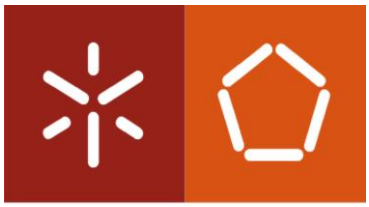

Universidade do Minho
Escola de Engenharia

Edgar Wchua Pires Guilherme

O Problema de Orientação de Equipas Capacitado com Janelas Temporais Aplicado à Recolha e Transporte de Leite Cru.

Tese de Mestrado
Mestrado Integrado em Engenharia e Gestão Industrial

Trabalho efetuado sob a orientação do
Professor Doutor José António Vasconcelos Oliveira

Outubro de 2016

DECLARAÇÃO

Nome:
Edgar Wchua Pires Guilherme
Endereço eletrónico: wchua043@gmail.com Telefone: 933 875 752
Número do Bilhete de Identidade: 875C3347X
Título da dissertação:
O Problema de Orientação de Equipas Capacitado com Janelas Temporais Aplicado à Recolha e Transporte de Leite Cru.
Orientador(es):
Professor Doutor José António Vasconcelos Oliveira
Ano de conclusão: 2016
Designação do Mestrado:
Mestrado Integrado em Engenharia e Gestão Industrial

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respetiva, deve constar uma das seguintes declarações:

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA DISSERTAÇÃO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;
2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA DISSERTAÇÃO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;
3. DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA TESE/TRABALHO

Universidade do Minho, ___/___/_____
Assinatura:

AGRADECIMENTOS

O desenvolvimento desta dissertação marca o fim de uma fase muito importante da minha vida e um objetivo concretizado.

Primeiramente agradeço a Deus pela minha vida, saúde e pelas bênçãos de conhecimento e sabedoria.

A realização e conclusão deste trabalho só foi possível pelo esforço dedicado e pelo apoio demonstrado por várias pessoas, em especial:

A minha mãe Maria Zany Andrade Pires pelo amor, carinho e apoio dado nesses anos todos de estudos. Sobretudo agradeço pelo esforço feito a fim de que eu pudesse realizar os meus estudos. Sem ela não seria possível atingir esse objetivo.

O professor Dr. José António Vasconcelos Oliveira, pela orientação e disponibilidade demonstrada ao longo da realização deste projeto.

A minha namorada Faty pelo apoio dado.

Aos meus irmãos, irmãs e amigos que me ajudaram direta ou indiretamente neste trabalho.

Ao governo de Cabo Verde pela bolsa de estudos atribuída porque sem a mesma não seria possível a realização do curso e ao governo de Portugal pelo subsídio de propinas.

RESUMO

A globalização obriga, de certa forma, a que as empresas tenham maior rigor na qualidade dos produtos e serviços prestados, bem como a entrega dos produtos e serviços certos, nos locais certos e nas horas certas, aumentando assim os seus níveis de serviço. Um setor muito importante nesse processo, porém muito dispendioso, é o transporte, o que torna essencial a sua incessante otimização com vista a minimizar os seus custos.

Nesta dissertação propõe-se a aplicação de modelos baseados nos problemas de orientação de equipas com restrições de capacidades e janelas temporais com vista a otimizar a recolha e transporte de leite cru por parte das pequenas e médias empresas portuguesas.

Inicialmente apresenta-se o atual estado da arte sobre os problemas de orientação, nomeadamente as diversas variantes dos problemas de orientação de equipas. De seguida é exposto o estado da arte dos algoritmos genéticos bem como da *framework* BRKGA.

Apresenta-se o problema da recolha e transporte de leite cru em Portugal por parte das pequenas e médias empresas bem como a modelação do mesmo num modelo matemático representativo para a obtenção de soluções a nível computacional.

Apresenta-se a aplicação *standalone* desenvolvida, em linguagem C++ com recurso ao IDE NetBeans e o Qt Creator, para facilitar a criação de instâncias e obtenção de soluções para as mesmas.

Apresenta-se ainda a análise dos resultados experimentais obtidos nos testes computacionais realizados às diversas instâncias geradas aleatoriamente. Os resultados são expostos em folhas de cálculo e gráficos para permitir uma melhor análise dos mesmos.

Por fim, são expostas as principais conclusões retiradas do trabalho desenvolvido e as metas para um trabalho futuro.

PALAVRAS-CHAVE

Problemas de Orientação de Equipas com Restrições de Capacidade e Janelas Temporais, Algoritmos Genéticos, Otimização da Cadeia de Abastecimento, Otimização de Rotas, BRKGA

ABSTRACT

Globalization forces, in a way, the companies have greater rigour in the quality of products and services, as well as the delivery of right goods and services, in the right places and at the right time, thereby increasing their levels of services. A very important sector in that process, though very expensive, is transport. Therefore, it is essential a continuous optimization in order to reduce their costs.

This dissertation proposes the application of models based on the capacitated team orienteering problems with time windows constraints in order to optimize the collection and transport of raw milk from small and medium-sized Portuguese's enterprises.

Initially presented the current state of the art about the orienteering problems, particularly the several variants of teams orienteering problems. Then is presented the state of the art of genetic algorithms as well as of the framework BRKGA.

It is presented the problem of the collection and transport of raw milk in Portugal by small and medium-sized enterprises as well as the modelling of a representative mathematical model in order to obtain solutions in a computational level.

It is presented the standalone application developed in C++ using the NetBeans IDE and the Qt Creator IDE to facilitate the creation of instances and obtaining solutions for them.

Presents the analysis of the results obtained in tests conducted at several instances randomly generated. The results are displayed in spreadsheets and graphics for a better analysis of them.

Finally, the main conclusions from this work and the goals for future work are exposed.

KEYWORDS

Capacitated Team Orienteering Problem with Time Windows, Genetic Algorithms, Supply Chain Optimization, Route Optimization, BRKGA

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÓNIMOS

SIGLA	SIGNIFICADO
ACO	Ant Colony Optimization
APROLEP	Associação dos Produtores de Leite de Portugal
BRKGA	Biased Random Key Genetic Algorithm
CGRMP	Column Generation Restricted Master Problem
CTOP	Capacitated Team Orienteering Problem
CTOPTW	Capacitated Team Orienteering Problem with Time Windows
FVNS	Fast Variable Neighbourhood Search
GRASP	Greedy Randomized Adaptive Search Procedure
HGTOPTW	Hierarchical Generalization of TOPTW
ILS	Iterated Local Search
MPOPMTW	Multi-Period Orienteering Problem with Multi Time Windows
OP	Orienteering Problem
PR	Path Relinking
SVNS	Slow Variable Neighbourhood Search
SkVNS	Skewed Variable Neighbourhood Search
SVRPTW	Selective Vehicle Routing Problem with Time Windows
TOP	Team Orienteering Problem
TOPCTW	Team Orienteering Problem with Capacity constraint and Time Windows
TOPTW	Team Orienteering Problem with Time Windows
TS	Tabu Search
TSA	Tabu Search Admissible
TSF	Tabu Search Feasible
TSU	Tabu Search Unfeasible
TTDP	Tourist Trip Design Problem
VNS	Variable Neighbourhood Search
VRPTW	Vehicle Routing Problem with Time Windows
WTILS	Well-Tuned ILS

1. INTRODUÇÃO

Neste capítulo serão descritos um breve enquadramento da problemática da recolha e transporte do leite cru, a motivação para a realização desta dissertação, bem como os objetivos a atingir. Apresenta-se também a estrutura adotada para a elaboração do documento.

1.1 Enquadramento

A concorrência entre as organizações obriga-as a terem maior rigor na qualidade dos produtos e serviços prestados, bem como a entrega dos produtos e serviços certos, nos locais certos e nas horas certas. Para fazer face à concorrência, as empresas necessitam de efetuar um planeamento logístico adequado para poder sobreviver à dinâmica do mercado. Entende-se por logística um conjunto de atividades de movimentação e armazenagem necessárias para facilitar o fluxo de produtos e informações entre os elos da cadeia de abastecimento. A componente mais dispendiosa e por isso mais importante das atividades logísticas é o transporte, sendo por isso importante a sua otimização e a redução dos seus custos.

No verão de 2015, a crise económica abateu-se de novo sobre a produção de leite na Europa fazendo com que as empresas do ramo tivessem como objetivo principal a redução dos custos, nomeadamente os custos de produção e custos logísticos, em particular os de transporte de leite cru (APROLEP, 2015). Apesar da crise financeira ter produzido uma redução significativa do número de explorações leiteiras, há atualmente um grande número de produtores de leite em Portugal (Gabinete de Planeamento e Politica - GPP, 2015). Grande parte destas explorações leiteiras são de pequena dimensão e o leite produzido é transferido para unidades de processamento através do transporte rodoviário, enquadrado numa rede de recolha.

As alterações resultantes da dinâmica do mercado obrigam a explorar novas reconfigurações para o transporte de leite cru das explorações para as unidades de processamento. Há uma série de restrições associados ao leite cru, por exemplo, “o leite deve ser mantido a uma temperatura não superior a 10ºC, a não ser que tenha sido recolhido nas duas horas seguintes à ordenha.” (Brito et al., 2015), o que torna particularmente relevante a otimização do seu transporte para que seja recolhido e entregue atempadamente.

Uma das reconfigurações aplicável ao transporte de leite cru encaixa-se nos problemas de roteamento de veículos baseados em prémios e com restrições de capacidade e janelas temporais, que por sua vez "enquadram-se no âmbito dos problemas de orientação de equipas "*Team Orienteering Problem*" (TOP) introduzido inicialmente por Butt and Cavalier (1994) como *Multiple Tour Maximum Collection Problem*" (Abreu, 2014). Este tipo de problema de roteamento tem vindo a ser aplicado com sucesso na modelação de recolha de produtos ou materiais que necessitam de ser transferidos para unidades de consolidação/processamento no âmbito da cadeia de abastecimento de vários setores de atividade, proporcionando reduções significativas do custo de transporte.

Atendendo à especificidade da recolha e transporte de leite cru pretende-se modelar a realidade portuguesa baseada nas pequenas explorações leiteiras como um problema de roteamento de veículos baseado em prémios com restrições de capacidades e janelas temporais (*Capacitated Team Orienteering Problem with Time Windows* – CTOPTW). Trata-se de um problema de elevada complexidade em termos de otimização e requer a utilização de metodologia heurística apropriada à resolução de problemas NP-difíceis.

1.2 Objetivo

Nesta dissertação pretende-se estudar a gestão das rotas de forma a reduzir os custos de transporte de leite cru das pequenas explorações leiteiras e a garantir a melhor qualidade do leite recolhido e entregue nas respetivas unidades de processamento. Utilizar-se-á meta-heurísticas para a resolução do problema, nomeadamente *Biased Random-Key Genetic Algorithm* (BRKGA) que é uma meta-heurística de busca geral para encontrar soluções ótimas ou quase ótimas para problemas de otimização combinatória difíceis. Para isso desenvolver-se-á uma aplicação *standalone* baseada na *framework* BRKGA para modelar e resolver instâncias públicas do problema CTOPTW.

1.3 Motivação

O gosto pela área de otimização e a sua aplicação nas diversas áreas de estudos bem como nos problemas do quotidiano evoluiu muito no decorrer dos anos da formação académica. Um dos principais motivos para a escolha desse tema para a realização do projeto de dissertação é o fato de englobar um ramo da otimização muito interessante, que é o problema de orientação de equipas com restrições de capacidade e janelas temporais, e porque

permite ajudar na obtenção de soluções que reduza os custos das empresas do setor leiteiro e não só, melhorando também, de certa forma, a qualidade do leite entregue ao consumidor final.

1.4 Organização da dissertação

Em termos de estrutura, a dissertação encontra-se dividida nos seguintes capítulos:

- Introdução – Procura-se introduzir e contextualizar o tema, apresentando: a motivação para a realização desta dissertação; uma breve descrição da problemática em estudo; a definição dos objetivos a atingir; e por fim uma descrição da organização deste documento.
- Revisão bibliográfica – Apresenta-se uma revisão do estado da arte relacionada com os problemas de orientação de equipas, algoritmos genéticos e a *framework* BRKGA.
- Métodos – Descreve-se detalhadamente o problema subjacente, a modelação matemática e os métodos e técnicas utilizadas para a sua resolução. Aborda também o desenvolvimento de um *software* de apoio à decisão para a elaboração de rotas, e que permite a realização de experiências computacionais.
- Resultados – Reportam-se as experiências computacionais com instâncias públicas dos problemas CTOPTW e analisa-se os resultados obtidos.
- Conclusões – Apresentam-se as principais conclusões do projeto e apontam-se recomendações para trabalhos futuros.

CTOPTW aplicado à recolha e transporte de leite cru.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo apresentar-se-á uma revisão do estado da arte sobre: os problemas de orientação de equipas, os algoritmos genéticos e a *framework* BRKGA.

2.1 Problemas de orientação

O problema de orientação (Orienteering Problem – OP) tem origem no desporto de orientação. Neste jogo, os jogadores começam num ponto de controlo (origem) especificado e tentam visitar o maior número de postos de controlo possível e voltar à origem dentro de um período de tempo limite. Cada ponto de controlo tem uma pontuação e o objetivo é maximizar a pontuação total recolhida. O OP pode ser vista como combinação entre o problema da mochila e o problema do caixeiro-viajante (Vansteenwegen, Souffriau, & Oudheusden, 2011).

Este problema pode ser definido da seguinte forma: dado um conjunto de N vértices em que cada um tem uma recompensa não negativa S_i associada. O tempo de viagem t_{ij} entre o vértice i e o vértice j é conhecido para todos os pares de vértices. Em princípio pode não serem visitados todos os vértices, porque existe um tempo máximo T_{max} de operação. O objetivo do OP é encontrar um caminho, limitado pelo T_{max}, que visite alguns vértices, de tal forma que a recompensa seja maximizada. Assume-se que cada vértice só pode ser visitado uma vez, que o ponto inicial do caminho é o vértice 1, e que o vértice N é o ponto terminal do caminho.

O OP também pode ser definido por um grafo $G = (V, A)$ onde $V = \{v_1, \dots, v_N\}$ é um conjunto de vértices e $A = \{a_{ij}, \dots, a_{mN}\}$ é um conjunto de arcos. A cada vértice $v_i \in V$ está associada uma recompensa não negativa S_i e a cada arco $a_{ij} \in A$ está associado um tempo de viajem t_{ij}. O OP consiste em determinar um caminho hamiltoniano $G'(\in G)$ ao longo de um subconjunto de V, compreendendo os vértices origem e terminal, e tendo um comprimento não superior ao T_{max}, a fim de maximizar a recompensa total recolhida.
Em muitas aplicações do OP, v_1 coincide com o v_N e assume-se que o grafo é completo e não direcionado e que o tempo de viagem entre dois vértices é simétrico, ou seja, $t_{ij} = t_{ji}$.

Usando a notação acima referida, o OP pode ser formulado como um problema de programação inteira com as seguintes variáveis de decisão:

$$X_{ij} = \begin{cases} 1, & se\ o\ vértice\ j\ é\ visitado\ após\ o\ vértice\ i\ ser\ visitado \\ 0, & caso\ contrário \end{cases}$$

$u_i = posição\ do\ vertice\ i\ no\ caminho\ G'$

$$Max \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i * X_{ij} \tag{1}$$

Sujeito a:

$$\sum_{j=2}^{N} X_{1j} = \sum_{i=1}^{N-1} X_{iN} = 1 \tag{2}$$

$$\sum_{i=1}^{N-1} X_{ik} = \sum_{j=2}^{N} X_{kj} \ \leq 1 \quad \forall k = 2, \dots, N-1 \tag{3}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} * X_{ij} \ \leq T_{max} \tag{4}$$

$$2 \ \leq \ u_i \ \leq N \quad \forall i = 2, \dots, N \tag{5}$$

$$u_i - u_j + 1 \leq (N-1)\left(1 - X_{ij}\right) \quad \forall i, j = 2, \dots, N \tag{6}$$

$$X_{ij} \in \{0, 1\} \quad \forall i, j = 1, \dots, N \tag{7}$$

A função objetivo maximiza a recompensa total recolhida e está representada na equação (1). A restrição (2) garante que o caminho começa no vértice 1 e termina no vértice N. A conetividade do caminho e a garantia de que cada vértice é visitado no máximo uma vez é assegurada pela restrição (3). A restrição do limite temporal é garantida pela restrição (4). As restrições (5) e (6) previnem a existência de sub-rotas.

Existem muitas extensões dos problemas OP resultantes das diferentes restrições impostas aos problemas de orientação. Neste documento apresentar-se-ão os problemas de orientação de equipas (Team Orienteering Problem – TOP), problemas de orientação de equipas com restrições de capacidade (Capacitated Team Orienteering Problem – CTOP), problemas de orientação de equipas com restrições de janelas temporais (Team Orienteering Problem with Time Windows – TOPTW) e problemas de orientação de equipas com

restrições de capacidade e janelas temporais (Capacitated Team Orienteering Problem with Time Windows – CTOPTW). Apresentar-se-á para cada um dos problemas referidos, uma breve descrição, o modelo matemático e casos de aplicação.

2.1.1 Team Orienteering Problem – TOP

O problema de orientação de equipas constitui uma extensão de OP cujo objetivo é determinar P caminhos, cada um limitado pelo T_{max}, que maximiza a recompensa total recolhida. O TOP corresponde a jogar o jogo de orientação por equipas de várias pessoas recolhendo recompensas no mesmo tempo limite. Dado que o problema OP é NP-difícil (Golden, Levy, & Vohra, 1987), e sendo o problema TOP uma extensão do OP, tem pelo menos a mesma complexidade.

