva de resíduos sólidos urbanos, que envolve equipas de pessoas e veículos. O planeamento do processo de recolha é fundamental para que a reciclagem possa ser eficiente, e por isso, a gestão da frota pode, muitas vezes lidar com várias questões. O problema principal reside na determinação de rotas ótimas de recolha, onde um conjunto de pontos de recolha é selecionado, e a cada ponto é dado um nível de prioridade. Este problema pode ser descrito como um problema da área de roteamento, e exige ainda mais flexibilidade quando se trata de escolher apenas uma parte dos pontos de recolha a serem visitados, em vez de todo o conjunto, como acontece no problema de roteamento de veículos (VRP – *Vehicle Routing Problem*). Uma modelação mais adequada do processo de recolha seletiva de resíduos sólidos urbanos pode ser através do Problema de Orientação de Equipas (TOP - *Team Orienteering Problem*). Neste contexto, o TOP pode ser descrito como o problema de projetar rotas de recolha optimizadas para ser atribuídas a uma frota de veículos que realizam a recolha de diferentes tipos de resíduos armazenados ao longo de uma rede de pontos de recolha. Cada um destes pontos de recolha contém uma certa quantidade de resíduos que quantifica diretamente o respectivo nível de prioridade. As rotas de recolha tem duração máxima ou distância máxima e, consequentemente, a seleção dos pontos de recolha a serem visitados pelos veículos é feita através do equilíbrio entre as suas prioridades e suas contribuições para a duração ou para a distância da rota. O objetivo é maximizar a quantidade total de resíduos recolhidos por todas as rotas, respeitando as limitações de tempo ou distância.

O TOP foi apresentado pela primeira vez por Butt and Cavalier (1994) sob o nome de *Multiple Tour Maximum Collection Problem*. Posteriormente, Chao et al. (1996) introduziu formalmente o problema e concebeu um dos conjuntos mais utilizados de instâncias de teste. A maioria dos melhores resultados conhecidos atualmente para essas instâncias foram obtidos por Archetti et al. (2007), com duas versões de Pesquisa Tabu (TS – *Tabu Search*), juntamente com duas metaheurísticas baseadas na Pesquisa de Vizinhança Variável (VNS - *Variable Neighbourhood Search*). Outras abordagens competitivas foram realizadas por Ke et al. (2008), com quatro variantes de Otimização por Colónia de Formigas (ACO - *Ant Colony Optimization*); Vansteenwegen et al. (2009), com uma heurística baseada em VNS. Mais recentemente, Souffriau et al (2010) projetou duas variantes híbridas do procedimento de pesquisa local com reinícios (GRASP - *Greedy Randomized Adaptive Search Procedure*) com *Path Relinking*. Também em 2010, Bouly et al. apresentou um algoritmo memético (*memetic algorithm*). Mais tarde, em 2012, Sevkli e Sevilgen desenvolveu um algoritmo baseado em núvem de partículas (PSO - *Particle Swarm Optimization*). A aplicação de algoritmos evolutivos ao *Orienteering Problem* (OP) foi iniciada por Tasgetiren (2002).

O trabalho apresentado neste artigo é parte de uma série de experiências integradas no projecto de I&D GATOP - *Genetic Algorithm for Team Orienteering Problem*, que foi aprovado para financiamento pela Fundação para a Ciência e a Tecnologia (FCT) do Governo de Portugal. Trata-se de um projeto composto por cinco tarefas combinadas para atingir o objetivo desejado, que é o desenvolvimento de uma metodologia mais completa e eficiente para vários problemas multi-nível de roteamento de veículos do mundo real, com ênfase na gestão da Recolha Seletiva de Resíduos Sólidos Urbanos (RSRSU). Dentro do projeto GATOP, a principal tarefa é resolver o TOP, e é sugerido o desenvolvimento de soluções heurísticas baseadas em algoritmos genéticos (GA – *Genetic Algorithm*). A simplicidade de um GA na modelagem de problemas complexos e a sua fácil integração com outros métodos de otimização foram os fatores considerados para a sua escolha.

Neste trabalho apresentamos a resolução de instâncias de média-grande escala, considerando-se apenas uma restrição de tempo na construção das rotas. Iremos comparar o desempenho de dois métodos baseados em GA desenvolvidos, comparando os seus resultados num conjunto de instâncias de teste, com os resultados apresentados por outros autores em estudos anteriores. Este artigo está dividido em cinco secções. Depois da introdução ao tema na primeira secção, na secção seguinte discute-se o modelo matemático. A terceira secção é a mais extensa e está dedicada à apresentação detalhada dois dois algoritmos genéticos. Os resultados computacionais são apresentados e discutidos na quarta secção. Na última secção apontam-se algumas conclusões.

2. FORMULAÇÃO MATEMÁTICA

O objectivo do presente estudo é resolver o TOP, o que significa o desenvolvimento de um método que determine *P* caminhos (simples) disjuntos que se iniciam no mesmo local e têm o mesmo destino, de forma a maximizar o lucro global obtido pela contribuição de cada caminho, respeitando a restrição de tempo em cada caminho. Os caminhos, além de serem caminhos disjuntos (não partilham nenhum arco), não partilham nenhum vértice, para além do vértice inicial e vértice final que são os mesmos para cada caminho. Em seguida, os caminhos gerados são atribuídos a uma frota limitada de veículos, geralmente um caminho para cada veículo disponível.

Nós seguimos de perto a formulação matemática apresentada no trabalho de Ke et al. (2008), e, assim, a função de objectivo para o TOP é dado na equação (1), onde *n* é o número total de vértices de um grafo, *m* é o número de veículos disponíveis, o valor *y* indica se o vértice *i* é visitado ou não por um veículo *k* e, finalmente, *r* é o prémio associado ao vértice *i*. A função objetivo pretende maximizar a recompensa total ou lucro, através da definição de *m* rotas viáveis. No caso particular do TOP aplicado em RSRSU, a função objetivo do modelo corresponde à soma dos resíduos recolhidos em cada ponto de recolha visitado pelas rotas estabelecidas.

$$\max \sum_{i=2}^{n-1} \sum_{k=1}^{m} r_i \cdot y_{ik} \tag{1}$$

Foram realizadas várias experiências computacionais. O modelo matemático foi implementado na linguagem AMPL e utilizando NEOS Server e AMPL / Gurobi / XpressMP (Czyzyk et al. 1998), XLI_MathProg v4.7 / LP_Solve e OPL / Cplex. O modelo proposto por Ke et al. (2008) permite a solução de instâncias de pequena dimensão, limitadas normalmente a menos de duas dezenas de vértices. No NEOS Server não foi possível obter-se uma solução para as instâncias de 50 vértices de Chao et al. (1996) que foram utilizadas nesta pesquisa. Ao fim de 3 horas de utilização do AMPL / Gurobi os processos são terminados sem indicação de qualquer solução incumbente. Usando o AMPL / XpressMP no NEOS Server consegue-se por vezes obter uma solução incumbente, mas que apresenta um valor elevado de GAP, ao fim de 3 horas. Outros casos de insucesso reportados pelo NEOS Server reportam falta de memória, e as execussões são interrompidas.