O TOP foi apresentado pela primeira vez na literatura por Butt & Cavalier (1994) sob o nome de *Multiple Tour Maximum Collection Problem* e posteriormente Chao et al. (1996) introduziram formalmente o problema e concebeu um dos conjuntos mais utilizados de instâncias de teste.

Modelo matemático

O TOP pode ser formulado como um problema de programação inteira com as seguintes variáveis de decisão:

$$X_{ijp} = \begin{cases} 1, & se\ no\ caminho\ p, o\ vértice\ j\ é\ visitado\ após\ o\ vértice\ i\ ser\ visitado \\ 0, & caso\ contrário \end{cases}$$

$$Y_{ip} = \begin{cases} 1, & se\ o\ vértice\ i\ é\ visitado\ no\ caminho\ p \\ 0, & caso\ contrário \end{cases}$$

$$u_i = posição\ do\ vertice\ i\ no\ caminho\ p$$

$$Max \sum_{p=1}^{P} \sum_{i=2}^{N-1} S_i * Y_{ip} \tag{8}$$

Sujeito a:

$$\sum_{p=1}^{P} \sum_{j=2}^{N} X_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} X_{iNp} = P \tag{9}$$

$$\sum_{p=1}^{P} Y_{kp} \leq 1 \quad \forall k = 2, \dots, N-1 \tag{10}$$

$$\sum_{i=1}^{N-1} X_{ikp} = \sum_{j=2}^{N} X_{kjp} = Y_{kp} \quad \forall k = 2, \dots, N-1 \quad \forall p = 1, \dots, P \tag{11}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} * X_{ijp} \leq T_{max} \quad \forall p = 1, \dots, P \tag{12}$$

$$2 \leq u_{ip} \leq N \quad \forall i = 2, \dots, N \quad \forall p = 1, \dots, P \tag{13}$$

$$u_{ip} - u_{jp} + 1 \leq (N-1) * \left(1 - X_{ijp}\right) \quad \forall i, j = 2, \dots, N \quad \forall p = 1, \dots, P \tag{14}$$

$$X_{ijp}, Y_{ip} \in \{0,1\} \quad \forall i, j = 1, \dots, N \quad \forall p = 1, \dots, P \tag{15}$$

A função objetivo (8) maximiza a recompensa total recolhida. A restrição (9) obriga a que o número de arcos que saem do vértice 1 é igual ao número de arcos que incidem no vértice N. A garantia de que cada vértice é visitado no máximo uma vez é assegurada pela restrição (10). A restrição (11) garante a conetividade de cada caminho, como demonstrado na Figura 1.

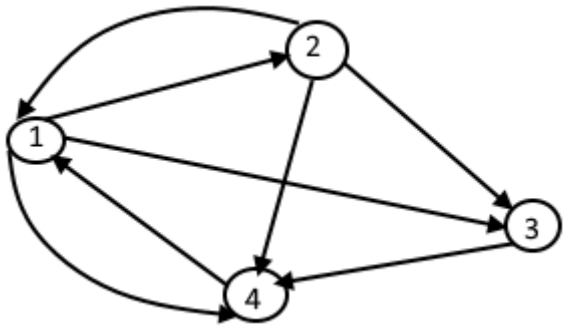

Figura 1 - Esquema de conetividade de um caminho

A propriedade da conectividade pode ser ilustrada na Figura 1 considerando um caminho p que visita o vértice 2; se no caminho p, o vértice 2 é visitado (Y2p = 1), significa que no caso da Figura 1 só poderá ser a partir do vértice 1 (X12p=1) então, ou o vértice 3 ou o vértice 4 terão de ser imediatamente visitados, porque X23p + X24p = 1.

A restrição do limite temporal é garantida pela restrição (12). A inexistência de sub-rotas é garantida pelas restrições (13) e (14).

Abordagem de soluções

Um algoritmo exato para resolver o TOP, usando geração de colunas (*column generation*), foi publicado por Butt & Ryan (1999). Este algoritmo é capaz de resolver os problemas com até 100 vértices, quando o número de vértices em cada caminho permanece pequeno, uma média de seis vértices por caminho.

Boussier et al. (2006) apresentaram um algoritmo exato, baseado no esquema *branch-and-price*, para solucionar problemas TOP e TOPTW. O esquema *branch-and-price* é obtido a partir da conjugação do método de geração de colunas com o *branch-and-bound*. A fim de melhorar o desempenho, diferentes técnicas de aceleração são aplicadas: "*limited discrepancy search*", um método de árvore de busca heurística de programação por restrições, "*loading label*" e "*meta extensions*", dois procedimentos de pré-processamento que podem ser interpretadas como regras de prioridade.

Esta abordagem permite resolver problemas com 100 vértices, onde alguns podem ser selecionados (até cerca de 15 por caminho), em menos de duas horas de tempo de computação.

A primeira heurística publicada para o TOP foi desenvolvida por Chao et al. (1996) e é semelhante à heurística *five-steps* para problemas OP. Em vez de apenas selecionar o melhor caminho, os melhores caminhos são selecionados e dois passos de reiniciação são usados em vez de uma.

Archetti et al. (2007) desenvolveram duas variantes da heurística *Tabu Search - TS*, um *Slow Variable Neighbourhood Search - SVNS* e uma *Fast Variable Neighbourhood Search - FVNS*. Todas as quatro meta-heurísticas começam a partir de uma solução incumbente s, fazendo um salto para uma solução intermédia s'. Em seguida, a TS é usada para tentar melhorar a solução s' e a nova solução s'' é comparada com a solução s. Na estratégia *Variable Neighbourhood Search - VNS*, a solução s'' apenas é aceite se tiver maior pontuação do que a solução s enquanto que na estratégia TS, s'' sempre é aceite. Este processo é repetido até atingir o critério de paragem. Os vértices não incluídos são organizados em caminhos. A TS usa dois movimentos, o *1-move* para mover um vértice de um caminho para outro e o *swap-move* para trocar dois vértices entre caminhos. Os vértices são sempre incluídos usando uma estratégia de inserção de menor custo. Uma vez que o

algoritmo aceita soluções inviáveis, diferentes procedimentos são desenvolvidos para reduzir ou remover a inviabilidade de um caminho. Dois tipos de saltos são usados nestes algoritmos, o primeiro é uma série de movimentos *1-move* com os vértices não incluídos e o segundo troca dois conjuntos de vértices entre os caminhos selecionados e os caminhos com vértices não incluídos. Ambas as variantes de TS utilizam apenas o segundo salto. Uma variante da TS usa apenas soluções viáveis (*Tabu Search Feasible – TSF*) enquanto que a outra apenas aceita soluções inviáveis (*Tabu Search Unfeasible – TSU*). A VNS também usa a TS como pesquisa local, mas com muito menos iterações e apenas considera soluções viáveis. A duração do caminho é reduzida sempre que for melhorada a solução incumbente, usando o 2-Opt. Para comparar a qualidade das soluções, cinco diferentes funções baseadas na pontuação, duração e viabilidade são usadas. Com estes algoritmos, os autores apresentaram os melhores resultados para os problemas OP e TOP. Isto deve-se provavelmente ao fato de que todos os vértices não incluídos são agrupados em caminhos viáveis.

Uma abordagem de otimização por colónia de formigas (*Ant Colony Optimization - ACO*) foi implementada por Archetti et al. (2008) para resolver instâncias do TOP. Em cada ciclo, cada formiga constrói uma solução exequível, que é melhorada por um processo de pesquisa local e posteriormente os trilhos de feromonas são atualizados. O critério de paragem do algoritmo é o número máximo de ciclos realizados. O processo de pesquisa local consiste em encurtar cada caminho usando 2-Opt e, em seguida, inserir o maior número de vértices possível. Este procedimento é iterado até que um ótimo local é atingido. Para construir soluções exequíveis no âmbito da ACO, quatro métodos diferentes são propostos:

- Método Sequencial (*Ant Sequential - ASe*) – caminhos completos são construídos um após o outro.
- Método Aleatório Concorrente (*Ant Random-Concurrent - ARC*) – um caminho é selecionado aleatoriamente a cada iteração para adicionar um novo vértice.
- Método Determinístico Concorrente (*Ant Deterministic-Concurrent - ADC*) – a sequência de caminhos a considerar é fixo.
- Método Simultâneo (*Ant Simultaneous Method - ASi*) – a cada iteração, um vértice é adicionado a um dos caminhos até que todos os caminhos atinjam os seus comprimentos limites.

O método sequencial é um excelente compromisso entre a qualidade da solução e o tempo de cálculo. A qualidade dos resultados desse método é, pelo menos, tão boa quanto as obtidas por Archetti et al. (2007) e claramente mais rápido.

Com o objetivo de obter boas soluções para TOP em alguns segundos de tempo de computação pela primeira vez, Vansteenwegen et al. (2009) implementaram uma *framework* de pesquisa local guiada (*Guided Local Search – GLS*) e posteriormente uma *framework* de pesquisa na vizinhança variável enviesada (*Skewed Variable Neighbourhood Search* – SkVNS). Ambos os algoritmos aplicam uma combinação de procedimentos de intensificação e diversificação. Dois processos de diversificação simplesmente removem uma cadeia de atrações em cada caminho. Outro procedimento tenta recolher a propagação do tempo disponível ao longo dos diferentes caminhos dentro da solução atual, em um único caminho na nova solução. Dois tipos de processos de intensificação são concebidos, o primeiro tenta aumentar a pontuação e o segundo tenta diminuir o tempo de viagem num caminho. O algoritmo SkVNS apresenta tempo de computação e performance melhor do que a do algoritmo GLS. Porém, a qualidade do algoritmo SkVNS é sempre determinada pela combinação específica e pela sequência de diferentes movimentos.

Duas variantes híbridas do procedimento de pesquisa local com reinícios (*Greedy Randomized Adaptive Search Procedure - GRASP*) com *Path Relinking* foram concebidas por Souffriau et al. (2010). Neste algoritmo GRASP, quatro procedimentos são executados em sequência até que nenhuma melhoria for identificada durante um número fixo de iterações. Estes procedimentos são a seguir descritos:

- Procedimento de construção – este procedimento gera uma solução inicial. Com base numa relação entre a ganância e aleatoriedade, os vértices são inseridos um a um, até que todos os caminhos fiquem cheios. Devido ao caráter aleatório, uma nova solução inicial é obtida a cada iteração.
- Pesquisa local nas vizinhanças (*Local Search Neighbourhood - LSN*) – este processo é usado para melhorar a solução inicial. O procedimento de pesquisa local alterna entre a redução do tempo total da solução e aumentando a sua pontuação total, até que a solução é localmente ótima em relação a todas as vizinhanças.
- Religação do caminho (*Path Relinking - PR*) – introduz um conjunto de soluções de elite como um componente de memória de longo prazo. Além disso, considera as soluções num caminho virtual no espaço de soluções entre a solução de pesquisa local e cada solução de elite. A melhor solução encontrada nesses caminhos é devolvida.

- Atualização – o conjunto de soluções de elite é atualizado. Mantém-se um registo da melhor solução encontrada durante todas as iterações.

Modificar apenas o critério de paragem permite a mudança entre uma versão mais lenta, com excelentes resultados, e uma versão mais rápida, com apenas resultados de qualidade elevada. A qualidade dos resultados da versão mais lenta é comparável à qualidade obtida pelos melhores algoritmos de Archetti et al. (2007) e de Ke et al. (2008).

Oliveira et al. (2014) desenvolveram dois algoritmos genéticos para resolver instâncias do TOP. Um algoritmo construtivo do tipo inserção de menor custo, para construção de rotas válidas, foi incluído na decodificação dos cromossomas. O algoritmo construtivo elabora soluções rapidamente, todavia a qualidade das soluções pode ser muito baixa em termos de função objetivo. De modo a melhorar a qualidade das soluções, os autores fizeram o uso de uma meta-heurística – algoritmo genético.

Casos de aplicação

Butt & Cavalier (1994) descreveram uma aplicação do TOP para recrutamento de atletas de escolas secundárias, onde um recrutador tem de visitar várias escolas num determinado número de dias. Ele pode atribuir pontuações a cada escola com base no seu potencial de recrutamento. Porque o tempo disponível do recrutador é limitado, ele tem que escolher as escolas a visitar a cada dia para tentar maximizar o potencial de recrutamento.

Uma aplicação do TOP para afetação e encaminhamento de técnicos para serviços ao cliente foi descrito por Tang & Miller-Hooks (2005). Cada caminho representa um técnico, que só pode trabalhar um número limitado de horas por dia. Um subconjunto de clientes terá de ser selecionado, tendo em conta a importância do cliente e a urgência da tarefa.

Oliveira et al. (2013) descreveram uma aplicação do TOP para o processo de recolha seletiva de resíduos sólidos urbanos. O objetivo é maximizar a soma dos resíduos recolhidos em cada ponto de recolha visitado pelas rotas estabelecidas.

Oliveira et al. (2014) aplicaram o TOP a um sistema de apoio à decisão para transporte não urgente de doentes em veículo partilhado.

2.1.2 Team Orienteering Problem with Time Windows – TOPTW

O problema de orientação de equipas com restrições de janelas temporais constitui uma extensão de TOP, em que cada vértice tem uma janela temporal $[O_i - C_i]$ associada e apenas pode ser visitado nesse período de tempo.

Modelo matemático

Baseado na notação introduzido anteriormente, o TOPTW pode ser formulado como um problema de programação inteira mista, com as seguintes varáveis de decisão:

$$X_{ijp} = \begin{cases} 1, & se\ no\ caminho\ p, o\ vértice\ j\ é\ visitado\ após\ o\ vértice\ i\ ser\ visitado \\ 0, & caso\ contrário \end{cases}$$

$$Y_{ip} = \begin{cases} 1, & se\ o\ vértice\ i\ é\ visitado\ no\ caminho\ p \\ 0, & caso\ contrário \end{cases}$$

$$S_{ip} = o\ início\ do\ serviço\ no\ vértice\ i\ no\ caminho\ p$$

$$ST_{ip} = tempo\ de\ serviço\ no\ vértice\ i\ no\ caminho\ p$$

$$O_i = hora\ de\ início\ do\ serviço\ no\ vértice\ i$$

$$C_i = hora\ do\ término\ do\ serviço\ no\ vértice\ i$$

$$M - uma\ grande\ constante$$

$$Max \sum_{p=1}^{P} \sum_{i=2}^{N-1} S_i Y_{ip} \tag{16}$$

Sujeito a:

$$\sum_{p=1}^{P} \sum_{j=2}^{N} X_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} X_{iNp} = P \tag{17}$$

$$\sum_{i=1}^{N-1} X_{ikp} = \sum_{j=2}^{N} X_{kjp} = Y_{kp} \quad \forall k = 2, \dots, N-1 \quad \forall p = 1, \dots, P \tag{18}$$

$$S_{ip} + ST_{ip} + t_{ij} - S_{jp} \leq M(1 - X_{ijp}) \quad \forall i, j = 1, \dots, N \quad \forall p = 1, \dots, P \tag{19}$$

$$\sum_{p=1}^{P} Y_{kp} \leq 1 \quad \forall k = 2, \dots, N-1 \tag{20}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} * X_{ijp} \leq T_{max} \quad \forall p = 1, \dots, P \tag{21}$$

$$O_i \leq S_{ip} \quad \forall i = 1, \dots, N \quad \forall p = 1, \dots, P \tag{22}$$

$$S_{ip} + ST_{ip} \leq C_i \quad \forall i = 1, \dots, N \quad \forall p = 1, \dots, P \tag{23}$$

$$X_{ijp}, Y_{ip} \in \{0, 1\} \quad \forall i, j = 1, \dots, N \quad \forall p = 1, \dots, P \tag{24}$$

A função objetivo (16) maximiza a recompensa total recolhida. A restrição (17) significa que P caminhos tem origem no vértice 1 e é igual a P caminhos que chegam ao vértice N. A conetividade e o respeito das janelas temporais em cada caminho são garantidos pelas restrições (18) e (19) respetivamente. A restrição (20) assegura que todos os vértices são visitados no máximo uma vez. A restrição (21) limita o tempo estimado para cada caminho. As restrições (22) e (23) restringem o início do serviço à janela temporal.

Abordagem de soluções

O movimento 2-Opt permite obter resultados de boa qualidade para os problemas OP e TOP, mas devido à existência de janelas temporais, não pode ser aplicado para resolver eficazmente o TOPTW. Além disso, a redução do tempo de viagem alterando a ordem das visitas já não é possível, divido às janelas temporais.

Um algoritmo baseado em otimização de colônia de formigas foi desenvolvido por Montemanni & Gambardella (2009), para resolver instâncias do TOPTW. O método baseia-se na solução de uma generalização hierárquica do TOPTW – *Hierarchical Generalization of TOPTW (HGTOPTW)*.

Tricoire et al. (2010) resolveram uma generalização do TOPTW, o *Multi-Period Orienteering Problem with Multiple Time Windows* (MPOPMTW). No MPOPMTW, cada vértice pode ter mais do que uma janela temporal num dado dia e as janelas temporais podem ser diferentes em dias diferentes. Eles propuseram um algoritmo VNS e incluíram um algoritmo exato para lidar com o sub-problema de viabilidade dos caminhos. Várias experiências computacionais mostraram que alcançaram soluções de boa qualidade para problemas com 100 vértices e 2 caminhos, em torno de um minuto de tempo de computação. Esta aplicação tem por objetivo facilitar o planeamento do calendário de trabalho por trabalhadores de campo e representantes de vendas.

Vansteenwegen et al. (2009) desenvolveram uma meta-heurística muito rápida – *Iterated Local Search (ILS)* – para resolver instâncias do TOPTW. Este algoritmo aborda o problema de viagens turísticas, onde as viagens são planeadas tendo em conta as horas de abertura dos monumentos. A qualidade dos resultados obtidos por este algoritmo é pior do

que os obtidos por Montemanni & Gambardella (2009) sobre os mesmos conjuntos de testes, mas o tempo de computação é reduzido significativamente.

É impossível fazer uma comparação detalhada das abordagens de soluções apresentadas, uma vez que as instâncias de testes usadas em cada um dos casos são ligeiramente diferentes. Todavia, pode-se concluir que a abordagem ILS tem a vantagem de ser muito rápida e que as abordagens HGTOPTW e MPOPMTW têm a vantagem de obter soluções de alta qualidade.

Boussier et al. (2006) modificaram o algoritmo *branch-and-price* para o TOP, para resolver instâncias do problema de roteamento com escolha de veículos e com janelas temporais (*Selective Vehicle Routing Problem with Time Windows – SVRPTW*), com 100 vértices e até 10 caminhos. O SVRPTW generaliza o TOPTW adicionando duas restrições ao TOPTW. A primeira restrição limita a capacidade dos veículos, enquanto que cada cliente tem uma determinada procura. A segunda restrição limita a distância dos caminhos.
Deve notar-se que a imposição deste limite à distância do percurso, não é sempre o mesmo que impor o limite de tempo T_{max}. Na maioria dos casos práticos reais, visitar um vértice exige algum tempo de serviço ou de visita. Se for esse o caso, o T_{max} vai limitar o tempo de viagem bem como o tempo de serviço. Quando a distância de viagem é limitada, o horário de visita não é considerado. O SVRPTW corresponde a um VRPTW em que, devido a algum motivo logístico, nem todos os clientes podem ser visitados.

Gunawan et al. (2015) propuseram um modelo matemático para o TOPTW que incorpora restrições tais como diferentes tempos totais de viagens, diferentes origens e destinos para as rotas. Esta extensão foi motivada por uma aplicação no mundo real – o *Tourist Trip Design Problem (TTDP).* Eles desenvolveram um algoritmo ILS, denominado por *Well-Tuned ILS – WTILS*, para resolver ambos os problemas TOPTW e TTDP. Demonstraram que o WTILS leva a melhorias em termos da qualidade das soluções, mais precisamente, é capaz de melhorar 31 melhores valores de soluções conhecidas para instâncias de *benchmark.* O algoritmo demonstrou também ser eficaz para resolver instâncias reais para os problemas TTDP em poucos segundos de tempo de computação.

Casos de aplicação

Uma grande variedade de problemas de decisão pode utilizar o TOPTW como um método de apoio à tomada de decisão. Problemas de planeamento de viagens, prestação de serviços a clientes, tendo em conta horas de abertura, são alguns dos casos de aplicação. Entre outros casos, o TOPTW também pode ser aplicado a planeamento de trabalhos de dias futuros

para trabalhadores, ao processo de recolha seletiva de resíduos sólidos urbanos (Oliveira et al., 2013) e ao transporte não urgente de doentes em veículo partilhado (Oliveira et al., 2014).

2.1.3 Capacitated Team Orienteering Problem – CTOP

O problema de orientação de equipas com restrições de capacidade foi introduzido pela primeira vez por Archetti et al. (2007) e é uma variante do TOP, onde limites de capacidade encontram-se associados aos veículos. Neste tipo de problema, um subconjunto potencial de clientes tem de ser selecionada de tal modo que a procura total dos clientes numa rota não exceda a capacidade do veículo e o comprimento total de cada rota não exceda o comprimento máximo – T_{max}. Cada veículo começa o percurso a partir do depósito, serve um determinado número de clientes e recolhe – apenas uma vez – os seus lucros associados, e retorna para o depósito.

O CTOP consiste em projetar m rotas que maximizam o lucro total recolhido pelos veículos.

Modelo matemático

O problema CTOP pode ser definido num grafo não direcionado $G = (V, A)$, onde $V = \{0, \dots, N\}$ é o conjunto de vértices e $A = \{(i,j): i,j \in V\}$ é o conjunto de arcos. O vértice 0 é conhecido como o depósito e o conjunto potencial de clientes é denotado por $V_c = V \backslash \{0\}$. Um custo não-negativo c_{ij} está associado a cada arco $(i,j) \in A$, enquanto a matriz correspondente de custo de viagens $[c_{ij}]$ é simétrica, isto é, $c_{ij} = c_{ji}$, e satisfaz a desigualdade triangular. Existe uma frota homogênea de P veículos, com capacidade máxima de carga Q, que pode operar até T_{max} unidades de tempo. Além disso, cada cliente $i \in V_c$ tem um lucro pré-definido w_i, um tempo de serviço ST_i e uma procura não nula d_i $(0 < d_i \leq Q)$.

Devido à duração da rota e restrições de capacidade dos veículos, não é possível servir todos os clientes. Assim, o conjunto V_c pode ser dividido em dois subconjuntos disjuntos, o conjunto de clientes servidos V_s e o conjunto de clientes não servidos V_u. O objetivo é determinar V_s, isto é, o subconjunto de clientes que irão ser atendidos, juntamente com a sequência correspondente das visitas, de tal forma que a recompensa total recolhida é maximizada. Cada cliente $i \in V_s$ é visitado apenas uma vez por um veículo $p \in P$. Cada rota começa e termina no depósito e a quantidade acumulada transportada por cada veículo $p \in P$

não deve exceder a capacidade de carga máxima Q. Finalmente, a distância total percorrida por cada veículo $p \in P$ é restrito por um limite T_{max}.

Dada a representação acima, o CTOP pode ser matematicamente representada como um problema de programação inteira mista, com as seguintes varáveis de decisão:

$$X_{ijp} = \begin{cases} 1, & se\ o\ arco\ (i,j) \in A\ é\ atravessado\ pelo\ veículo\ p\ \in P \\ 0, & caso\ contrário \end{cases}$$

$$Y_{ip} = \begin{cases} 1, & se\ o\ cliente\ i\ é\ visitado\ pelo\ veículo\ p \\ 0, & caso\ contrário \end{cases}$$

$$Max \sum_{i \in V_c} \sum_{p \in P} p_i * Y_{ip} \tag{25}$$

Sujeito a:

$$\sum_{p \in P} \sum_{j \in V_c} X_{0jp} = \sum_{p \in P} \sum_{i \in V_c} X_{i0p} = P \tag{26}$$

$$\sum_{p \in P} Y_{ip} \leq 1 \quad \forall i \in V_c \tag{27}$$

$$\sum_{j \in V} X_{ijp} = \sum_{j \in V} X_{jip} = Y_{ip} \quad \forall i \in V_c, \quad \forall p \in P \tag{28}$$

$$\sum_{i \in V} \sum_{j \in V} c_{ij} * X_{ijp} + \sum_{i \in V_c} ST_i * Y_{ip} \leq T_{max} \quad \forall p \in P \tag{29}$$

$$\sum_{i \in V_c} d_i * Y_{ip} \leq Q \quad \forall p \in P \tag{30}$$

$$\sum_{i \in S} \sum_{j \in S} X_{ijp} \leq |S| - 1 \quad \forall S \subseteq V_c, \quad |S| \geq 2, \quad \forall p \in P \tag{31}$$

$$Y_{ip}, X_{ijp} \in \{0,1\} \quad \forall i,j \in V, \quad \forall p \in P \tag{32}$$

A função objetivo (25) maximiza a recompensa total recolhida. A restrição (26) impõe que P veículos saem e voltam para o depósito. A restrição (27) garante que cada cliente é visitado no máximo uma vez. A restrição (28) assegura a conetividade da rota (o mesmo veículo chega e parte de um determinado cliente). As restrições (29) e (30) impõem restrições de duração da rota e de capacidade para cada veículo p. A restrição (31) garante a inexistência de sub-rotas, enquanto a restrição (32) impõe restrições binárias para as varáveis de decisão.

Abordagem de soluções

Archetti et al. (2007) propuseram um algoritmo exato e três heurísticas para resolver problemas CTOP. O algoritmo exato é uma abordagem *branch-and-price* baseada no método idealizado por Boussier et al., (2006) para o problema TOP, onde o problema principal de geração de colunas (*column generation restricted master problem* - *CGRMP*) é um conjunto de problemas de cobertura, e o sub-problema é um problema de caminho mais curto elementar com restrições de recursos. O CGRMP é resolvido usando o método simplex enquanto o sub-problema é resolvido por programação dinâmica. Este algoritmo exato pode encontrar as soluções ótimas nos casos que a capacidade e a distância máxima de percurso são pequenas. As três heurísticas propostas são um algoritmo de pesquisa em vizinhança variável (VNS), um algoritmo de pesquisa tabu que explora soluções viáveis (TSF) e um algoritmo de pesquisa tabu que explora soluções admissíveis (TSA). As vizinhanças para as duas heurísticas de pesquisa tabu são geradas por dois operadores chamado *1-move* e *swap-move*. O operador *1-move* muda um cliente de uma rota para outra e a operação *swap-move* move dois clientes de duas rotas diferentes. Dada uma solução incumbente s, a heurística de pesquisa tabu tenta escolher uma solução s' da vizinhança de s que é não-tabu ou uma nova melhor solução global em cada etapa. Em seguida, usa-se s' como solução incumbente na próxima iteração. Repete-se a heurística até que um determinado número de iterações sem melhorias é executado. No VNS, os autores utilizam dois procedimentos de salto. Em cada passo, o VNS salta da solução incumbente s para uma nova solução s', aplica-se quer TSF ou TSA em s' para obter uma melhor solução s''. Designa-se s'' como solução incumbente para a próxima iteração se s'' é melhor do que s. O VNS termina depois de um número prescrito de saltos. Resumindo, VNS chama as duas heurísticas TS e, portanto, produz as melhores soluções entre as três heurísticas, mas requer maior tempo de computação.

Luo et al. (2013) propuseram uma nova abordagem de pesquisa local baseada em *ejection pool* para o CTOP, que inclui uma componente de adaptação e um mecanismo de

diversificação. Uma *ejection pool* é uma lista de prioridade que contém os clientes não atendidos. Em cada iteração, é inserido o cliente com maior prioridade, a partir da *ejection pool* numa rota existente. Esta rota pode tornar-se inadmissível como resultado, devido a violação de capacidade ou restrições de distância máxima de percurso, ao que é realizado uma pesquisa local que tenta restaurar a admissibilidade. Se esta tentativa de restaurar a viabilidade da rota falhar, retiram-se alguns clientes da rota com o intuito de viabilizar a rota, e estes clientes são inseridos na *ejection pool* após a aplicação de uma penalidade ao seu valor de prioridade. Além disso, sempre que for detetado que o processo de pesquisa pode cair num ótimo local, redefine-se as prioridades dos clientes, perturbando a solução atual e muda-se a regra de prioridade da "*ejection pool*", na tentativa de escapar desse ótimo local. O algoritmo proposto foi testado em instâncias de *benchmark* padrão. Os resultados das experiências computacionais mostraram que este algoritmo supera todos os três algoritmos heurísticos propostos por Archetti et al. (2007) em termos de qualidade de solução e tempos de computação, e é capaz de encontrar várias novas melhores soluções.

Casos de aplicação

Archetti et al. (2009) descreveram um caso de aplicação do CTOP, em que carregadores colocam as suas procuras no serviço de transporte na web, e as transportadoras identificam essas procuras e oferecem os seus serviços para os carregadores. Normalmente, uma transportadora tem uma frota de veículos e um conjunto de clientes regulares que têm de ser servido. Se a capacidade dos veículos não é totalmente utilizada, o transportador pode querer procurar clientes locais na internet. Neste caso, o transportador tem que selecionar, dentro do conjunto de potenciais clientes, aqueles que são mais convenientes para ele.

2.1.4 Capacitated Team Orienteering Problem with Time Windows – CTOPTW

O problema de orientação de equipas com restrições de capacidade e janelas temporais é uma extensão do TOP, onde cada cliente tem uma procura e uma janela temporal associada à sua visita. Cada cliente só poderá ser visitado na determinada janela temporal e apenas uma vez. A capacidade de cada veículo é limitada. O problema CTOPTW consiste em determinar um subconjunto de clientes, de tal forma que a recompensa total recolhida seja maximizada.

Modelo matemático

O CTOPTW pode ser definido por um grafo ponderado não direcionado $G = (V, A)$, onde $V = \{v_1, \dots, v_N\}$ é o conjunto de vértices e $A = \{a_{ij}, \dots, a_{nm}\}$ é o conjunto de arcos. Cada vértice v_i tem uma procura q_i, uma janela temporal $[e_i - l_i]$, um tempo de serviço τ_i e

um peso $w_i \geq 0$. Onde τ_i pode ser interpretado como o tempo necessário para servir o vértice v_i e w_i pode ser interpretado como a recompensa obtida por servir o vértice v_i. O vértice inicial é o v_1 e o final é o v_N. Cada arco $a_{ij} \in A$, tem um peso t_{ij}, que pode ser interpretado como o tempo para um veículo se deslocar do vértice v_i para o vértice v_j. O objetivo do CTOPTW é encontrar P caminhos $P_1, P_2, \dots, P_P$ no grafo G que satisfaz as seguintes condições:

- P_i $(i = 1,2, \dots, P)$ começa no vértice v_1 e termina no vértice v_N;
- As procuras totais em cada caminho P_i $(i = 1,2, \dots, P)$ não pode ultrapassar a capacidade Q_i $(i = 1,2, \dots, P)$;
- A distância total do caminho P_i $(i = 1,2, \dots, P)$ não pode ultrapassar o limite L_i $(i = 1,2, \dots, P)$;
- Para cada vértice num caminho, o tempo de chegada satisfaz as suas restrições de janelas temporais;
- A recompensa total dos vértices em todos os caminhos P_i $(i = 1,2, \dots, P)$ é maximizada.

O CTOPTW pode ser formulado como um problema de programação linear inteira com os seguintes parâmetros e variáveis de decisão:

Parâmetros:

τ_i é o tempo de serviço no vértice v_i, onde $\tau_1 = \tau_n = 0$;

Q_p é a capacidade do veículo p;

q_i é a procura do vértice v_i;

$[e_i - l_i]$ é a janela temporal associada ao vértice v_i;

L_k é o tempo limite para o veículo k chegar ao vértice v_n.

Variáveis de decisão:

$$X_{ip} = \begin{cases} 1, & se\ o\ vértice\ i\ é\ visitado\ no\ caminho\ p \\ 0, & caso\ contrário \end{cases}$$

$$Y_{ijp} = \begin{cases} 1, & se\ o\ vértice\ j\ é\ visitado\ após\ o\ vértice\ i\ no\ caminho\ p \\ 0, & caso\ contrário \end{cases}$$

$$S_{ip} = inicio\ do\ serviço\ no\ vértice\ i\ no\ caminho\ p$$

$Se\ o\ vértice\ i\ não\ é\ visitado\ no\ caminho\ p, então\ S_{ip} = 0.$

$M - uma\ grande\ constante$

Função objetivo:

$$\max \sum_{i=2}^{N-1} \sum_{p=1}^{P} w_i * X_{ip} \tag{33}$$

Sujeito a:

$$\sum_{j=2}^{N} Y_{1jp} = 1 \quad \forall p = 1, \dots, P \tag{34}$$

$$\sum_{i=1}^{N-1} Y_{iNp} = 1 \quad \forall p = 1, \dots, P \tag{35}$$

$$\sum_{i=1}^{N-1} Y_{ijp} = \sum_{l=2}^{N} Y_{jlp} = X_{jp} \quad \forall j = 2, \dots, N-1 \quad \forall p = 1, \dots, P \tag{36}$$

$$\sum_{p=1}^{P} X_{ip} \leq 1 \quad \forall i = 2, \dots, N-1 \tag{37}$$

$$\sum_{i=2}^{N-1} q_i * X_{ip} \leq Q_p \quad \forall p = 1, \dots, P \tag{38}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} Y_{ijp} * \left(t_{ij} + \tau_i\right) \leq L_p \quad \forall p = 1, \dots, P \tag{39}$$

$$S_{1p} = 0 \quad \forall p = 1, \dots, P \tag{40}$$

$$S_{ip} + \tau_i + t_{ij} - S_{jp} \leq M * \left(1 - Y_{ijp}\right) \quad \forall i, j = 1,2, \dots, N \quad \forall p = 1, \dots, P \tag{41}$$

$$e_j * X_{jp} \leq S_{jp} \leq l_j * X_{jp}, \quad \forall j = 2, \dots, N-1 \quad \forall p = 1, \dots, P \tag{42}$$

$$S_{np} \leq L_p, \quad \forall p = 1, \dots, P \tag{43}$$

$$X_{ip}, Y_{ijp} \in \{0,1\}, \quad \forall i, j = 1, \dots, N \quad \forall p = 1, \dots, P \tag{44}$$

$$S_{ip} \geq 0, \quad \forall i = 1, \dots, N \quad \forall p = 1, \dots, P \tag{45}$$

A função objetivo (33) maximiza a recompensa total recolhida. As restrições (34) e (35) garantem que cada rota se inicia no vértice v_1 (34) e termina no vértice v_n (35). A restrição (36) garante que se um veículo no caminho p entrar no vértice v_j, este deve sair do vértice v_j. A restrição (37) significa que para cada vértice v_i, existe no máximo um veículo para o servir. A restrição de capacidade dos veículos é garantida pela restrição (38). A restrição (39) garante que o tempo de viagem em cada caminho p não seja maior do que L_p, inclui o tempo de serviço e de deslocação. A restrição (40) significa que para todos os caminhos, o tempo inicial de partida é zero. A restrição (41) indica a relação do tempo de chegada entre dois vértices visitados consecutivamente numa viagem. A garantia de que as janelas temporais são respeitadas é dada pela restrição (42). A restrição (43) garante que o tempo de chegada do veículo no caminho p ao vértice v_n não ultrapassa o tempo limite L_p. A restrição (44) impõe restrições binárias para as varáveis de decisão. A restrição (45) indica que S_{ip} são variáveis contínuas não negativas.