Perante estes resultados preleminares, abandonou-se a possibilidade de desenvolvimento de métodos de solução exata mais complexos e eficazes, optando-se pelas abordagens heurísticas, dado o alinhamento com os estudos do VRP, onde também é prática corrente o uso deste tipo de técnicas. As abordagens heurísticas permitem a obtenção de soluções aproximadas em tempo útil, que é uma questão importante dada a natureza da urgência de obtenção de soluções neste problema real de nível operacional. As decisões para este problema real tomam-se dia a dia, e por vezes mais do que uma vez no mesmo dia, dado a necessidade de serem feitos ajustes aos planos iniciais.

3. AS FERRAMENTAS DESENVOLVIDAS

O algoritmo genético (GA) é uma heurística de pesquisa que imita o processo natural da evolução, uma vez que se acredita acontecer com todas as espécies de seres vivos. Este método utiliza técnicas inspirados na natureza, tais como mutação, cruzamento, herança e seleção, para gerar soluções para problemas de otimização. O sucesso de um GA depende do tipo e da complexidade do problema ao qual ele é aplicado, embora, no nosso entender, seja o método mais apto a ser aplicado em problemas onde se tem pouco conhecimento específico do problema. O conhecimento específico de um problema é necessário para se desenvolver e aplicar estruturas de vizinhanças mais adequadas em métdos baseados em pesquisa local, como acontece na maioria dos métodos heurísticos, e em particular nas meta-heurísticas de solução única, nomeadamente Pesquisa Tabu (TS - *Tabu Search*), Pesquisa por Arrefecimento Simulado (SA - *Simulated Annealing*), Pesquisa em Vizinhança Variável (VNS - *Variable Neighborhood Search*), entre outros.

O conhecimento específico de um problema é também importante e útil no caso dos algoritmos genéticos, pois permite o desenvolvimento de operadores genéticos (cruzamento e mutação) mais eficazes e eficientes para esse problema. Porém, mesmo com operadores genéticos, e independentemente do problema, um GA é capaz de "otimizar", isto é, consegue melhorar as soluções iniciais e fornecer soluções de boa qualidade. Assim, para uma primeira aproximação a um "novo" problema, ou a um problema para o qual a equipa de investigadores não possui ainda muito conhecimento específico do problema, o algoritmo genético é sempre uma boa opção de escolha. Pelo facto de o GA ser um método populacional, isto é, um método de pesquisa que em cada iteração lida com um conjunto de soluções (a população), facilita a cobertura do espaço de soluções e a obtenção de soluções de boa qualidade em problemas de otimização combinatória. Por seu lado, os métodos de solução única lidam em cada iteração somente com uma solução corrente e com a vizinhança dessa solução, pelo que a cobertura eficaz do espaço de soluções depende fortemente do conhecimento específico que se tem do problema.

Num GA os cromossomas ou indivíduos são representados como cadeias de caratéres que codificam soluções candidatas para um problema de otimização, que vão evoluindo para melhores soluções. No TOP, um cromossoma pode ser composto por n genes, ou seja, um para cada vértice, embora possa haver outras configurações com mais ou menos genes. O gene i do cromossoma corresponde ao vértice i no grafo. O alelo (valor ou informação contida no gene) identifica a prioridade do vértice. A concepção de um GA requer uma representação genética do domínio das soluções, bem como uma função de aptidão para avaliar a qualidade das soluções produzidas. No que diz respeito ao problema da recolha seletiva de resíduos urbanos, a função de aptidão dos algoritmos genéticos desenvolvidos corresponde à soma das quantidades de resíduos recolhidos nas rotas, isto é, corresponde à soma das quantidades de resíduos recolhidos em cada ponto de recolha que foi visitado por um veículo.

O processo evolutivo GA começa por inicializar uma população de soluções (normalmente aleatória), que vai evoluir e melhorar durante três passos principais:

- Seleção: sucessivamente, uma parte de cada população é selecionada com base na sua aptidão, a fim de produzir uma nova população e, provavelmente, melhor adaptada ao seu meio ambiente (problema).
- Reprodução: as soluções selecionadas no passo anterior produzem a próxima geração através de operadores genéticos: cruzamento e / ou mutação, propagando por herança as mudanças mais importantes para as futuras gerações.
- Término: uma vez encontrado o critério de paragem, o processo geracional termina.

A operação de cruzamento envolve a combinação de dois cromossomas, a fim de se obter um cromossoma melhor por herança dos bons (melhores) genes de cada progenitor. Por seu lado, a mutação é o processo onde uma solução válida pode alterar o seu próprio código genético, modificando alguns (poucos) dos seus genes.

Nas ferramentas desenvolvidas e apresentadas neste trabalho, ao aplicar os conceitos de um algoritmo genético para modelar um TOP dedicado ao problema de RSRSU, assumiu-se que:

- Há um conjunto de *n* pontos de recolha que devem ser visitados, no máximo, uma vez e por um único veículo;
- Cada ponto de recolha é representado por um vértice num grafo;
- A frota de *m* veículos está disponível num depósito;
- O depósito é representado por um vértice num grafo;
- Um cromossoma codifica uma solução possível, a qual corresponde a um conjunto de *m* rotas;
- Um cromossoma é formado por *m* sub-cromossomas, cada um codifica uma rota;
- O comprimento de um sub-cromossoma (número de genes) podem variar, ou pode ser um valor fixo;
- Cada gene representa um ponto de recolha e está associado ao vértice respectivo do grafo;
- O gene *i* de um sub-cromossoma corresponde ao *i-ésimo* ponto de recolha a ser visitado;
- Um gene tem associado um alelo que corresponde à prioridade do ponto de recolha em ser visitado.

Também está implícita a utilização de um procedimento construtivo para interpretar / descodificar o cromossoma, isto é, a associação de um cromossoma (simples cadeia de caracteres) a uma solução para o TOP é feita por um algoritmo construtivo, que constroi a solução pela leitura dos alelos presentes nos genes do cromossoma. Este procedimento pode ser realizado de várias maneiras e com diferentes técnicas, mas, essencialmente, consiste na adição de vértices (pontos de recolha) para as rotas de tal maneira que, quando um vértice *v* é adicionado: 1) o vértice *v* não pertencem a uma outra rota dentro da solução atual, 2) é possível ir do último vértice adicionado ao vértice *v* e de *v* diretamente para o vértice final, sem violar o limite de tempo.

3.1 GATOP-1

Nesta abordagem, o algoritmo começa com um procedimento construtivo e a cada veículo é dada uma lista de prioridades em termos de pontos de recolha (vértices) a visitar, gerada aleatoriamente, que representa a sequência desejável de visita dos pontos de recolha. A Figura 1 apresenta duas lista de prioridades para um exemplo com 6 vértices e duas viaturas. Cada um dos sub-cromossoma corresponde a uma lista de prioridades diferente de pontos de recolha.

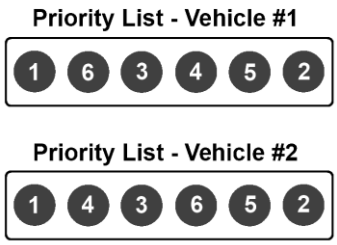


Figura 1: Listas prioritárias para dois veículos numa instância com seis vértices.

Uma representação gráfica de uma solução viável para uma instância com seis vértices e dois veículos é apresentado na Figura 2. As duas rotas são construídas considerando as listas de prioridades da Figura 1: Vehicle #1 (linha sólida) e Vehicle #2 (linha a tracejado).

- Rota Vehicle #1 - A rota contém os vértices 1-6-3-4, e esses vértices tornam-se automaticamente indisponíveis para o veículo 2. Os vértices 5 e 2 não estão incluídos no percurso, porque nesse caso, o limite de tempo seria excedido.
- Rota Vehicle #2 - Quatro vértices da lista de prioridades do veículo #2 já foram consideradosna rota do veículo #1. O próximo vértice disponível é o vértice 5, e uma vez adicionado ao percurso Vehicle #2, a solução fica completa, uma vez que a adição do vértice 2 violaria a restrição de tempo.

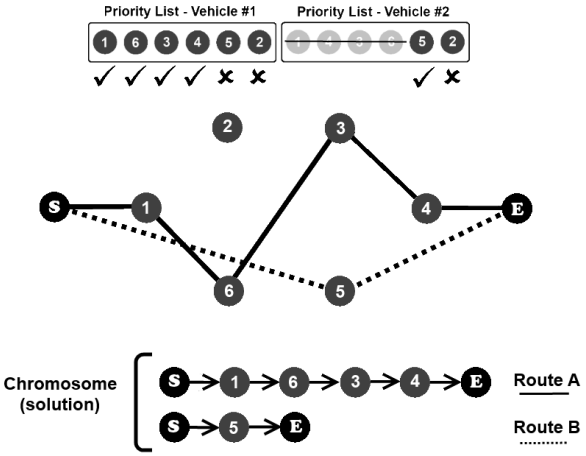


Figura 2: Exemplo de solução para uma instância com seis vértices e dois veículos.

O GA implementado segue uma estrutura de funcionamento muito simples e comum. Apresenta-se de seguida o pseudocódigo do algoritmo GATOP–1.

Algoritmo GATOP-1

```
Begin
P ← InitializePopulation (population_size)
While stopping condition not met do:
      I ← Immune_Selection (P)
      ΔP ← Random_Generation (n)
      M ← Mutation_Selection (P)
      M ← Mutation (M)
      C ← Crossover_Selection (P)
      O ← Crossover (C)
      P ← O + M + ΔP + I
End
```

Onde P é a população de soluções, I é um subconjunto de P, que inclui as soluções de elite (os elementos com maior aptidão), que permanecem imunes ao processo de mutação; ΔP representa uma parte residual de P, que inclui n soluções geradas aleatoriamente, utilizadas para promover a diversificação no processo reprodutivo, e para evitar o início de convergência para soluções ótimos locais; M designa o grupo de soluções seleccionadas de P que irão ser mutadas; C é o subconjunto de soluções de P seleccionadas para produzir a descendência O por meio de cruzamento.

Conceptualmente espera-se que na operação de cruzamento, dois cromossomas (soluções) de boa qualidade sejam combinados a fim de se obter um cromossoma (solução) de melhor qualidade que os seus progenitores, porque herdará os bons (melhores) genes de cada progenitor. Para realizar este procedimento no GATOP-1, um bloco de genes consecutivos é copiado de uma solução progenitora e é colocada numa posição aleatória no interior da outra solução progenitora. O bloco de genes tem um

tamanho aleatório, e a posição de inserção poderá variar aleatoriamente. Depois da inserção, os vértices que aparecem duas vezes na solução descendente são, em seguida, removidos. O esquema da Figura 3 mostra como o processo de cruzamento é feito. Neste exemplo, a passagem ocorre entre dois cromossomas, *A* e *B*, com dois sub-cromossomas (rotas) cada. Em primeiro lugar, um conjunto de genes consecutivos é copiado de *B*. Em seguida, o mesmo bloco é inserido dentro de um sub-cromossoma escolhido aleatoriamente e numa posição arbitrária de *A*. A remoção dos vértices repetidos inicia-se na posição em que o sub-cromossoma de *A* foi modificado, e assim impedindo a remoção dos vértices recentemente adicionados. A nova solução gerada está então pronta para lutar por uma vaga na próxima geração.

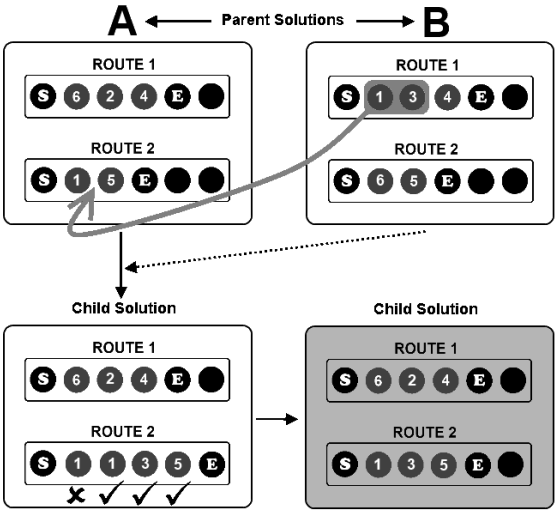


Figura 3: O procedimento de cruzamento.