Abordagem de soluções

Li & Hu, (2011) referem-se ao problema CTOPTW pela primeira vez na literatura como *Team Orienteering Problem with Capacity constraint and Time windows – TOPCTW* e apresentaram um modelo de programação linear inteira baseado na teoria de fluxo de rede para o resolver. Resolvendo a programação linear inteira, pode-se obter a solução ótima global para o TOPCTW. Este modelo e algoritmo são as primeiras abordagens para o TOPCTW. Soluções de boa qualidade podem ser obtidas por esta abordagem para instâncias de tamanhos pequenos (menor do que 30 vértices).

Casos de aplicação

Li & Hu, (2011) aplicaram o CTOPTW no caso de transporte de mercadorias, de um centro de distribuição para os respetivos clientes.

O CTOPTW pode ser aplicado também ao processo de recolha seletiva de resíduos sólidos urbanos, ao transporte não urgente de doentes em veículo partilhado, e entre outros casos, à recolha de leite cru, conforme será ilustrado neste trabalho.

2.2 Algoritmos genéticos – AGs

Os algoritmos genéticos são métodos de pesquisas probabilísticos para otimização global, inspirados nos princípios de seleção natural e da genética propostos por Charles Darwin no seu livro "*On the Origin of Species by Means of Natural Selection*". Os AGs foram desenvolvidos por John Holland e pelos seus alunos da Universidade de Michigan durante os anos de 1960 (Holland, 1994). Os AGs utilizam uma estratégia de pesquisa paralela e estruturada, embora aleatória, no espaço de soluções, direcionada à busca de soluções de alta qualidade (idealmente a solução ótima), ou seja, soluções que apresentam valores extremos (máximos ou mínimos) da função objetivo.

Adotaram-se um conjunto de designações para qualificar situações e entidades relacionados com o modo de funcionamento dos AGs. O conjunto das soluções processadas é designado por *população* e a sequência das várias populações reflete a *evolução* ao longo do tempo. Os elementos constituintes de uma população são designados por *cromossomas*. Os *genes* são elementos básicos dos cromossomas, podendo ser considerados como caraterísticas elementares de uma solução e cada um deles pode tomar um valor de entre um conjunto de possibilidades, denominadas de *alelos*. Cada iteração do processo de otimização é designada por *geração*.

A implementação de um algoritmo genético inicia-se com uma população de cromossomas, tipicamente aleatória. As populações são submetidas ao processo de evolução ao longo de sucessivas gerações, conforme demonstrado a seguir na Figura 2. O processo de evolução é constituído pelas seguintes etapas:

- **Avaliação dos indivíduos –** avalia-se a aptidão dos indivíduos, ou seja, mede-se quão boa é a solução codificada por um indivíduo, com base no valor da função objetivo;
- **Seleção –** indivíduos são selecionados para o cruzamento, para que depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos.
- **Cruzamento ou Recombinação –** é considerado o principal responsável pela pesquisa efetuada pelo algoritmo genético. Nesta etapa, um operador estocástico de recombinação efetua troca de material genético entre dois indivíduos progenitores,

dando origem a dois indivíduos descendentes, formados por sequências genéticas parciais de cada progenitor.

- **Mutação –** é um processo aleatório, com o objetivo de manter um nível de diversidade adequado na população e garantir que alelos que eventualmente desapareçam tenham a possibilidade de reaparecer. Nesta etapa, um operador unário de mutação altera ligeiramente algumas das caraterísticas dos indivíduos resultantes do cruzamento, acrescendo assim variedade à população.
- **Atualização –** os indivíduos criados nesta geração são inseridos na população.
- **Avaliação do critério de paragem –** verifica se o critério de paragem foi atingido, terminando a execução do algoritmo em caso positivo e iterar para a próxima geração em caso negativo.

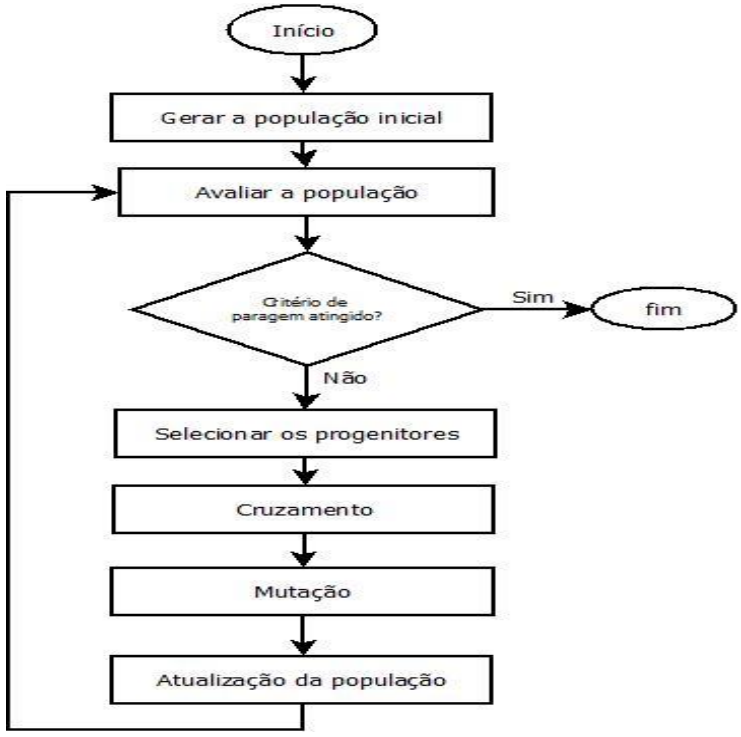

Figura 2 - Estrutura genérica de um algoritmo genético

É importante analisar como os parâmetros do AG influenciam o seu comportamento e desempenho, para que se possa estabelece-los de acordo com as necessidades do problema e dos recursos disponíveis.

2.2.1 Parâmetros Genéticos

Tamanho da população

Este parâmetro afeta o desempenho global e a eficiência dos AGs. Uma população pequena pode diminuir o desempenho do algoritmo, pois fornece uma pequena cobertura do espaço de soluções para o problema. Uma população grande geralmente fornece uma cobertura do espaço de soluções representativa do domínio do problema, além de prevenir convergências prematuras para ótimos locais. No entanto, grandes populações requerem mais recursos computacionais e maior tempo de computação.

Taxa de cruzamento

É a frequência com que o cruzamento ou recombinação é feita. Quanto maior for essa taxa, mais rapidamente novos indivíduos serão inseridos na população. Se essa taxa for muito alta, indivíduos com boas aptidões poderão ser retiradas mais rapidamente que a capacidade de a recombinação criar indivíduos melhores, causando um desequilíbrio genético na população. Por outro lado, se a taxa for muito baixa, a pesquisa pode estagnar-se.

Taxa de mutação

É a frequência com que os genes sofrerão mutação. Se essa taxa for nula, os descendentes resultantes da recombinação não sofrem nenhuma alteração. Uma taxa baixa previne a estagnação da pesquisa em sub-regiões do espaço de soluções e possibilita que qualquer ponto desse espaço seja atingido. Por outro lado, uma taxa muito alta torna a pesquisa essencialmente aleatória.

Intervalo de geração

É a percentagem dos cromossomas que serão substituídos a cada geração. Valores muito altos implicam que grande parte dos indivíduos serão substituídos, podendo levar a perda de indivíduos de alta aptidão. Já valores muito baixos podem tornar o algoritmo muito lento.

Elitismo

Trata-se da repetição dos indivíduos com melhores aptidões na próxima geração a fim de evitar que todos os indivíduos com boas aptidões sejam alterados pela recombinação e pela mutação, permitindo que o algoritmo convirja mais rapidamente para uma solução. Deve-se ter em atenção que quanto maior for o número de indivíduos mantidos, maior é a probabilidade de o algoritmo convergir-se para um ótimo local.

Critério de paragem

Diferentes critérios de paragem podem ser utilizados para terminar a execução de um AG. Os critérios mais comuns são o número limite de gerações atingidos, descoberta de uma solução com a qualidade pretendida e a inexistência de melhorias durante um determinado período de tempo de execução.

2.2.2 Casos de aplicação

Segundo Carvalho et al. (2003), os algoritmos genéticos têm sido aplicado com sucesso a uma grande variedade de problemas de otimização, como é o caso dos problemas de escalonamento, afetação de tarefas em sistemas multiprocessadores, controle de irrigação, controle de sistemas dinâmicos, desenvolvimento de controladores *fuzzy* para robôs móveis, otimização de rotas e entre outros.

2.3 *Framework* BRKGA

A *framework* BRKGA é uma biblioteca multiplataforma implementada em linguagem de programação C++ para a estrutura algorítmica de algoritmos genéticos de chave aleatória tendenciosa (*Biased Random-Key Genetic Algorithms* – BRKGA). Foi desenvolvido e apresentado pelos investigadores Maurício Resende e Rodrigo Toso (Toso & Resende, 2014).

Essa biblioteca lida automaticamente com a grande parte dos módulos independentes do problema, incluindo a gestão da população e a dinâmica evolutiva, deixando ao utilizador a tarefa de implementar um procedimento dependente do problema para converter um vetor de chaves aleatórias em uma solução para o problema de otimização subjacente, além de permitir decodificar múltiplas populações simultaneamente. Na Figura 3 é ilustrada a estrutura da *framework* BRKGA.

A ferramenta BRKGA foi utilizado para a resolução de problemas de otimização com algum sucesso no que diz respeito à qualidade das soluções, como são os casos apresentados por Resende et al., (2012) na resolução de problemas de cobertura tripla de Steiner, e a resolução de problemas de otimização combinatória (Gonçalves & Resende, 2011).

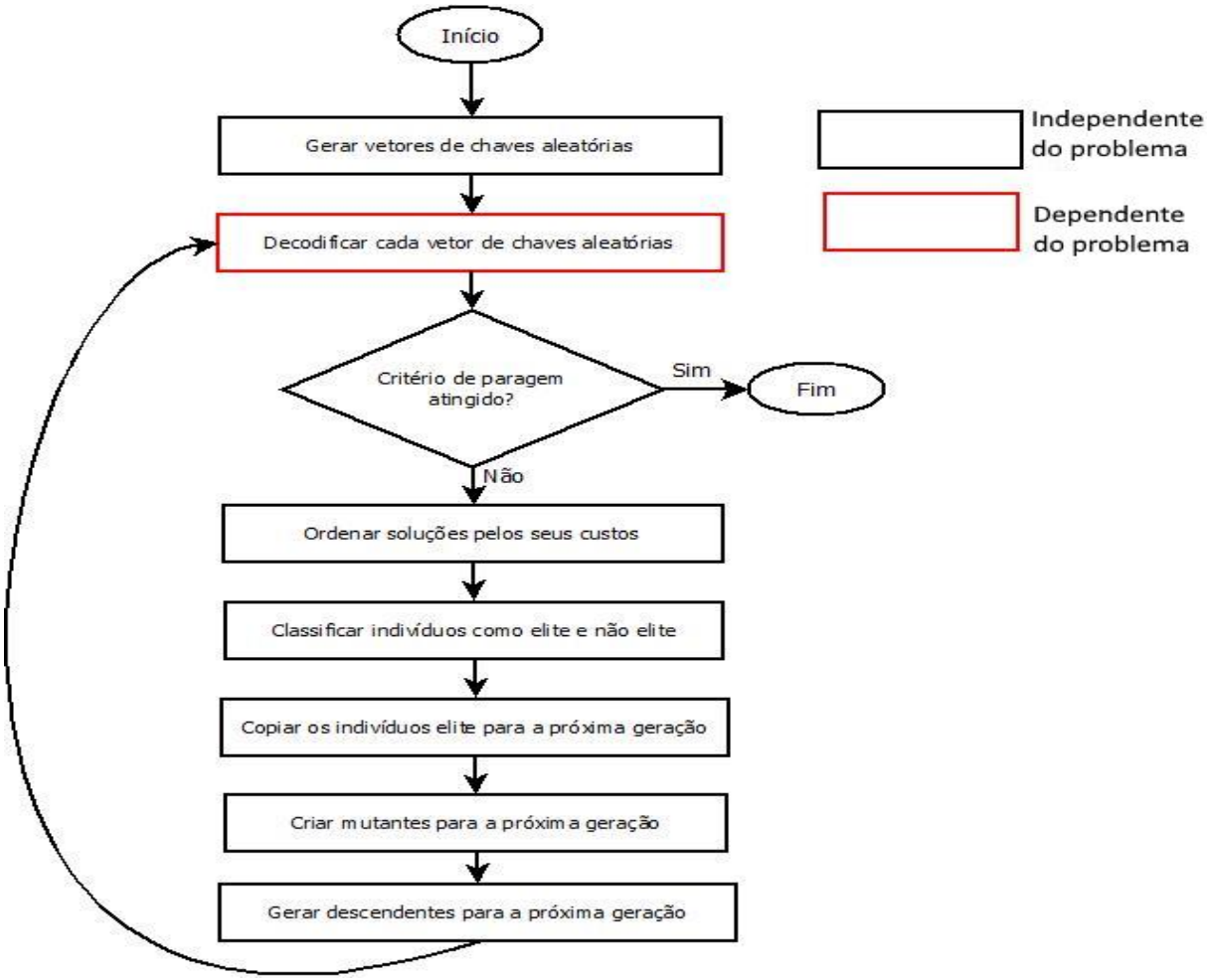

Figura 3 - Estrutura da framework BRKGA

3. MÉTODOS

Neste capítulo descrever-se-á de forma detalhada o problema da recolha do leite, bem como as metodologias adotadas para a resolução do problema em estudo. É também descrita a modelação do problema real em um modelo matemático, ou seja, como se passa do problema real da recolha do leite cru e se chega ao modelo do CTOPTW.

3.1 Problema da recolha e transporte do leite cru

Segundo a Associação dos Produtores de Leite de Portugal – APROLEP (2015), o setor leiteiro tem sofrido uma crise económica por vários fatores, o que torna imperativo que todos os organismos envolvidos neste setor diminuam os seus custos operacionais. Os custos de uma empresa são compostos na sua maioria por custos logísticos, cujo componente mais dispendioso e por isso mais importante é o transporte, sendo por isso importante otimizá-lo e reduzir os seus custos.

No caso das empresas do setor leiteiro é importante racionalizar a recolha e o transporte do leite cru dos produtores até à indústria, para beneficiar toda a cadeia do leite. Para poder otimizar o transporte do leite cru é necessário saber como é feito o seu transporte.

3.1.1 Transporte do leite cru

O transporte do leite cru pode ser feito em carros-cisterna ou em latões, sendo preferível o transporte em carros-cisterna uma vez que se consegue uma maior manutenção da qualidade do leite e menor custo de transporte (quando as rotas de recolhas não são muito extensas). Durante o transporte, o leite cru deve ser mantido a uma temperatura não superior a 10ºC, porque se ficar durante doze horas à temperatura ambiente (10ºC – 12ºC) sem adição de qualquer conservante, fica ácido e sem qualidade. No entanto os transportadores podem não cumprir os requisitos da temperatura, desde que o leite cru seja recolhido nas duas horas seguintes à ordenha ou se for necessária uma temperatura mais elevada por razões tecnológicas ligadas ao fabrico de determinados produtos lácteos e desde que a autoridade competente o autorize.

Segundo Lemos (2013), é indispensável a criação de postos de receção do leite cru, onde a indústria o possa recolher, uma vez que os locais de produção de leite cru são dispersos e a recolha individual se torna cada vez mais cara, e em certos casos é impraticável

o uso de refrigeração individual. Os postos de receção teriam que estar perto dos produtores de leite cru e com fácil acesso para os veículos da indústria.

3.2 Modelação do problema da recolha do leite

O leite cru tem associado várias restrições, nomeadamente as restrições de temperatura e armazenamento, o que torna o uso de refrigeração individual muitas vezes impraticável. Portanto, grande parte das pequenas explorações leiteiras transferem o leite produzido para unidades de processamento para armazenamento e transformação. A transferência do leite é feita através dos transportes rodoviários, o que se enquadra numa rede de recolha.

Nesta rede de recolha existem vários veículos disponíveis que fazem a recolha com o objetivo de ter o maior lucro possível por recolha completa, tendo em conta as suas capacidades e a quantidade de leite a ser carregado em cada cliente, bem como as restrições de janelas temporais associadas aos mesmos.

Tendo em conta as caraterísticas da rede de recolha pode-se construir um modelo matemático que represente o problema real da recolha de leite cru, com base no problema de orientação de equipas com restrições de capacidades e janelas temporais, conforme o modelo apresentado na seção 2.1.4.

3.3 Metodologias e estratégias adotadas

Nesta seção serão abordados as principais estratégias e métodos adotados neste trabalho.

3.3.1 Métodos de alocação dos vértices

Tendo em conta que o principal objetivo deste problema da recolha do leite é encontrar um subconjunto de clientes (vértices) a serem visitados de modo a maximizar o lucro total da frota, então o desafio principal passa por definir estratégias e algoritmos de alocação dos clientes aos veículos de recolha que dê bons resultados.

A seguir são descritos os três métodos de alocação aplicados neste trabalho:

- **Série –** aloca-se os vértices a um veículo até que este não tenha mais capacidade ou horas de trabalho disponíveis e passa-se para o próximo veículo repetindo o processo

até que todos os veículos fiquem cheios ou não tenha mais vértices disponíveis para alocar. A Figura 4 demonstra o funcionamento da alocação em série.

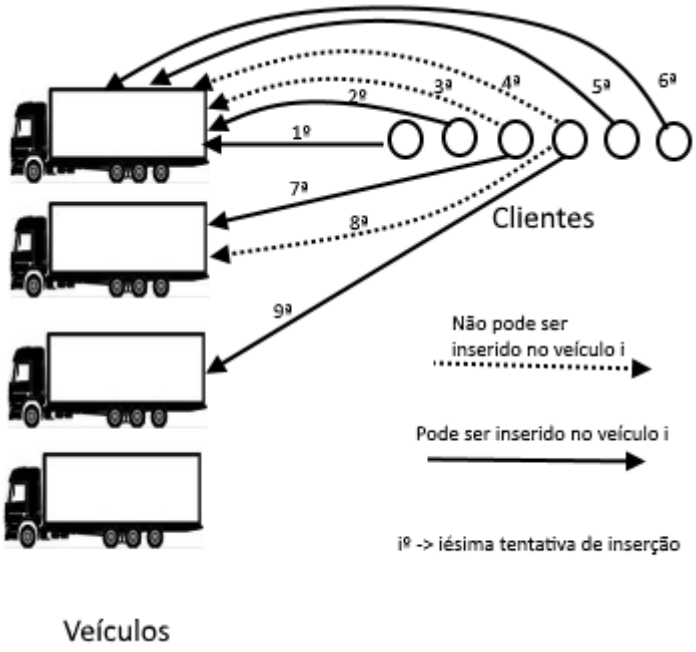

Figura 4 - Alocação em série

- **Paralelo –** aloca-se os vértices aos veículos onde couberem, ou seja, percorre-se o conjunto de veículos tentando encontrar o que possa atender às condições impostas pelo vértice atual. Na Figura 5 encontra-se representada o funcionamento deste método.

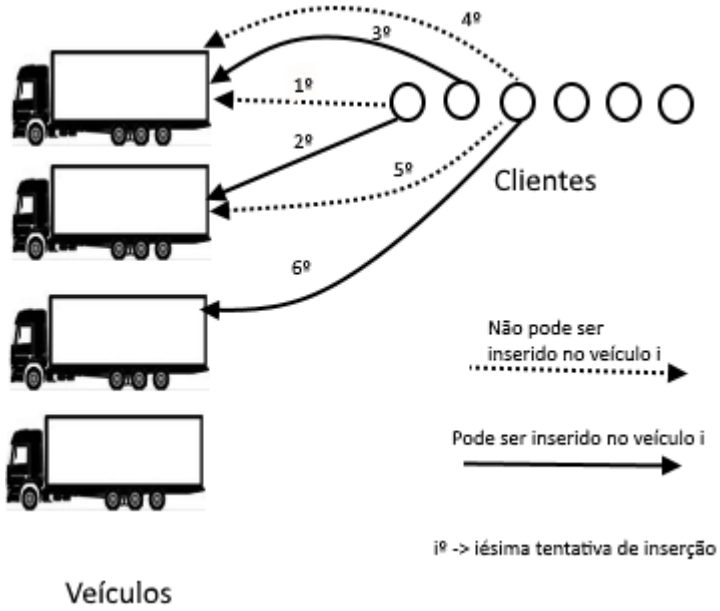

Figura 5 - Alocação em paralelo

- **Menor custo –** os vértices são alocados em série, sem uma ordem específica, mas sim nas posições onde representam o menor custo para o veículo em que está alocado. A Figura 6 representa o funcionamento deste método.

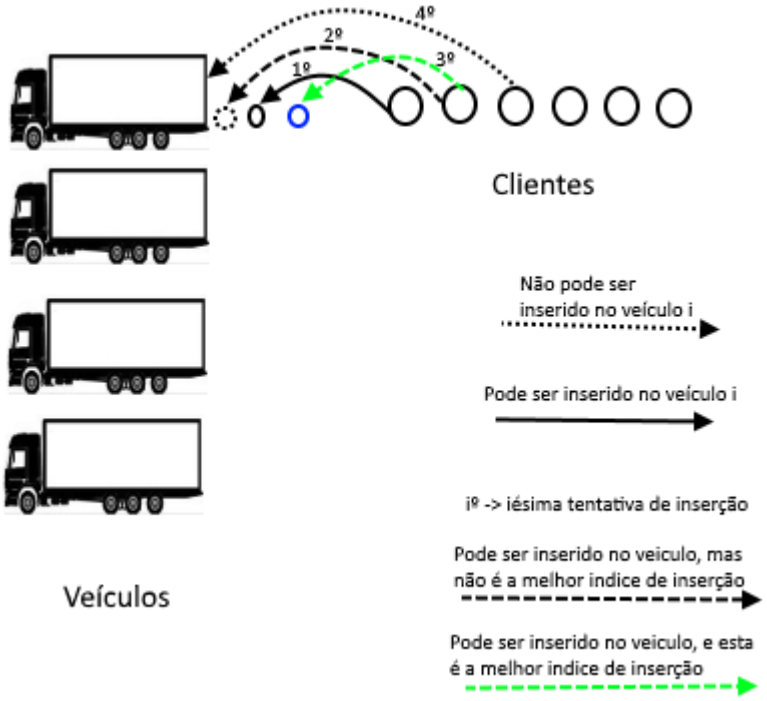

Figura 6 - Alocação de menor custo

3.3.2 Framework para o desenvolvimento dos algoritmos

Com objetivo de focar no desenvolvimento dos algoritmos de inserção dos vértices e não no desenvolvimento de base dos algoritmos genéticos para a gerência da população e da dinâmica evolutiva, optou-se por utilizar a *framework* BRKGA (seção 2.3) para esse fim. Mas uma vez que essa *framework* foi desenvolvida para problemas de minimização, foram necessários alguns reajustes para adaptá-la a problemas maximização.

3.3.3 Tipos de instâncias

Com o intuito de gerar instâncias que possam representar as mais diversas situações do mundo real no que diz respeito a localização dos vértices bem como o deslocamento entre eles, optou-se por utilizar instâncias de dois tipos, a seguir descritos:

- **Euclidiano –** são dados onde as distâncias são representadas numa matriz simétrica e respeitam a desigualdade triangular. Para fazer respeitar a desigualdade triangular, optou-se por aplicar o algoritmo de Floyd às distâncias entre os pontos do grafo.
- **Assimétrico –** neste tipo de dados as distâncias são representadas por uma matriz assimétrica e podem não respeitar a desigualdade triangular. Os dados assimétricos podem representar ruas urbanas onde ir do ponto A para o ponto B pode ser diferente de ir de B para A devido a ruas de via única por exemplo.

3.4 Desenvolvimento do *software*

Neste capítulo serão abordadas as diversas etapas do desenvolvimento do *software* criado para realizar testes computacionais aos algoritmos desenvolvidos e facilitar a criação de instâncias para os mesmos. O *software* desenvolvido no âmbito desta dissertação foi desenvolvido em linguagem C++ usando a *framework* BRKGA, o IDE Netbeans 8.1 e o Qt Creator 3.6.1.

A seguir são descritas as etapas do desenvolvimento do *software*.

3.4.1 Esquema dos ficheiros de instâncias

Nesta etapa desenvolveu-se um modelo intuitivo e de fácil elaboração para os ficheiros de instâncias. Este modelo encontra-se representado na Figura 7.

```
# Este ficheiro foi desenvolvido no âmbito da dissertação de Mestrado Integrado em Engenharia e Gestão Industrial
# pelo aluno Edgar Wchua Pires Guilherme, para instâncias do problema CTOPTW.
# O cardinal '#' é usado para fazer uma linha de comentário.
# Duração do serviço para a origem é 00:00 por padrão. As informacões de Time são representados por: hh:mm
# Este ficheiro tem o seguinte template:
#
# PolulationSize = xxx
# Evolutions = xxx
# NumClients = xxx
# NumVehicles = xxx
#
# Origin = id; name; x; y; score; capacity; serviceDuration; openTime; closeTime;
# Deposit = id; name; x; y; score; capacity; serviceDuration; openTime; closeTime;
#
#Vertices
#{
#       id; name; x; y; score; capacity; serviceDuration; openTime; closeTime;
#   ... (others vertices)
#}
#
#Vehicles
#{
#       id; name; maxCapacity; origin_id; deposit_id; departureTime; deadlineTime;
#       ... (others vehicles)
#}
#
#Travels
#{
#       id; time_orig0; time_orig1; time_orig2; ...;time_origN;
#       id; time_dep0; time_dep1; time_dep2; ...; time_depN;
#       id; time_00; time_01; time_02; ...; time_0N;
#       ...
#       id; time_N0; time_N1; time_N2; ...; time_NN;
#}
```

Figura 7 - Modelo do ficheiro de instâncias

Como se pode verificar na Figura 7, é necessário introduzir o tamanho da população, o número de evoluções ou gerações, o número de clientes e o número de veículo. A seguir deve-se introduzir as informações relativamente à origem, ao depósito, aos vértices (clientes) e aos veículos. Após isso, introduz-se as informações relativamente ao tempo de viagem entre todos os pontos do grafo.

3.4.2 Algoritmos desenvolvidos

Desenvolveu-se quatro principais algoritmos nesta dissertação, três dos quais são algoritmos de alocação dos vértices e um para corrigir os tempos de viagens entre os pontos do grafo.

Os três algoritmos desenvolvidos para a alocação dos vértices foram descritos na seção 3.3.1 e nesta seção apenas apresentar-se-á os pseudo-códigos para cada um deles. Nas Figuras Figura 8, Figura 9 e Figura 10 são apresentados os pseudo-códigos da alocação em série, em paralelo e de menor custo respetivamente.

```
Função AlocacaoEmSerie(indices Inteiro[N]) Veiculo[V]

    Var idsInseridos Inteiro[]
    Var veiculos Veiculo[V]

    veiculos = inicializarVeiculos()

    Para veiculo em veiculos
        Se indices == Vazio Então
            FimCiclo
        FimSe

        Para i = 0 até N-1
            Vertice vertice = instancia.clientes[indices[i]]

            Se idsInseridos != Vazio e vertice.id em idsInseridos Então
                ContinuarCiclo
            FimSe

            Se podeInserir(vertice, veiculo)
                veiculo.adicionar(vertice)
                idsInseridos.adicionar(vertice.id)
                indices.remover(i)
            FimSe
        FimPara
    FimPara

    Retorna veiculos
FimFunção
```

Figura 8 - Pseudo-código da alocação em série

```
Função AlocacaoEmParalelo(indices Inteiro[N]) Veiculo[V]

    Var idsInseridos Inteiro[]
    Var veiculos Veiculo[V]

    veiculos = inicializarVeiculos()

    Para i = 0 até N-1
        Vertice vertice = instancia.clientes[indices[i]]

        Se idsInseridos != Vazio e vertice.id em idsInseridos Então
            ContinuarCiclo
        FimSe

        Para veiculo em veiculos
            Se podeInserir(vertice, veiculo)
                veiculo.adicionar(vertice)
                idsInseridos.adicionar(vertice.id)
                FimCiclo
            FimSe
        FimPara
    FimPara

    Retorna veiculos

FimFunção
```

Figura 9 - Pseudo-código da alocação em paralelo

```
Função AlocacaoMenorCusto(indices Inteiro[N]) Veiculo[V]
    Var idsInseridos Inteiro[]
    Var veiculos Veiculo[V] = inicializarVeiculos()
    Para veiculo em veiculos
        Se indices == Vazio Então
            FimCiclo
        FimSe
        Para i = 0 até N-1
            Var vertice Vertice = instancia.clientes[indices[i]]
            Se idsInseridos != Vazio e vertice.id em idsInseridos Então
                ContinuarCiclo
            FimSe
            Var adicionado Booleano = Falso
            Var tamanhoRota = veiculo.rota.tamanho

            Se tamanhoRota == 0 Então
                Se podeAdicionarNoTopo(vertice, veiculo) Então
                    veiculo.adicionarTopo(vertice)
                    adicionado = Verdadeiro
                FimSe
            Senão
                # Testar qual a melhor índice
                Var aux Par<Veiculo, Inteiro>[]
                Para j = 0 até tamanhoRota
                    Var v Veiculo = veiculo
                    Se v.podeAdicionarNoIndice(vertice, j, v) Então
                        aux.adicionar(Par<>(v, j))
                    FimSe
                FimPara
                Se aux != Vazio Então
                    Var p Par<Veiculo, Inteiro> = aux[0]
                    Para a em aux
                        Se p.first.tempoTotal > a.first.tempoTotal Então
                            p = a
                        FimSe
                    FimPara
                    veiculo.adicionarNoIndice(vertice, p.second)
                    adicionado = Verdadeiro
                FimSe
            FimSe
            Se adicionado == Verdadeiro Então
                idsInseridos.adicionar(vertice.id)
                indices.remover(i)
            FimSe
        FimPara
    FimPara
    Retorna veiculos
FimFunção
```

Figura 10 - Pseudo-código da alocação de menor custo

Para obter instâncias euclidianas respeitando a simetria nas distâncias e a triangularidade entre os vértices aplicou-se o algoritmo de Floyd. O algoritmo de Floyd é um algoritmo que resolve o problema de caminho mais curto entre todos os pares de vértices em um grafo orientado. A Figura 11 representa um pseudo-código do algoritmo.

```
Função AlgoritmoFloyd(grafo Inteiro[1..N, 1..N]) Inteiro[,]

        Var distancias Inteiro[1..N, 1..N] = grafo
        Var predecessores Inteiro[1..N, 1..N]

        #inicializar os predecessores

        Para i = 1 até N
                Para j = 1 até N
                        Se distancias[i, j] < Infinito Então
                                predecessores[i, j] = i
                        FimSe
                FimPara
        FimPara

        # corrigir distâncias e predecessores

        Para k = 1 até N
                Para i = 1 até N
                        Para j = 1 até N
                                Se distancias[i,j] > distancias[i,k] + distancias[k,j] Então
                                        distancias[i,j] = distancias[i,k] + distancias[k,j]
                                        predecessores[i, j] = predecessores[k, j]
                                FimSe
                        FimPara
                FimPara
        FimPara

        Retorna distancias

FimFunção
```

Figura 11 - Pseudo-código do algoritmo de Floyd

3.4.3 Funcionamento do *software*

O software desenvolvido no âmbito deste trabalho tem algumas funcionalidades básicas que permitem fazer testes a instâncias do problema CTOPTW.

Uma das funcionalidades é a de geração de instâncias aleatórias em que o usuário pode escolher o tipo de instância a gerar, ou seja, se pretende uma instância euclidiana ou assimétrica dependendo das suas necessidades bem como os dados requeridos pelo modelo adotado para o ficheiro de dados, anteriormente citados na seção 3.4.1. A Figura 12 mostra o painel *Random Data* desenvolvida para esta funcionalidade, onde também permite guardar as instâncias em ficheiros para a base de dados e para a realização de testes posteriormente.

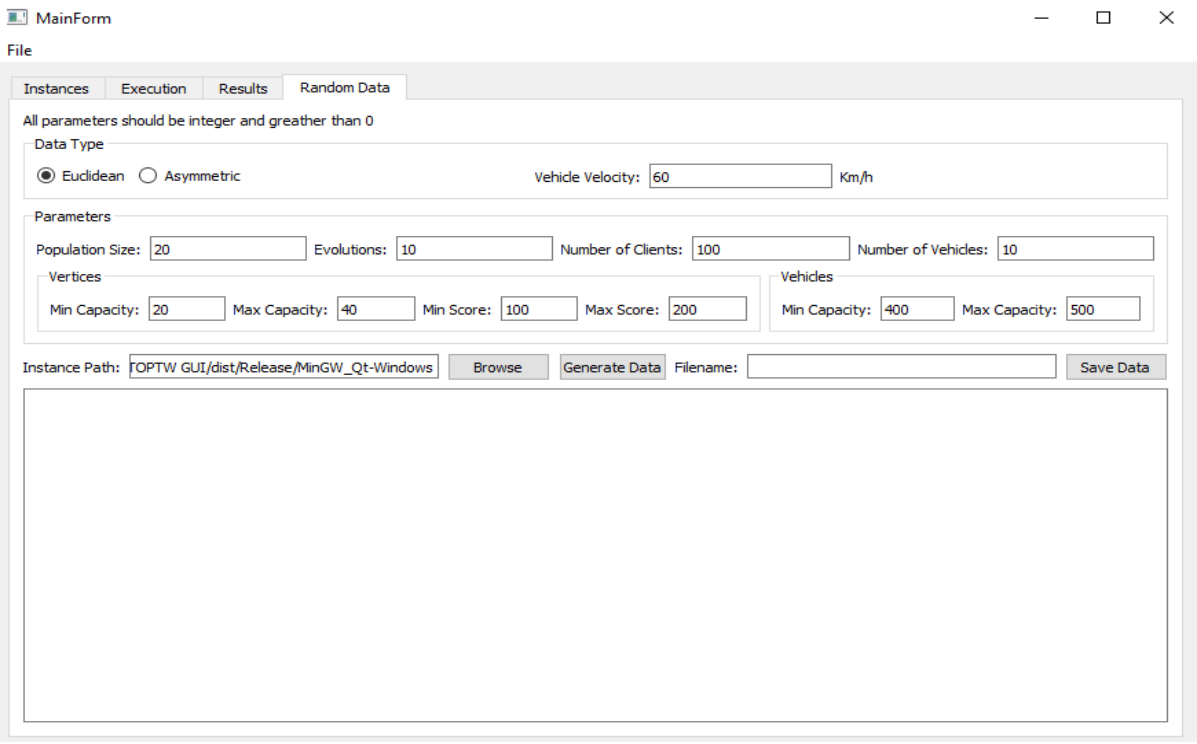

Figura 12 - Painel de geração de instâncias aleatórias

Uma vez que o usuário possui os ficheiros de instâncias pode-se, através do painel *Instances*, carregar esses ficheiros, ver as informações neles contidas e depois proceder aos testes. A Figura 13 apresenta a interface do painel *Instances*.