No outro método de reprodução, mutação, uma solução válida pode alterar o seu próprio código genético, modificando de forma aleatória a ordem dos genes. Neste caso, decidiu-se realizar uma troca simples de posição, primeiro selecionando um vértice (ou gene) de uma das listas de prioridades e depois trocar com o vértice imediatamente a seguir, na mesma lista. Um exemplo que mostra como o processo funciona pode ser observado na Figura 4.

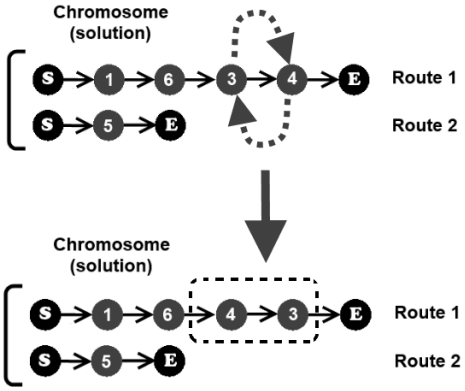


Figura 4: O processo de mutação.

Os indivíduos são seleccionados para ser reproduzidos por cruzamento e mutação de acordo com uma distribuição triangular, em que quanto maior for a aptidão de um cromossoma, maior é a probabilidade que tem de tomar parte no processo de reprodução. Neste algoritmo, existem parâmetros que definem o número de mutações e crossovers que podem ocorrer durante cada fase de reprodução (ciclo de otimização). Outros parâmetros da ferramenta GATOP-1 permitem ajustar o número de cromossomas imunes, bem como a alteração do número de cromossomas gerados aleatoriamente em cada nova iteração. Também é possível ajustar o algoritmo para aceitar apenas cruzamentos e / ou mutações que melhoram o grupo de soluções. No caso de apenas aceitar cruzamentos melhores, o descendente de

dois cromossomas é medido em termos de aptidão, e se ele é mais apto do que um de seus progenitores, então o novo cromossoma é mantido, caso contrário, ele é descartado. Todos os parâmetros anteriormente mencionados vão determinar a eficiência global do algoritmo.

Uma aplicação de software JAVA foi desenvolvida para implementar o algoritmo descrito e designou-se por GATOP-1. A aplicação tem uma simples, mas funcional, Interface Gráfica do Utilizador (GUI - *Graphical User Interface*), com um conjunto de opções que permite que os parâmetros do GA possam ser ajustados e ajudar a obter melhores resultados num problema ou instância específica do TOP. Embora cientes disso, decidiu-se manter as mesmas configurações durante todos os testes computacionais para que uma comparação com outros autores da revisão bibliográfica pudesse ser feita, bem como uma avaliação global do nosso GA. O software desenvolvido pode carregar diferentes instâncias e é possível definir a condição de paragem. Os outros parâmetros descritos anteriormente também podem ser alterados através do menu da aplicação. Há também um elemento de visualização (*Solution Viewer*), que ilustra o exemplo carregado e apresenta as rotas da melhor solução encontrada até ao momento, que são atualizadas em tempo real. Na Figura 5 apresenta-se o "*Solution Viewer*", onde os círculos correspondem aos vértices no TOP. Os vértices apresentam um diâmetro proporcional ao seu lucro.

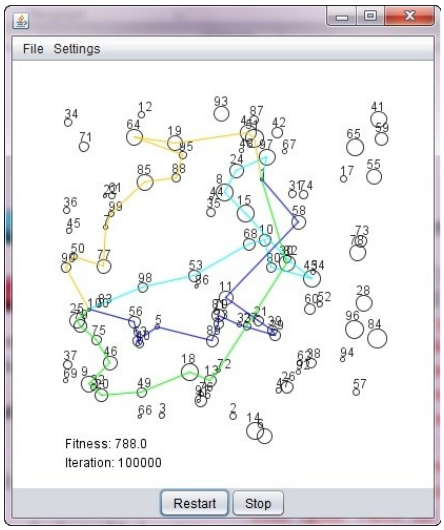


Figura 5: A "*Solution Viewer*" representa as rotas enquanto a aplicação resolve a instância do TOP.

3.2 GATOP-2

O algoritmo desenvolvido para a ferramenta GATOP-2 difere da abordagem anteriormente apresentada (GATOP-1), principalmente no processo construtivo das soluções e nos operadores genéticos de cruzamento e mutação. Existem algumas diferenças menores em relação à hierarquia de cromossomas na população, uma vez que dois grupos de elite são considerados no GATOP-2 em vez de apenas um grupo imunitário utilizado no GATOP-1.

No GA desenvolvido para o GATOP-2, uma solução válida tem de conter uma rota para cada veículo disponível, e cada percurso inclui uma sequência de vértices a serem visitados num determinado limite de tempo. Um exemplo de uma solução válida é mostrada na Figura 6. Há seis vértices numerados e estão disponíveis dois veículos. Existe também um ponto de partida (S) e um ponto de chegada (E). Neste caso, o vértice 6 não está incluído em nenhuma rota, pois, caso contrário, ele iria tornar a solução inválida por violar a restrição de tempo.

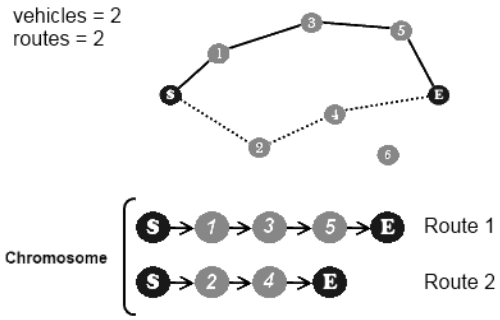


Figura 6: Exemplo de uma solução válida (seis vértices e dois veículos).

A fim de produzir soluções para o TOP, o algoritmo começa por gerar uma população inicial de cromossomas válidos, ou seja, composta de rotas válidas. Enm seguida, os operadores genéticos são aplicados à população, a fim de promover a sua evolução para melhores níveis de aptidão. Este processo é repetido até que um critério de paragem seja encontrado. A fim de se criar um percurso válido, os vértices são adicionados a partir de uma lista de disponibilidade, que é comum a todas as rotas de um determinado cromossoma. A tentativa de adicionar um vértice numa rota só é bem sucedida se a adição mantiver esse caminho viável, não excedendo o tempo disponível. Uma vez adicionada a um percurso, o vértice é removido da lista e assinalado como visitado. Cada um dos vértices na lista é testado para uma inserção na rota atual. Um cromossoma deve conter tantas rotas como o número de veículos disponíveis. Na Figura 7 mostra-se um esquema simples para uma instância do TOP do modo como um percurso é criado. O algoritmo usa processamento semi-paralelo para a criação das rotas, isto é, quando um vértice não pode ser adicionado numa rota , tenta-se a sua adição na rota seguinte, até se esgotar todas as posibilidades. Deste modo as rotas vão sendo construídas em paralelo, mas continua a haver uma predominância na construção das primeiras rotas, que se "servem" em primeiro lugar dos vértices disponíveis. Ao invés, o algoritmo do GATOP-1 conclui sempre a primeira rota antes de atribuir vértices à segunda rota, que por sua vez é "fechada" antes de iniciar a seguinte, e assim sucesivamente.