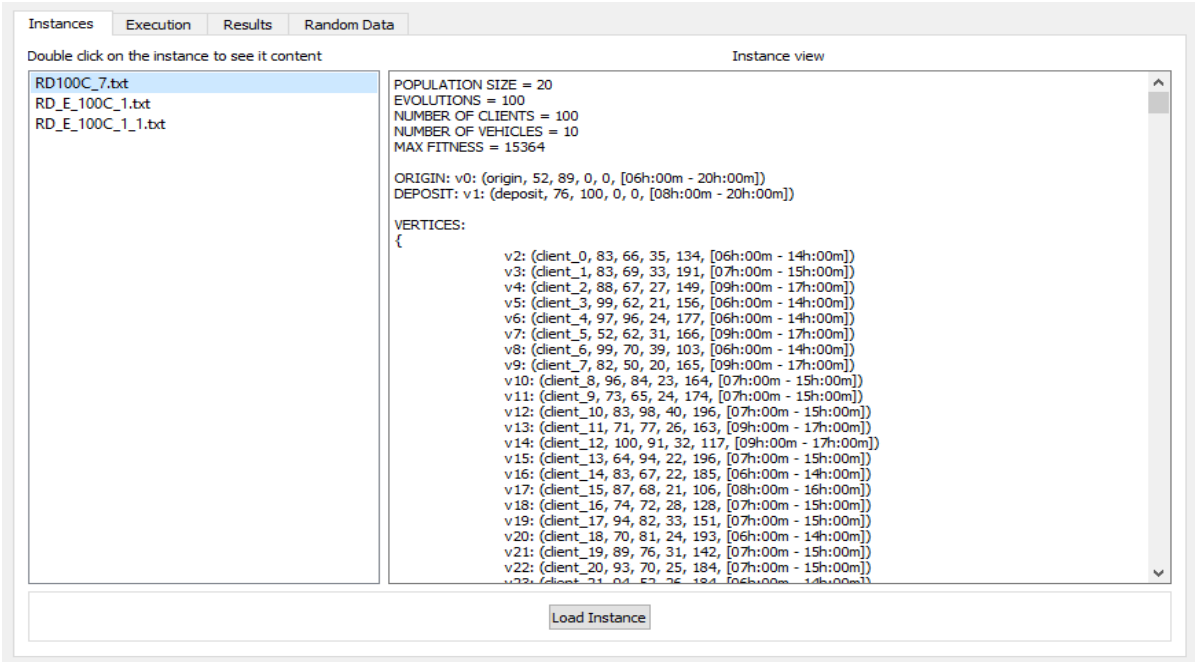

Figura 13 - Painel de carregamento e visualização de instâncias

Para a realização dos testes às instâncias o usuário deve usar o painel *Execution*, onde o usuário escolhe a instância desejada, o método de alocação dos vértices e os respetivos parâmetros do BRKGA nomeadamente as percentagens da população elite, dos mutantes e a probabilidade de um cromossoma filho herdar caraterísticas do pai elite bem como o numero de populações independentes, o número máximo de *threads* independentes e o número máximo de evoluções permitidas sem que haja melhorias na solução.

Com o intuito de facilitar a realização dos testes, o usuário pode optar por utilizar a opção de testar uma instância (*One Instance*) para executar três testes automáticos sobre uma instância (um teste para cada método de inserção), a opção de testar todas as instâncias (*All Instance*) presentes na lista de instâncias a executar, de modo a economizar tempo gasto na realização das experiências. Pode ainda optar pelas opções de salvar automaticamente após a realização de um teste e a de realizar testes sequências, ou seja, o software realiza automaticamente um determinado número de *runs* sobre uma mesma instância. Na Figura 14 encontra-se apresentado o painel *Execution*.

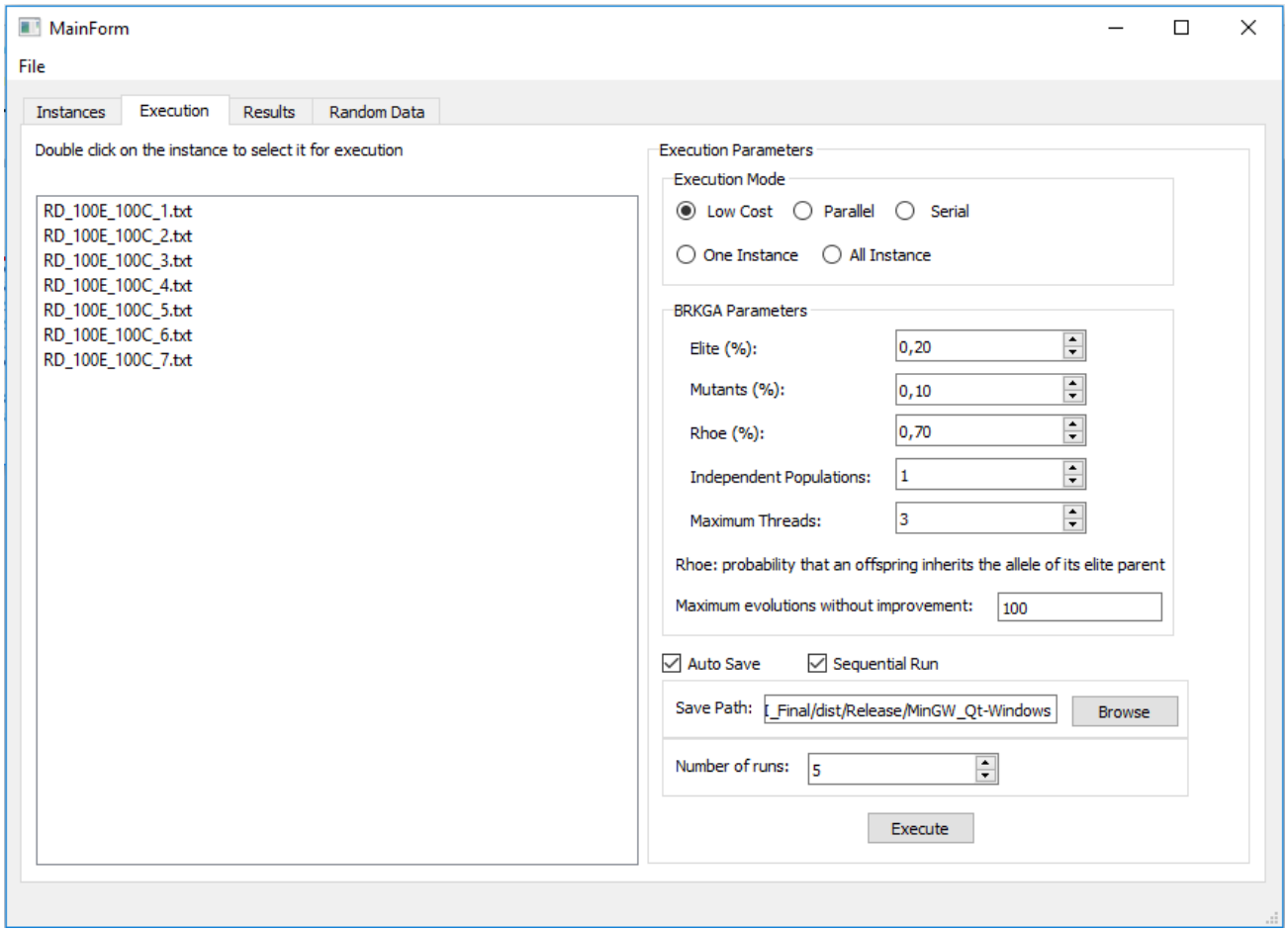

Figura 14 - Painel de realização de testes

O usuário pode verificar o resultado de um teste no painel *Results*, onde se desejar pode guardar o respetivo resultado num ficheiro texto (.txt), conforme indica a Figura 15.

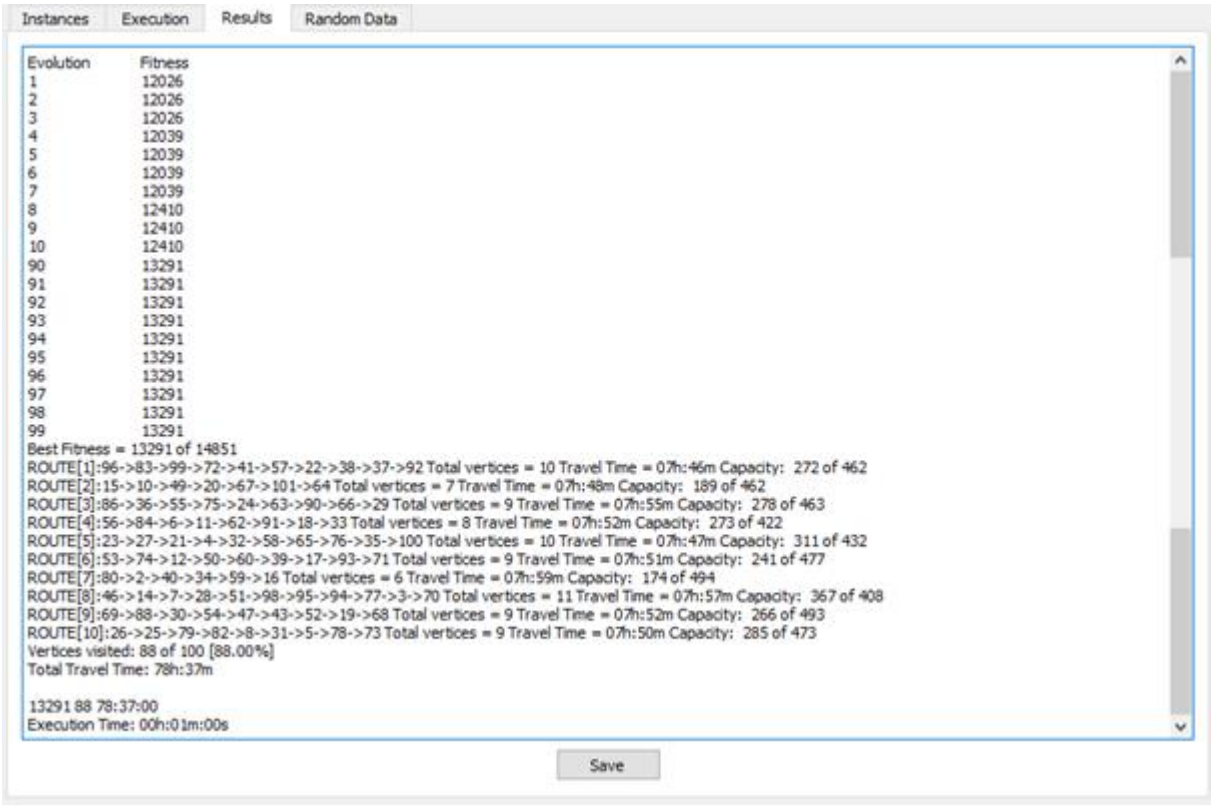

Figura 15 - Painel de resultados

Os ficheiros de resultados contêm informações relativamente ao método de alocação usado no respetivo teste, o tamanho da população, o numero de evoluções, o número máximo de *fitness* para a respetiva instância e o *fitness* final obtido no teste bem como os *fitnesses* obtidos em cada geração para uma posterior análise gráfica da evolução das soluções, por exemplo, no Excel. Conforme indica a Figura 16, para cada rota ou veículo é indicada a sequência dos vértices visitados, o número total de clientes visitados bem como a capacidade total ocupada e o tempo total de viagem realizado pelo mesmo.

Também se encontra disponível informações relativamente ao número total de clientes visitados e a respetiva percentagem bem como o tempo total de viagem gasto pela frota e o tempo de execução consumido pelo CPU.

```
Allocation Mode = Parallel
Population Size = 20
Evolutions = 100
Evolution       Fitness
1       11869
2       11869
3       11956
4       11956
5       11956
6       11956
7       11956
8       11956
9       11956
10      11956
11      11956
12      12087
13      12087
14      12087
15      12301
90      13292
91      13321
92      13321
93      13450
94      13450
95      13450
96      13450
97      13450
98      13450
99      13450


Best Fitness = 13450 of 14851
ROUTE[1]:56->83->53->84->70->87->101->91->42 Total vertices = 9 Travel Time = 07h:47m Capacity:  286 of 462
ROUTE[2]:77->72->48->40->27->14->7->99->39->73->71 Total vertices = 11 Travel Time = 07h:53m Capacity:  337 of 462
ROUTE[3]:68->50->95->28->62->88->33->55->30 Total vertices = 9 Travel Time = 08h:00m Capacity:  293 of 463
ROUTE[4]:74->46->37->21->90->51->16->80->86->17 Total vertices = 10 Travel Time = 07h:58m Capacity:  286 of 422
ROUTE[5]:31->11->85->36->18->9->64->93 Total vertices = 8 Travel Time = 07h:58m Capacity:  261 of 432
ROUTE[6]:54->8->24->89->47->22->4->43 Total vertices = 8 Travel Time = 07h:55m Capacity:  252 of 477
ROUTE[7]:96->23->92->82->97->63->94->13->66 Total vertices = 9 Travel Time = 07h:53m Capacity:  223 of 494
ROUTE[8]:65->10->75->98->20->34->60->57->32 Total vertices = 9 Travel Time = 07h:57m Capacity:  271 of 408
ROUTE[9]:45->3->79->5->25->61->78->29 Total vertices = 8 Travel Time = 07h:59m Capacity:  245 of 493
ROUTE[10]:26->35->15->76->58->100->19->59->6 Total vertices = 9 Travel Time = 07h:35m Capacity:  266 of 473
Vertices visited: 90 of 100 [90.00%]
Total Travel Time: 78h:55m

13450 90 78:55:00 00:01:00

Execution Time: 00h:01m:00s
```

Figura 16 - Ficheiro de soluções

4. RESULTADOS

Neste capítulo apresentam-se os resultados obtidos nas experiências computacionais realizadas a partir da aplicação do CTOPTW a diversas instâncias assimétricas e euclidianas, explorando estratégias alternativas usadas para tentar obter soluções melhores.

Inicialmente tentou-se realizar os testes na plataforma *Search* da Universidade do Minho pois esta permite submeter, ao mesmo tempo, várias instâncias ao teste, o que facilita muito a obtenção de soluções. A maior vantagem da *Search* é que permite o uso de computação paralela e distribuída, o que é compatível com a *framework* BRKGA. Infelizmente, devido a incompatibilidades entre o código desenvolvido e o compilador utilizado pela *Search*, não foi possível atingir este objetivo.

Portanto, acabou-se por realizar os testes num computador pessoal do modelo Acer Aspire E5-573 com um processador Intel ® Celeron ® 2957U @ 1.40 GHz e memória RAM de 4 GB.

4.1 Resultados para instâncias assimétricas

Foram geradas 10 instâncias assimétricas e para cada uma delas foi feito 5 *runs* para cada um dos três métodos de alocação dos vértices fazendo um total de 15 *runs* por instância. Cada uma das instâncias contêm informações relativas a 100 clientes e 10 veículos diferentes e cada *run* foi executado para uma população de tamanho 20 e 100 evoluções.

Na Tabela 1 encontra-se os resultados médios para cada instância e nas Tabelas Tabela 6, Tabela 7 e Tabela 8 presentes no Anexo 1 encontram-se apresentados os resultados obtidos em cada *run*.

Tabela 1- Resultados médios para instâncias assimétricas

Instância	Fitness					Vertices (%)				Viagens (h)				CPU			
	Max	Serial	Parallel	Low Cost	Best	Serial	Parallel	Low Cost	Best	Serial	Parallel	Low Cost	Best	Serial	Parallel	Low Cost	Best
RD_100C_6	14721	8561,80	8692,80	12045,20	12045,20	56,80	58,00	80,40	80,40	75:15:00	75:21:36	74:59:12	74:59:12	00:03:00	00:01:00	00:10:24	00:01:00
RD_100C_7	15364	9314,20	9279,00	12557,60	12557,60	59,00	58,40	80,20	80,20	74:47:00	74:14:12	75:49:24	74:14:12	00:03:00	00:01:00	00:10:48	00:01:00
RD_100C_8	14731	8217,40	8330,40	11678,60	11678,60	55,60	55,40	78,00	78,00	71:55:12	71:31:24	73:51:00	71:31:24	00:03:24	00:01:00	00:10:00	00:01:00
RD_100C_9	15343	8990,60	9045,60	12364,40	12364,40	57,20	57,40	79,60	79,60	73:00:24	73:15:36	74:31:00	73:00:24	00:03:36	00:01:00	00:10:00	00:01:00
RD_100C_10	15513	8372,20	8441,20	12212,40	12212,40	53,00	52,80	77,00	77,00	73:20:48	72:19:12	74:02:12	72:19:12	00:03:48	00:01:00	00:09:48	00:01:00
RD_100C_11	14893	8952,00	8888,60	12194,80	12194,80	59,60	58,40	80,20	80,20	74:52:36	74:37:24	74:56:24	74:37:24	00:04:48	00:01:12	00:12:12	00:01:12
RD_100C_12	14757	8839,20	8755,80	11906,40	11906,40	58,60	58,20	78,60	78,60	72:24:36	72:45:48	74:11:48	72:24:36	00:05:12	00:02:00	00:15:12	00:02:00
RD_100C_13	15091	8849,40	8549,20	12054,40	12054,40	57,00	55,00	78,20	78,20	72:58:24	72:56:36	73:44:36	72:56:36	00:03:00	00:01:00	00:13:24	00:01:00
RD_100C_14	15271	8499,40	8825,40	11907,20	11907,20	54,60	56,60	77,00	77,00	74:35:36	75:13:00	75:19:00	74:35:36	00:03:12	00:01:00	00:09:24	00:01:00
RD_100C_15	14495	7880,60	7830,60	11453,40	11453,40	52,80	51,60	76,80	76,80	72:45:24	73:13:48	74:51:36	72:45:24	00:03:24	00:01:00	00:10:00	00:01:00
Média	15018	8648	8664	12037		56,4	56,2	78,6		73:35:30	73:32:52	74:37:37		00:03:38	00:01:07	00:11:07	
% da média		57,6%	57,7%	80,2%													

Na Tabela 1 a coluna *Instância* refere-se às instâncias submetidas a testes computacionais, a coluna *Max* refere-se ao valor máximo de *fitness* que pode ser recolhido para cada instância e este é um valor grosseiro para avaliar o desempenho dos métodos. Uma vez que não é expectável que seja possível recolher a totalidade dos prémios disponíveis (valor nominal da instância) consideramos este valor como um indicador "grosseiro" dado que pode estar muito distante do valor ótimo da instância que não é conhecido. A coluna *Fitness* refere-se à recompensa média recolhida por cada método para cada uma das instâncias, a coluna *Vertices(%)* indica a percentagem média de vértices visitados por cada método, a coluna *Viagens(h)* refere-se ao tempo médio de viagem realizado pelas frotas, a coluna *CPU* indica o tempo médio de execução para um determinado método.

Como se pode verificar na Tabela 1 o método *Low Cost* (inserção de menor custo) obteve melhores resultados a nível do *fitness* com uma média de 80.2% da média das recompensas máximas (nominais) e a nível da percentagem dos vértices visitados com uma média de 78.6%. A nível do tempo médio das viagens, o método de inserção em paralelo obteve globalmente melhores resultados, mas fazendo uma relação entre a percentagem de vértices visitados e o tempo médio das viagens, globalmente o método de inserção de menor custo obteve melhores resultados e isto deve-se ao facto deste método tentar reduzir ao máximo o tempo de viagem realizado pela frota, como se pode verificar no pseudo-código anteriormente apresentada na Figura 10. A nível do tempo de *CPU* consumido, o método inserção em paralelo obteve os melhores tempos com uma média de 1 minuto e 7 segundos.

Elaborou-se, no *Excel*, os gráficos apresentados nas Figuras Figura 17 e Figura 18, de modo a analisar o comportamento dos algoritmos ao longo das gerações. Nesta experiência, foi usada a instância *RD_100C_6*, escolhida aleatoriamente.

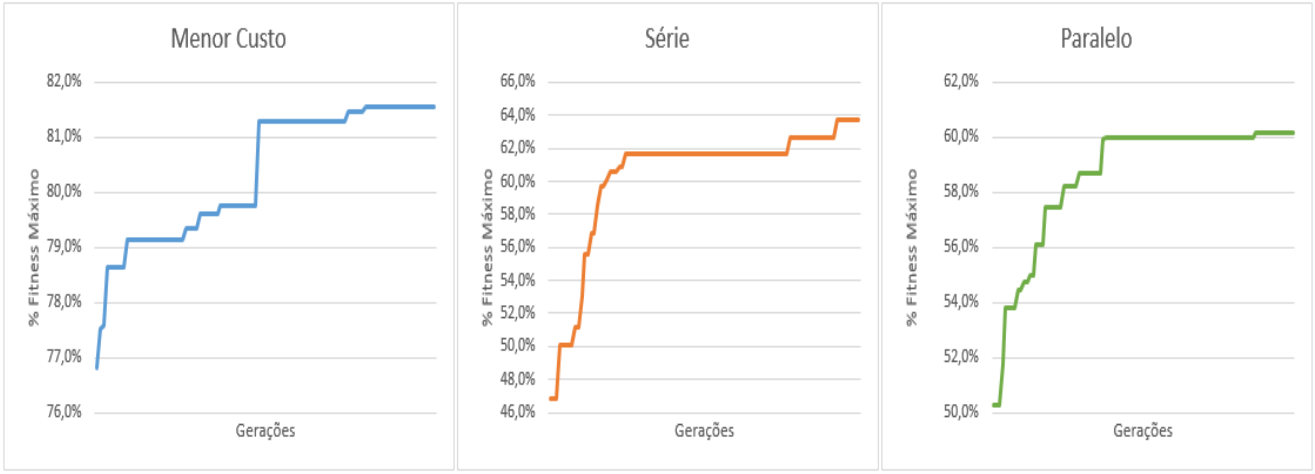

Figura 17 – Comparação da evolução de uma solução – Assimétrica

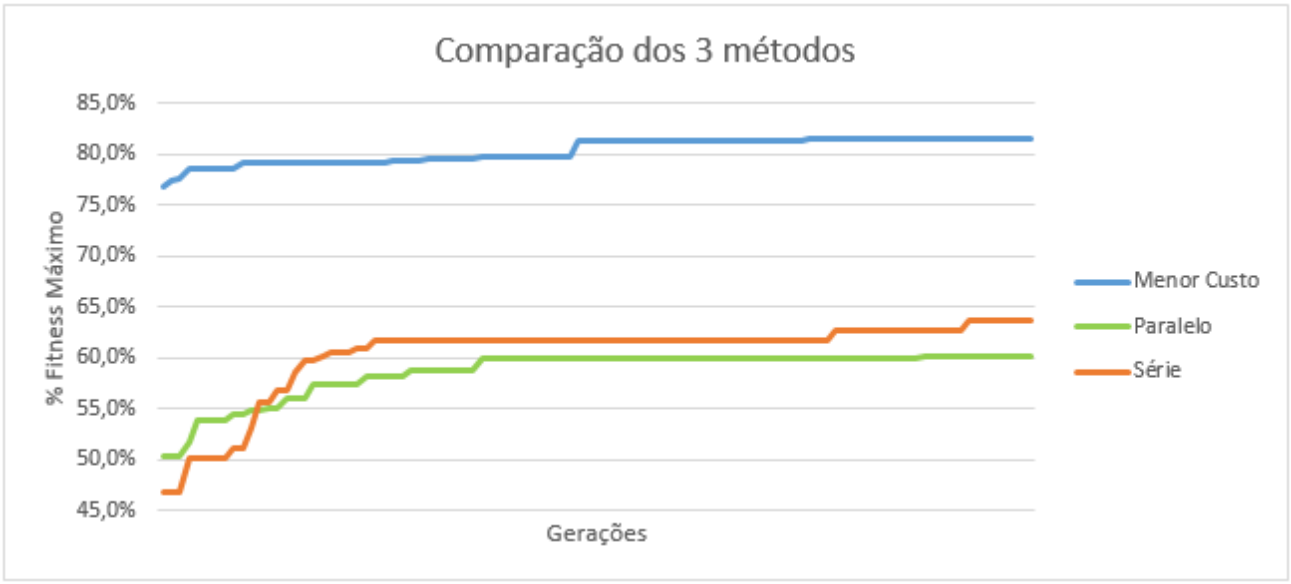

Figura 18 - Comparação dos 3 métodos - Assimétrica

Da observação do gráfico, pode-se verificar que os métodos normalmente encontram novas soluções mais rapidamente na fase inicial da execução e depois vão progredindo de forma mais lenta passando algumas gerações. O método de inserção de menor custo obteve logo à partida o valor de *fitness*, 76.8% do *fitness* máximo (nominal), melhor do que os outros métodos conseguem alcançar no final das suas evoluções, no caso da inserção em paralelo 60.2% do *fitness* máximo (nominal) e no caso da inserção em série 63.8% do *fitness* máximo (nominal).

4.2 Resultados para instâncias euclidianas

Foram realizados testes a 10 instâncias euclidianas e de forma análogo à das instâncias assimétricas foram feitos 15 *runs* por instância. As instâncias contêm informações relativas a 100 clientes e 10 veículos e cada *run* foi executado para uma população de tamanho 20 e 100 evoluções.

Na Tabela 2 encontra-se os resultados médios para cada instância e nas Tabelas Tabela 9, Tabela 10 e Tabela 11 presentes no Anexo 2 encontram-se os resultados obtidos em cada *run*.

Tabela 2- Resultados médios para instâncias euclidianas

Instância	Fitness					Vertices (%)				Viagens (h)				CPU			
	Max	Serial	Parallel	Low Cost	Best	Serial	Parallel	Low Cost	Best	Serial	Parallel	Low Cost	Best	Serial	Parallel	Low Cost	Best
RD_100E_100C_1	15209	10876,20	11176,00	11614,60	11614,60	70,40	72,00	75,80	75,80	70:40:48	70:49:24	71:22:24	70:40:48	00:03:36	00:01:00	00:10:48	00:01:00
RD_100E_100C_2	14957	12285,20	12411,80	12960,20	12960,20	81,00	81,60	86,00	86,00	78:39:48	78:44:00	78:54:24	78:39:48	00:03:00	00:01:12	00:15:00	00:01:12
RD_100E_100C_3	14851	12706,00	12990,60	13254,20	13254,20	84,40	86,40	88,60	88,60	78:24:24	78:55:36	79:12:24	78:24:24	00:03:00	00:01:24	00:14:36	00:01:24
RD_100E_100C_4	14548	11299,00	11516,00	11898,00	11898,00	76,40	77,60	81,00	81,00	78:47:24	78:47:12	79:16:24	78:47:12	00:03:36	00:01:12	00:11:36	00:01:12
RD_100E_100C_5	15169	12591,80	12824,60	12700,00	12824,60	82,40	83,40	83,00	83,40	78:55:12	78:59:24	79:31:48	78:55:12	00:03:00	00:01:00	00:13:12	00:01:00
RD_100E_100C_6	14900	12440,20	12373,80	12692,40	12692,40	83,00	82,20	84,80	84,80	78:36:36	78:55:36	79:30:12	78:36:36	00:03:24	00:01:00	00:12:00	00:01:00
RD_100E_100C_7	14946	11556,60	11699,20	12133,60	12133,60	76,20	76,80	80,00	80,00	78:49:00	78:56:48	79:14:36	78:49:00	00:05:00	00:01:00	00:11:00	00:01:00
RD_100E_100C_8	14592	11735,00	12264,20	12352,00	12352,00	79,40	82,60	83,80	83,80	78:38:36	78:55:00	79:18:12	78:38:36	00:05:24	00:01:00	00:12:24	00:01:00
RD_100E_100C_9	15405	12207,00	12476,80	13038,00	13038,00	77,80	80,40	84,00	84,00	78:43:36	78:15:00	79:11:24	78:15:00	00:03:12	00:01:00	00:13:36	00:01:00
RD_100E_100C_10	14773	11432,60	11704,40	12339,00	12339,00	76,20	77,60	82,20	82,20	78:20:12	78:55:48	78:57:00	78:20:12	00:03:00	00:01:00	00:12:00	00:01:00
Média	14935	11913	12144	12498		78,7	80,1	82,9		77:51:34	78:01:23	78:26:53		00:03:37	00:01:05	00:12:37	
% da média		79,8%	81,3%	83,7%													

Pode-se constatar na Tabela 2 que globalmente o método de inserção de menor custo obteve melhores resultados a nível do *fitness* com uma média de 83.7% da média das recompensas máximas nominais e a nível da percentagem dos vértices visitados com uma média de 82.9%. A nível do tempo médio das viagens, o método de inserção em série obteve globalmente melhores resultados, mas fazendo uma relação entre a percentagem de vértices visitados e o tempo médio da viagem realizada pela frota, globalmente o método de inserção de menor custo obteve melhores resultados. O método de inserção em paralelo obteve melhores resultados no que diz respeito a tempo gasto pelo CPU na execução dos algoritmos.

Os três métodos de inserção obtiveram resultados muito próximos, possivelmente próximos do valor ótimo, porque as instâncias euclidianas são mais fáceis do que as instâncias assimétricas.

As Figuras Figura 19 e Figura 20 apresentam gráficos para a comparação da evolução das soluções obtidas em experiências realizadas sobre a instância *RD_100E_100C_3*, escolhida aleatoriamente.

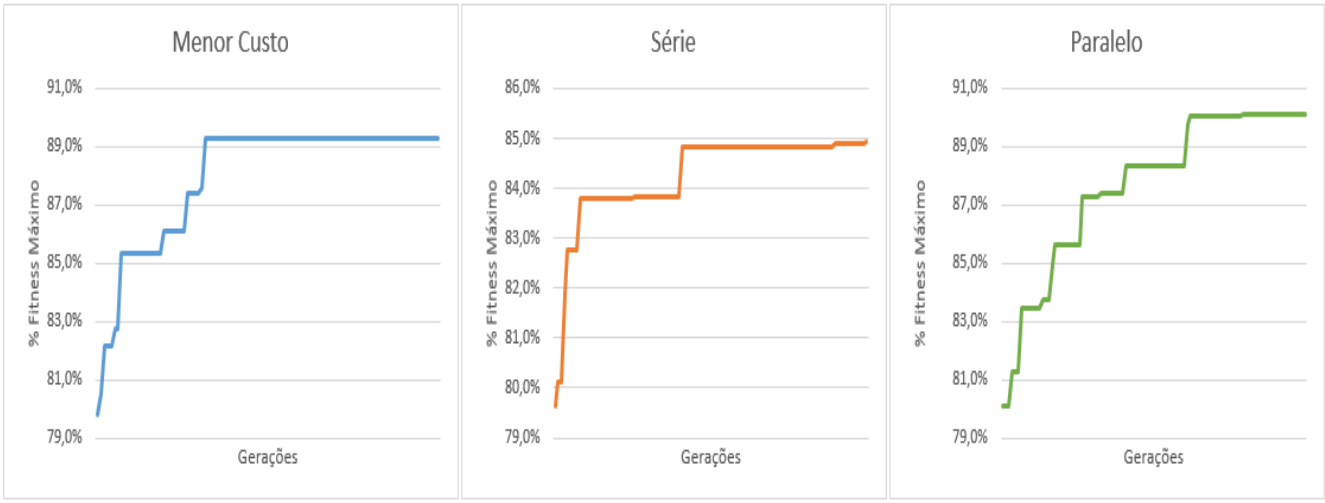

Figura 19 - Comparação da evolução de uma solução – Euclidiana

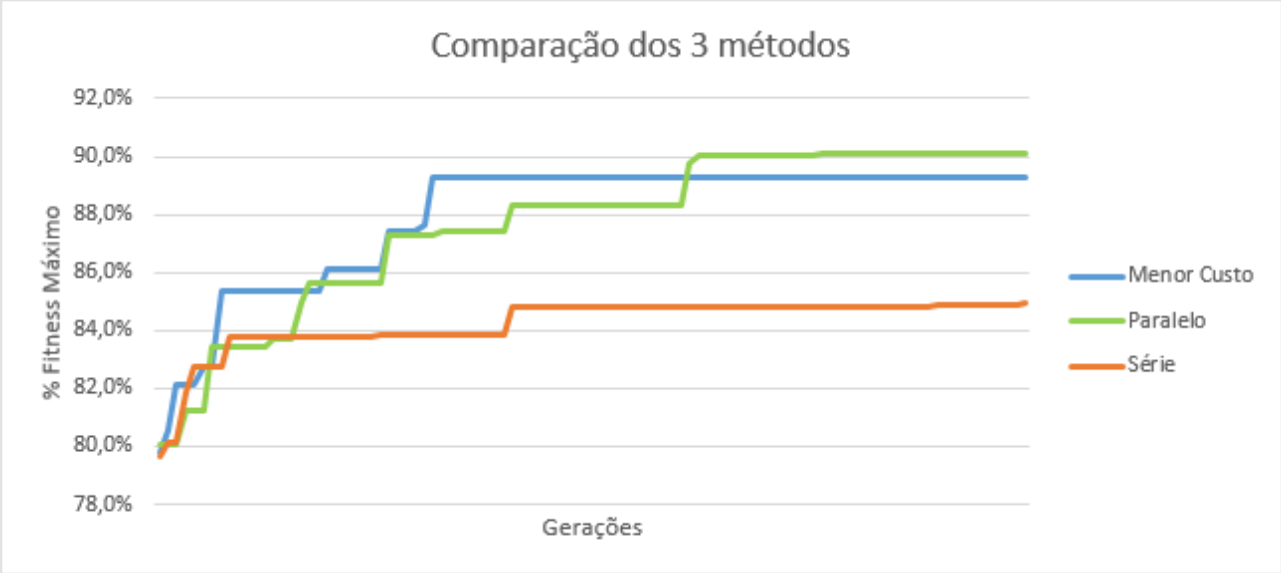

Figura 20 – Comparação dos 3 métodos – Euclidiana

Como se pode observar nos gráficos, acima representados, as soluções evoluíram de forma mais rápida nas primeiras gerações e depois melhoraram de forma mais lenta à medida que o número de gerações aumentaram. Diferente da evolução das soluções para as instâncias assimétricas (Figura 18), as soluções para as instâncias euclidianas atingem valores de fitness muito próximos nas primeiras evoluções e distanciando-se muito pouco ao longo das gerações.