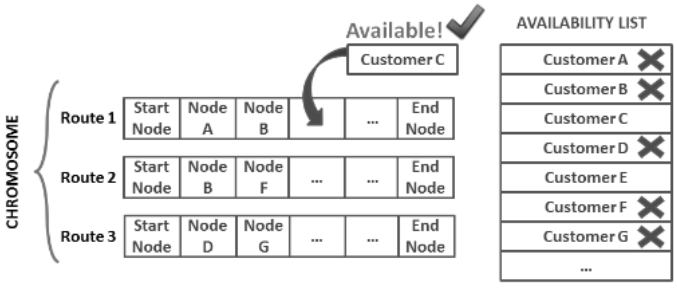


Figura 7: Esquema obtenção de uma solução válida (seis vértices e dois veículos).

No que diz respeito ao processo evolutivo, este inclui duas operações genéticas: cruzamento e mutação. O processo de cruzamento é realizado através da troca completas de rotas entre dois cromossomas, resultando na criação de dois novos cromossomas (ver Figura 8). As rotas a serem trocadas são seleccionadas de forma aleatória, e um bloco inteiro de rotas consecutivas é copiado para o novo cromossoma, tal como pode ser observado na Figura 8.

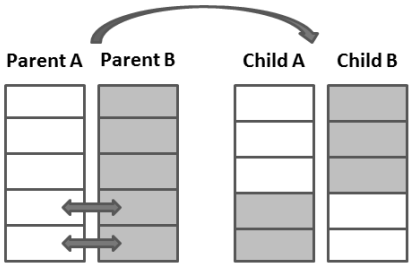


Figura 8: Crossover entre os dois progenitores (cada barra representa uma rota).

Os cromossomas utilizados para cruzamento são escolhidos com base na selecção por roda de roleta, também conhecida como a selecção proporcional à aptidão, que avalia a probabilidade de um cromossoma ser utilizado. Um cromossoma com o nível de aptidão mais elevado é mais provável de ser escolhido como um progenitor para a próxima geração. O número de cromossomos selecionados deve ser um número par uma vez que esses cromossomas são distribuídos em duas listas de progenitores: A e B.

Na equação (2) a probabilidade p de um cromossoma i ser selecionado é indicado como p_i ; a aptidão f desse cromossoma i é f_i e N é o tamanho da população.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2}$$

A outra operação genética utilizada é a mutação e consiste na remoção de um vértice de uma rota escolhida aleatoriamente dentro de um cromossoma. Em seguida, é feita uma tentativa para inserir um ou mais vértices da lista disponibilidade do cromossoma (vértices que não foram visitados na solução progenitora). Os vértices dessa lista são verificados um a um, em ordem aleatória, e quando um vértice corrente representar uma opção válida é adicionado à rota. Este processo representa uma tentativa de adicionar novos vértices, tantos quantos for possível. Na Figura 9 é apresentado um esquema simples da mutação, em que apenas um vértice é removido de um percurso. Há também a possibilidade de realizar mutações mais complexas através da remoção de mais do que um vértice a partir de uma rota (multi-*switch*), e o processo é executado de uma maneira semelhante como na mutação simples (*switch*-único).

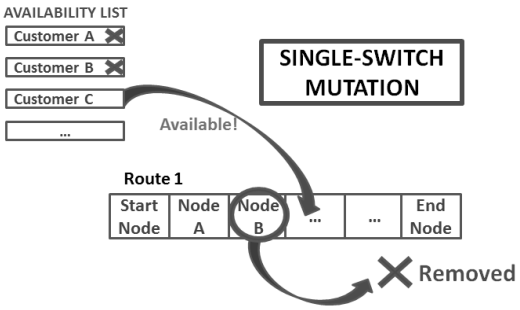
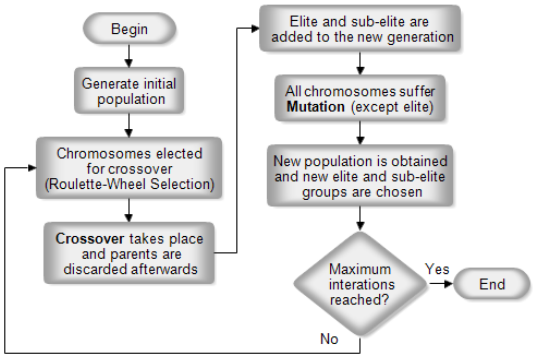


Figura 9: Esquema de uma mutação-switch único.

Existem duas classes especiais de cromossomas dentro da população, a elite e a sub-elite, para as quais diferentes regras são aplicadas. A classe de elite é o grupo dos cromossomas mais aptos dentro da população de uma certa geração, isto é, são os cromosomas que apresentam os melhores valores da função de avaliação, normalmente relacionada com a função objetivo. Estes cromossomas são imunes à mutação e ao cruzamento até que sejam substituídos por novos cromossomas em futuras gerações. Quanto à classe sub-elite, esta inclui os cromossomas mais aptos imediatamente após a elite. Este grupo é mantido intacto durante a fase de cruzamento, mas já sofre mutação. Durante o processo evolutivo, os cromossomas resultantes de ambos os processos de mutação e cruzamento são mantidos, mesmo que tenham menos valor de aptidão (*fitness*) que os cromossomas que os originaram. Em seguida ilustra-se um fluxograma do algoritmo GATOP-2-

Algoritmo GATOP-2



O algoritmo apresentado foi implementado utilizando a linguagem de programação JAVA, o que resultou numa ferramenta de software que incorpora uma GUI criada com a framework Java Swing, que se apresenta na Figura 10. A GUI permite o ajuste dos seguintes parâmetros: cromossoma, tamanho da população, o nível de mutação (número de genes mutados por cromossoma) e número de cromossomas de elite e sub-elite. Há também a possibilidade de carregar diferentes instâncias, se os dados estiverem acessíveis num arquivo devidamente formatado. A GUI apresenta no painel da esquerda a atual população de cromossomas, enquanto no da direita é apresentada a melhor solução produzida em cada geração.

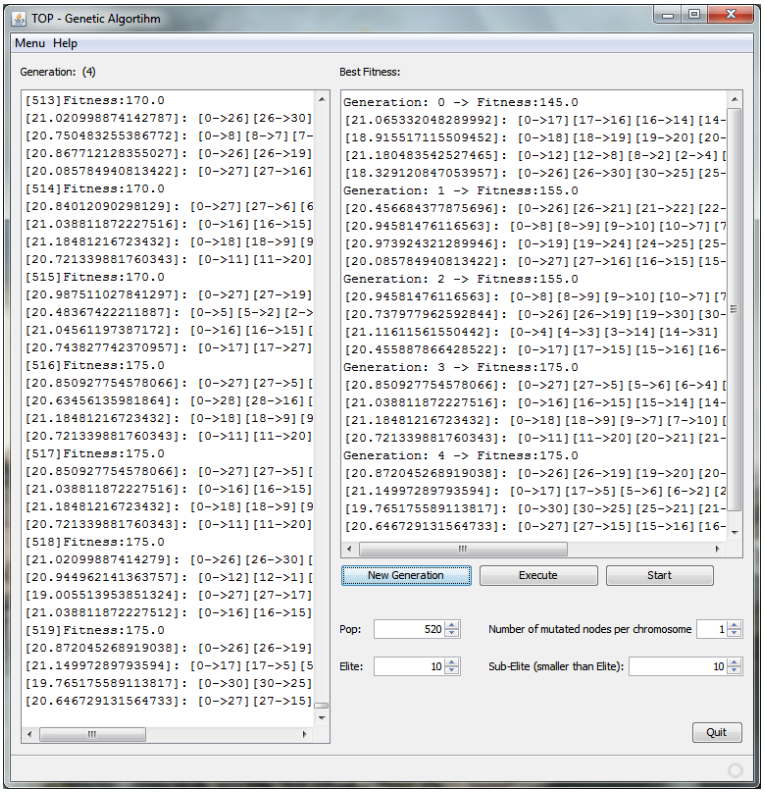


Figura 10: GUI da ferramenta GATOP-2.

Quando a aplicação executa o algoritmo, os resultados são exibidos no formato apresentado na Tabela 1, onde o símbolo # denota o índice de um cromossoma numa lista ordenada por nível de aptidão. Cada itinerário é representado como $[X]:[A \to B][B \to C][C \to D]$, onde A, B, C e D denotam vértices visitados numa rota, e X é a distância de cada caminho. A Tabela 1 representa uma solução de uma instância do TOP. A ferramenta de software apresentada é confiável e produz sempre resultados válidos para o TOP.

Generic Representation	Example Representation
[#] Fitness: F	[519] Fitness: 210.0
Route 1	$[21.2] : [0 \rightarrow 26][26 \rightarrow 3][3 \rightarrow 25][25 \rightarrow 31]$
Route 2	$[21.0] : [0 \rightarrow 5][5 \rightarrow 4][4 \rightarrow 6][6 \rightarrow 2][2 \rightarrow 31]$
Route 3	$[20.8] : [0 \rightarrow 18][18 \rightarrow 19][19 \rightarrow 20][20 \rightarrow 8][8 \rightarrow 31]$
Route 4	$[20.7] : [0 \rightarrow 27][27 \rightarrow 28][28 \rightarrow 16][16 \rightarrow 14][14 \rightarrow 31]$

Tabela 1: Apresentação das soluções no GATOP-2.

4. RESULTADOS COMPUTACIONAIS

Realizaram-se uma série de testes computacionais com o intuito de avaliar a performance dos algoritmos apresentados, comprando os resultados obtidos entre eles e também com os valores do estado da arte. Para os testes utilizaram-se 24 instâncias públicas de teste para o TOP, selecionadas aleatoriamente de entre as 320 instâncias publicadas por Chao et al. .

Os resultados obtidos nos testes com os dois algoritmos foram comparados com os melhores resultados encontrados na literatura relativa ao TOP. Os testes foram realizados com recurso a um computador laptop com processador Intel Pentium Core2Duo P8700 2.53 GHz 64-bit com 4 GB de memória RAM. Foram efecutadas dez execuções com cada um dos algoritmos em cada instância. Na Tabela 2 estão apresentadas as configurações utilizadas para os algoritmos durante os testes.

GATOP-1		**GATOP-2**	
Critério de Paragem (iterações/ciclos)	5.000	Critério de Paragem (iterações/ciclos)	5.000
População Total	55	População Total	220
Crossovers por iteração	30	Crossovers por iteração	100
Mutações por iteração	10	Mutações por iteração	210
Cromosomas Imunes	5	Cromosomas Elite	10
Cromosomas gerados aleatoriamente	10	Cromosomas Sub-Elite	10

Tabela 2: Configurações dos algoritmos GATOP-1 e GATOP-2.

Pretendeu-se que os tempos de computação para executar as 5000 iterações fosse aproximadamente igual para ambos os algoritmos GATOP-1 e GATOP-2. Esta opção implicou que a população do GATOP-1 seja significativamente menor (cerca de um quarto) que a população do GATOP-2. Obviamente esta opção prejudica o desempenho do GATOP-1 em termos de eficácia, pois não irá avaliar tantas soluções como o GATOP-2. O GATOP-1 é então um algoritmo mais lento que o GATOP-2, e isso deve-se à utilização do "*Solution Viewer*" em tempo real.

Os resultados alcançados constam na Tabela 3, onde o valores na coluna Avg correspondem ao valor médio para cada instância após as dez execuções do algoritmo. Os valores em fmin e fmax são respectivamente os valores mínimos e máximos obtidos para cada instância. Na coluna BEST SCORE constam os valores máximos actuais para cada instância e que constam na literatura.

A Tabela 4 apresenta um sumário do desempenho de cada um dos algoritmos nos testes propostos. Na coluna Avg Time consta o tempo médios em minutos que os algoritmos demoraram para encontrar o melhor valor obtido em cada instância. Na coluna % to MAX, os valores expressos indicam a taxa de sucesso alcançado com cada algoritmo, focando o valor médio relativamente ao valor máximo encontrado na literatura (Avg/BEST SCORE*100%). Comparando os desempenhos de ambos os algoritmos, o GATOP-2 é melhor que GATOP-1 em todas as instâncias testadas, obtendo um valor médio de 96.53%, contra 91.37% de GATOP-1. Além disso, o GATOP-2 é claramente mais rápido na obtenção da melhor solução.

Instance	Fitness Level (Score)						BEST SCORE
	GATOP-1			GATOP-2			
	Avg	*fmin*	*fmax*	Avvg	*fmin*	*fmax*	
p4.2.f	609.6	565	643	662.0	648	684	**687**
p4.2.o	1004.0	923	1081	1116.7	1048	1178	**1218**
p4.3.j	764.8	722	818	826.9	790	854	**861**
p4.3.p	985.7	935	1062	1119.9	1036	1174	**1222**
p4.4.k	736.7	697	775	794.5	733	821	**821**
p4.4.r	992.6	922	1054	1139.0	1110	1171	**1211**
p5.2.p	1066.5	1045	1090	1090.0	1055	**1150**	**1150**
p5.2.z	1627.0	1515	1675	1633.0	1595	1660	**1680**
p5.3.t	1225.0	1205	1245	1242.5	1220	**1260**	**1260**
p5.3.y	1500.5	1455	1555	1561.0	1520	1590	**1595**
p5.4.w	1303.5	1250	1355	1357.0	1310	1380	**1390**
p5.4.z	1501.0	1460	1535	1525.5	1455	1575	**1620**
p6.2.l	1093.2	1050	1110	1090.2	1068	**1116**	**1116**
p6.2.m	1161.6	1146	**1188**	1157.4	1134	1176	**1188**
p6.3.m	1033.8	996	**1080**	1069.8	1056	**1080**	**1080**
p6.3.n	1124.4	1104	1158	1158.6	1146	**1170**	**1170**
p6.4.m	885.0	852	906	902.4	888	**912**	**912**
p6.4.n	1051.8	1032	**1068**	1068.0	1068	**1068**	**1068**
p7.2.l	711.3	692	722	740.8	692	763	**767**
p7.2.s	967.1	910	1030	1071.3	991	1110	**1136**
p7.3.o	766.6	729	808	845.9	812	865	**874**
p7.3.t	947.8	891	1017	1072.5	1037	1112	**1116**
p7.4.p	776.7	755	814	820.8	795	844	**846**
p7.4.s	896.6	855	958	991.6	929	1020	**1022**

Tabela 3: Resultados obtidos nos testes com GATOP-1 e GATOP-2.

Instance	GATOP-1		GATOP-2	
	Avg Time (min)	% to MAX	Avg Time (min)	% to MAX
p4.2.f	5.56	88.73%	1.40	96.36%
p4.2.o	6.89	82.43%	4.38	91.68%
p4.3.j	6.86	88.83%	1.55	96.04%
p4.3.p	8.51	80.66%	3.24	91.64%
p4.4.k	6.73	89.73%	1.10	96.77%
p4.4.r	9.21	81.97%	3.81	94.05%
p5.2.p	3.50	92.74%	0.28	94.78%
p5.2.z	5.03	96.85%	0.61	97.20%
p5.3.t	5.31	97.22%	0.16	98.61%
p5.3.y	5.39	94.08%	0.38	97.87%
p5.4.w	5.08	93.78%	0.42	97.63%
p5.4.z	6.47	92.65%	0.63	94.17%
p6.2.l	3.38	97.96%	0.99	97.69%
p6.2.m	3.96	97.78%	0.37	97.42%
p6.3.m	4.07	95.72%	0.17	99.06%
p6.3.n	3.73	96.10%	0.39	99.03%
p6.4.m	5.07	97.04%	0.13	98.95%
p6.4.n	4.49	98.48%	0.25	100.00%
p7.2.l	4.41	92.74%	1.47	96.58%
p7.2.s	6.57	85.13%	2.33	94.30%
p7.3.o	6.30	87.71%	1.80	96.78%
p7.3.t	7.03	84.93%	2.33	96.10%
p7.4.p	6.41	91.81%	1.85	97.02%
p7.4.s	7.27	87.73%	2.18	97.03%

Tabela 4: Estatísticas gerais dos testes.

A Tabela 5 resume algumas estatísticas relacionadas com a consistência e exactidão dos algoritmos, onde se pode observar o nível de desvio entre os valores fmin, fmax e BEST SCORE. É de destacar a superior consistência do GATOP-2 face ao GATOP-1, pois existe em média uma menor diferença entre os valores fmax e fmin. Em termos de exactidão, de novo o GATOP-2 mostrou-se superior ao obter uma diferença média de 0.9% em relação aos melhores valores conhecidos para cada instância (BEST SCORE). O algoritmo GATOP-2 conseguiu igualar os melhores resultados em 7 das 24 instâncias, enquanto que o GATOP-1 conseguiu igualar os melhores resultados apenas em 3 instâncias.

	GATOP-1	**GATOP-2**
Average gap *fmax* - BEST SCORE	4.8 %	0.9 %
Maximum gap *fmax* - BEST SCORE	13.1 %	3.9 %
Average gap *fmax – fmin*	8.2 %	6.1 %
Maximum gap *fmax - fmin*	14.6 %	11.8 %
Times equal to BEST SCORE	3	7

Tabela 5: Estatísticas intrínsecas dos algoritmos.

As principais causas que podem explicar a dificuldade em alcançar melhores resultados com GATOP-1 e GATOP-2 são a configuração dos parâmetros e o volume de operações genéticas (cruzamentos e mutações) que ocorrem a cada iteração, bem como os mecanismos definidos para executar essas operações genéticas. Causas específicas identificadas para o menor desempenho do GATOP-1 podem ser apontadas a técnica utlizada na construção de rotas (construção em série, uma rota após outra, em vez do processamento paralelo) e também a otimização menos eficaz das rotas, à luz dos operadores genéticos utilizados. Por fim, o facto de serem avaliadas menos soluções (por ser um método mais lento) justifica uma boa parte do desvio encontrado.

Os valores alcançados com os algoritmos propostos podem ser melhorados através de um aumento geral dos valores de parâmetros como a quantidade de cromosomas na população, o número de cromosomas imunes ou pertencentes às classes elite e sub-elite, e também a quantidade de cruzamentos e mutações que ocorrem por iteração. Contudo, esses incrementos traduzem-se em mais tempo dispendido para obter a solução, e torna-se uma opção menos válida neste contexto.

Quanto aos mecanismos utilizados nas operações genéticas, a mutação utilizada nos testes com o GATOP-2 foi a *single-switch* por ser mais rápida, no entanto, ao se aplicar a mutação *multi-switch*, obtiveram-se melhores resultados. No caso dos cruzamentos, no algoritmo GATOP-2, os cromosomas escolhidos para gerar a nova população são descartados no final dessa operação, e desse modo pode-se perder boas soluções.

No algoritmo GATOP-1, a construção de rotas é feita de um modo sequencial dentro de cada cromosoma, ou seja, uma rota só pode começar a ser construída após a anterior estar finalizada. Assim, as primeiras rotas ficam com mais vértices e mais completas, e as últimas rotas no cromosoma ficam mais vazias e sub-aproveitadas, o que pode prejudicar a solução global. Tal fenómeno não sucede tão explicitamente no GATOP-2, onde as rotas são construídas em modo quase paralelo, havendo uma melhor distribuição dos vértices. Esta diferença na construção das rotas pode explicar a diferença de desempenho entre os dois algoritmos.

Um fenómeno que se observou nos testes com o GATOP-1 foi a sub-otimização das rotas, onde se verificaram situações como a representada na Figura 11, em que um conjunto de vertices fazem parte da mesma rota e são visitados numa sequência que não é ideal, visto que se demora mais tempo a efectuar a rota do que se optarmos por visitar os mesmo número de vertices numa sequência diferente, mantendo-se igual o número de prémios recolhidos ao longo do percurso.

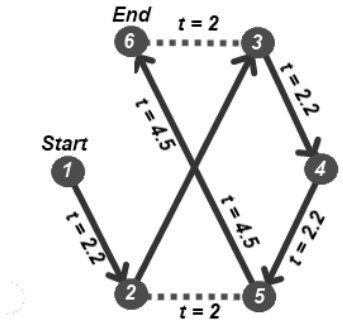


Figura 11: Exemplo de rota sub-aproveitada.

Assim, a introdução de um operador genético que verificasse a ocorrência de situações como esta e as corrigisse, poderia resultar em rotas mais optimizadas em termos de tempo gasto em viagem, permitindo a inclusão de mais vértices na solução. Um exemplo de como corrigir a situação apresentada consta na Figura 12.

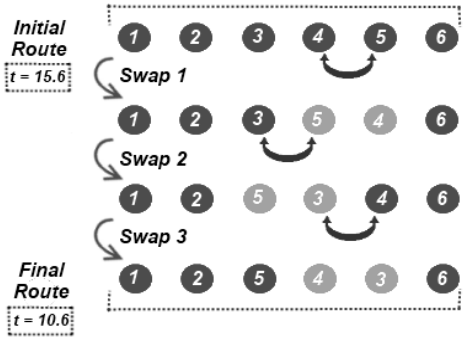


Figura 12: Exemplo de correção de uma rota sub-aproveitada.

5. CONCLUSIÓNS

Ao longo deste artigo foram apresentadas duas ferramentas de software denominadas GATOP-1 e GATOP-2, as quais permitem resolver o problema de orientação de equipas (TOP - *Team Orienteering Problem*) através de algoritmos genéticos. Os dois algoritmos genéticos foram testados e analisados num conjunto de 24 intâncias públicas, tendo-se concluído que o desempenho médio do GATOP-2 (96.53%) é claramente melhor do que o de GATOP-1 (91.37%) quando comparados aos melhores valores conhecidos para essas mesmas instâncias. Apesar de não serem resultados que espelham perfeição, continuam a ser valores bastante bons, principalmente no caso de GATOP-2. Ambos os algoritmos podem obter performance superior mas necessitam de melhorias ao nível do equilíbrio entre os parâmetros dos algoritmos genéticos, ou ao nível dos operadores genéticos como o cruzamento entre cromossomas e a mutação destes. No geral, as ferramentas desenvolvidas são interactives e de fácil utilização, e permitem resolver com sucesso o TOP.

Como trabalho futuro pretendemos continuar o estudo do TOP e das suas variantes que consideram janelas temporais ou incluem restrições à capacidade dos veículos. Em termos de métodos de resolução, os algoritmos genéticos continuarão a ser a nossa aposta para obter melhores resultados, sendo o mecanismo utilizado nos algoritmos genéticos celulares o nosso foco de estudo futuro.

AGRADECIMENTOS

Este trabalho é financiado por:

Fundos FEDER através do Programa Operacional Fatores de Competitividade – COMPETE e por Fundos Nacionais através da FCT – Fundação para a Ciência e Tecnologia no âmbito do Projeto: FCOMP-01-0124-FEDER-022674

GATOP - Genetic Algorithms for Team Orienteering Problem (Ref PTDC/EME-GIN/ 120761/2010), financiado por fundos nacionais pela FCT / MCTES e co-financiado pelo by the European Social Development Fund (FEDER) through the COMPETE Programa Operacional Fatores de Competitividade (POFC) Ref FCOMP-01-0124-FEDER-020609.

REFERENCIAS

Archetti, C., Hertz, A. and Speranza, M. G. (2007) Metaheuristics for the team orienteering problem. Journal of Heuristics,13, 49-76.

Bouly, H., Dang, D. C. and Moukrim, A. (2010) A memetic algorithm for the team orienteering problem. 4OR, 8, 49-70.

Butt, S. E. and Cavalier, T. M.(1994) A heuristic for the multiple tour maximum collection problem. Computers and Operations Research,21, 101-111.

Chao, I. M., Golden, B. and Wasil, E. A. (1996) Theory and Methodology - The Team Orienteering Problem. European Journal of Operational Research,88, 464-474.

Czyzyk, J., Mesnier,M. P., Moré. J. J. (1998) The NEOS server. IEEE J. Computational Sci. Engrg., 5, 68-75.

Ke, L., Archetti, C. and Feng, Z. (2008) Ants can solve the team orienteering problem. Computers and Industrial Engineering,54, 648-665.

Sevkli, A. Z., and Sevilgen, F. E. (2012) Discrete particle swarm optimization for the team orienteering problem. Turkish Journal of Electrical Engineering and Computer Sciences, 20, 231-239.

Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., and Van Oudheusden, D. (2010) A path relinking approach for the team orienteering problem. Computers and Operations Research, 37, 1853-1859.

Tasgetiren, M. F. (2002) A genetic algorithm with an adaptive penalty function for the orienteering problem. Journal of Economic and Social Research,4, 1-26.

Vansteewegen, P., Souffriau, W., Vanden Berghe, G. and Van Oudheusden, D. (2009) A guided local search metaheuristic for the team orienteering problem. European Journal of Operational Research, 196, 118-127.