4.3 Comparação dos resultados

Nos testes realizados verificou-se que as melhores soluções encontradas pertenciam a instâncias euclidianas. O que possivelmente pode explicar esses resultados pode ser o fato de que nas instâncias euclidianas foi aplicado o algoritmo de Floyd para corrigir as distâncias entre todos os clientes, ou seja, são usadas sempre as distâncias mais curtas nas visitas a um vértice, fazendo com que os veículos possam visitar mais clientes do que em relação a instâncias assimétricas, onde as distâncias podem ser diferentes para ir e voltar de um vértice, o que pode aumentar as distâncias percorridas pelos veículos numa visita a um cliente. Este fato torna as instâncias euclidianas mais fáceis de resolver pelos três métodos do que as instâncias assimétricas.

4.4 Análise de sensibilidade

Após os testes realizados com o intuito de conhecer qual dos três métodos de inserção obtém melhores resultados, procedeu-se a uma análise de sensibilidade do ponto de vista de gestão do processo de recolha. Sendo o método de inserção de menor custo o método que melhores resultados obteve nas experiências computacionais e os dados euclidianos os tipos mais fáceis de resolver, então foi utilizado o método de inserção de menor custo e dados euclidianos para realizar os testes a todos os cenários da análise de sensibilidade.

Para a análise da sensibilidade optou-se por testar 3 cenários distintos, a seguir apresentados.

4.4.1 Variação do número de viaturas

Neste cenário pretende-se estudar como a variação do número de viaturas disponíveis alteram as soluções. Para isso realizou-se testes com uma instância em que foi alterado o número de veículos em cada teste para 1, 5, 10, 12 e 15 veículos, todos com a capacidade de 500 e tempo de serviço de 8 horas, mas com início de serviços aleatórios entre as 06:00h e 09:00h. Nas Figuras Figura 21, Figura 22, Figura 23, Figura 24 e Figura 25 encontram-se apresentado os resultados obtidos em cada um dos testes e a Tabela 3 apresenta os valores mínimos, máximos e médios obtidos.

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 2156 of 14851
ROUTE[1]:27->14->92->46->28->95->83->47->60->38->85->15->66 Total vertices = 13 Travel Time = 07h:58m Capacity:  411 of 500
Vertices visited: 13 of 100 [13.00%]
Total Travel Time: 07h:58m
```

Figura 21 - Resultado obtido para 1 veículo

Na Figura 21 o veículo conseguiu visitar 13% dos clientes em 7 horas e 58 minutos e encheu 82.2% da sua capacidade de carga.

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 7598 of 14851
ROUTE[1]:72->58->20->47->94->9->36->5->12->33 Total vertices = 10 Travel Time = 08h:00m Capacity:  283 of 500
ROUTE[2]:42->60->96->83->65->18->30->29->16->101->70 Total vertices = 11 Travel Time = 07h:59m Capacity:  331 of 500
ROUTE[3]:2->35->31->53->63->39->86->80->90 Total vertices = 9 Travel Time = 08h:00m Capacity:  274 of 500
ROUTE[4]:27->46->50->74->4->68->32->62->11->71 Total vertices = 10 Travel Time = 08h:00m Capacity:  303 of 500
ROUTE[5]:7->15->21->67->6->59->99->95->52->28 Total vertices = 10 Travel Time = 07h:59m Capacity:  286 of 500
Vertices visited: 50 of 100 [50.00%]
Total Travel Time: 39h:58m
```

Figura 22 - Resultado obtido para 5 veículos

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 13347 of 14851
ROUTE[1]:54->53->91->81->40->8->27->5 Total vertices = 8 Travel Time = 07h:53m Capacity:  256 of 500
ROUTE[2]:37->94->65->83->32->10->74->60->4->20->34->67 Total vertices = 12 Travel Time = 08h:00m Capacity:  337 of 500
ROUTE[3]:92->86->99->95->26->47->11->46->24->48 Total vertices = 10 Travel Time = 07h:54m Capacity:  311 of 500
ROUTE[4]:56->19->62->70->71->18->6->73 Total vertices = 8 Travel Time = 07h:59m Capacity:  244 of 500
ROUTE[5]:97->29->14->63->42->84->30->51->93 Total vertices = 9 Travel Time = 08h:00m Capacity:  276 of 500
ROUTE[6]:2->41->15->72->80->52->57->31->25 Total vertices = 9 Travel Time = 07h:59m Capacity:  256 of 500
ROUTE[7]:55->90->28->13->33->96->76->78 Total vertices = 8 Travel Time = 07h:59m Capacity:  265 of 500
ROUTE[8]:22->35->12->88->69->64->23->39->100 Total vertices = 9 Travel Time = 07h:49m Capacity:  297 of 500
ROUTE[9]:58->61->82->36->17->87->77 Total vertices = 7 Travel Time = 07h:51m Capacity:  197 of 500
ROUTE[10]:3->66->44->7->59->38->98->43->101 Total vertices = 9 Travel Time = 07h:49m Capacity:  256 of 500
Vertices visited: 89 of 100 [89.00%]
Total Travel Time: 79h:13m
```

Figura 23 - Resultado obtido para 10 veículos

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 14489 of 14851
ROUTE[1]:58->93->43->47->32->50->57->95->63 Total vertices = 9 Travel Time = 07h:57m Capacity:  251 of 500
ROUTE[2]:99->62->28->46->2->4->87->44->25->39->64 Total vertices = 11 Travel Time = 07h:49m Capacity:  318 of 500
ROUTE[3]:94->24->40->5->7->26->77->69 Total vertices = 8 Travel Time = 07h:58m Capacity:  262 of 500
ROUTE[4]:80->91->45->10->37->75->29 Total vertices = 7 Travel Time = 08h:00m Capacity:  208 of 500
ROUTE[5]:72->41->96->53->78->86->90->23 Total vertices = 8 Travel Time = 07h:58m Capacity:  218 of 500
ROUTE[6]:66->30->100->12->83->65->79->16->27->36->88->51 Total vertices = 12 Travel Time = 07h:55m Capacity:  378 of 500
ROUTE[7]:82->19->59->11->85->97->8 Total vertices = 7 Travel Time = 07h:55m Capacity:  211 of 500
ROUTE[8]:13->18->84->49->98->20->22->6->101 Total vertices = 9 Travel Time = 07h:55m Capacity:  288 of 500
ROUTE[9]:55->38->52->21->92->14->81 Total vertices = 7 Travel Time = 07h:56m Capacity:  215 of 500
ROUTE[10]:17->33->31->61->35->3->9 Total vertices = 7 Travel Time = 07h:57m Capacity:  214 of 500
ROUTE[11]:60->15->54->74->68->42->70->73 Total vertices = 8 Travel Time = 07h:56m Capacity:  243 of 500
ROUTE[12]:71->48->76->89 Total vertices = 4 Travel Time = 05h:15m Capacity:  127 of 500
Vertices visited: 97 of 100 [97.00%]
Total Travel Time: 92h:31m
```

Figura 24 - Resultado obtido para 12 veículos

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 14851 of 14851
ROUTE[1]:82->8->53->61->96->93->101 Total vertices = 7 Travel Time = 07h:58m Capacity:  202 of 500
ROUTE[2]:29->69->26->57->92->66->33->73 Total vertices = 8 Travel Time = 07h:56m Capacity:  273 of 500
ROUTE[3]:91->32->58->74->63->89->84 Total vertices = 7 Travel Time = 07h:58m Capacity:  208 of 500
ROUTE[4]:68->80->5->3->22->48->55 Total vertices = 7 Travel Time = 07h:59m Capacity:  239 of 500
ROUTE[5]:90->75->97->28->9->85->41 Total vertices = 7 Travel Time = 07h:57m Capacity:  222 of 500
ROUTE[6]:30->31->71->16->4->87->7->100->62 Total vertices = 9 Travel Time = 07h:51m Capacity:  265 of 500
ROUTE[7]:46->18->44->76->10->20->49 Total vertices = 7 Travel Time = 08h:00m Capacity:  220 of 500
ROUTE[8]:19->77->94->14->37->21->72->47->17->23->39 Total vertices = 11 Travel Time = 07h:48m Capacity:  276 of 500
ROUTE[9]:38->40->51->42->54 Total vertices = 5 Travel Time = 08h:00m Capacity:  158 of 500
ROUTE[10]:52->95->45->11->60->34->6->70 Total vertices = 8 Travel Time = 07h:59m Capacity:  239 of 500
ROUTE[11]:12->2->56->86->25->50->35 Total vertices = 7 Travel Time = 07h:48m Capacity:  222 of 500
ROUTE[12]:13->24->79->78->59->15->65 Total vertices = 7 Travel Time = 07h:37m Capacity:  199 of 500
ROUTE[13]:98->27->99->88->81 Total vertices = 5 Travel Time = 05h:41m Capacity:  151 of 500
ROUTE[14]:67->64->36 Total vertices = 3 Travel Time = 03h:49m Capacity:  83 of 500
ROUTE[15]:83->43 Total vertices = 2 Travel Time = 02h:31m Capacity:  58 of 500
Vertices visited: 100 of 100 [100.00%]
Total Travel Time: 106h:52m
```

Figura 25 - Resultado obtido para 15 veículos

Tabela 3 - Resultados para a variação do nº viaturas

		Número de Veículos				
		1	5	10	12	15
% Vértices visitados		13	50	89	97	100
Vértices visitados	Mínimo	13	9	7	4	2
	Média		10	8,9	8,1	6,7
	Máximo		11	12	12	11
Tempo de viagens	Mínimo	07:58	07:59	07:49	05:15	02:31
	Média		07:59	07:55	07:42	07:07
	Máximo		08:00	08:00	08:00	08:00
Capacidade ocupada	Mínima	411	274	197	127	58
	Média		295,4	269,5	244,4	201,0
	Máxima		331	337	378	276

Da análise aos resultados obtidos conclui-se, como seria de esperar, que à medida que aumenta o número de veículos disponíveis para a recolha, aumenta também a percentagem de clientes visitados, porém as capacidades dos veículos não são muito bem aproveitadas, por exemplo, no caso de 15 veículos a capacidade ocupada, globalmente, pelos veículos é menor do que 50% da capacidade disponível.

Perante uma situação real sob as condições anteriormente apresentadas e os resultados obtidos nas experiências, o decisor teria de analisar as opções de 10 e 12 veículos e decidir se é mais favorável visitar 89% dos clientes e aproveitar melhor a disponibilidade e capacidade

dos veículos ou se é mais favorável visitar 97% dos clientes e ter o veículo da rota 12 a terminar o serviço aproximadamente 3 horas antes.

4.4.2 Variação da capacidade dos veículos

Neste cenário pretende-se estudar como a variação da capacidade dos veículos influencia as soluções. Optou-se por efetuar três testes à instância usada na Figura 23 e uma vez que os veículos não aproveitam da melhor forma a capacidade de 500, então optou-se por utilizar menores capacidades nestes testes, em que se utilizou capacidades de 400, 300 e 200 em cada um dos testes respetivamente. Nas Figuras Figura 26, Figura 27 e Figura 28 encontram-se apresentados os resultados obtidos e a Tabela 4 apresenta os valores mínimos, máximos e médios.

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 13208 of 14851
ROUTE[1]:81->83->76->54->35->50->23 Total vertices = 7 Travel Time = 07h:53m Capacity:  202 of 400
ROUTE[2]:17->9->38->20->101->90->3->100 Total vertices = 8 Travel Time = 07h:50m Capacity:  228 of 400
ROUTE[3]:5->25->42->94->34->92->14->29->66 Total vertices = 9 Travel Time = 07h:50m Capacity:  290 of 400
ROUTE[4]:2->13->57->96->15->62->28->87->72 Total vertices = 9 Travel Time = 07h:52m Capacity:  232 of 400
ROUTE[5]:93->65->75->10->47->67->52->98->22->16 Total vertices = 10 Travel Time = 08h:00m Capacity:  305 of 400
ROUTE[6]:46->63->26->48->31->77->58->7->71->70 Total vertices = 10 Travel Time = 07h:55m Capacity:  313 of 400
ROUTE[7]:80->56->41->82->24->37->43->60 Total vertices = 8 Travel Time = 07h:50m Capacity:  226 of 400
ROUTE[8]:84->91->19->64->97->11->32 Total vertices = 7 Travel Time = 07h:53m Capacity:  223 of 400
ROUTE[9]:68->6->49->73->39->88->61->36->18->27->51 Total vertices = 11 Travel Time = 07h:52m Capacity:  362 of 400
ROUTE[10]:86->99->4->74->69->78->30->95->33 Total vertices = 9 Travel Time = 07h:53m Capacity:  277 of 400
Vertices visited: 88 of 100 [88.00%]
Total Travel Time: 78h:48m
```

Figura 26 - Resultado obtido para capacidade 400

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 13420 of 14851
ROUTE[1]:24->18->92->41->98->56->32->57->47 Total vertices = 9 Travel Time = 08h:00m Capacity:  279 of 300
ROUTE[2]:51->13->27->21->79->15->35->94->3->39 Total vertices = 10 Travel Time = 08h:00m Capacity:  263 of 300
ROUTE[3]:80->19->14->78->12->60->58->52->10->72 Total vertices = 10 Travel Time = 07h:55m Capacity:  295 of 300
ROUTE[4]:87->63->5->45->66->77->7->83->48->97 Total vertices = 10 Travel Time = 08h:00m Capacity:  298 of 300
ROUTE[5]:17->26->31->100->59->28->95->22 Total vertices = 8 Travel Time = 08h:00m Capacity:  269 of 300
ROUTE[6]:25->89->91->29->62->96->8->82->30->64 Total vertices = 10 Travel Time = 07h:59m Capacity:  298 of 300
ROUTE[7]:46->61->54->53->43->99->101->2->11 Total vertices = 9 Travel Time = 07h:59m Capacity:  266 of 300
ROUTE[8]:6->23->84->70->9->40->88->71->73 Total vertices = 9 Travel Time = 07h:59m Capacity:  265 of 300
ROUTE[9]:90->68->34->20->36->44->93->86 Total vertices = 8 Travel Time = 07h:45m Capacity:  246 of 300
ROUTE[10]:81->69->33->74->4->67->65 Total vertices = 7 Travel Time = 07h:57m Capacity:  219 of 300
Vertices visited: 90 of 100 [90.00%]
Total Travel Time: 79h:34m
```

Figura 27 - Resultado obtido para capacidade 300

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 10383 of 14851
ROUTE[1]:53->35->72->29->7->89 Total vertices = 6 Travel Time = 06h:12m Capacity:  189 of 200
ROUTE[2]:57->21->3->74->51->95->62 Total vertices = 7 Travel Time = 06h:53m Capacity:  194 of 200
ROUTE[3]:46->31->82->40->63->30->28 Total vertices = 7 Travel Time = 05h:31m Capacity:  198 of 200
ROUTE[4]:2->38->94->5->64->45->99 Total vertices = 7 Travel Time = 07h:58m Capacity:  191 of 200
ROUTE[5]:27->19->39->20->58->92->60 Total vertices = 7 Travel Time = 07h:55m Capacity:  197 of 200
ROUTE[6]:67->10->76->81->14->25->23 Total vertices = 7 Travel Time = 07h:41m Capacity:  198 of 200
ROUTE[7]:13->68->24->79->18->15->16 Total vertices = 7 Travel Time = 07h:19m Capacity:  199 of 200
ROUTE[8]:100->6->52->78->65->71 Total vertices = 6 Travel Time = 06h:30m Capacity:  195 of 200
ROUTE[9]:80->96->47->8->42->66->101 Total vertices = 7 Travel Time = 07h:18m Capacity:  197 of 200
ROUTE[10]:37->59->56->36->97->50->33 Total vertices = 7 Travel Time = 07h:12m Capacity:  200 of 200
Vertices visited: 68 of 100 [68.00%]
Total Travel Time: 70h:29m
```

Figura 28 - Resultado obtido para capacidade 200

Tabela 4 - Resultados para a variação da capacidade

		Capacidade dos Veículos			
		500	400	300	200
% Vértices visitados		89	88	90	68
Vértices visitados	Mínimo	7	7	7	6
	Média	8,9	8,8	9,0	6,8
	Máximo	12	11	10	7
Tempo de viagens	Mínimo	07:49	07:50	07:45	05:31
	Média	07:55	07:52	07:57	07:02
	Máximo	08:00	08:00	08:00	07:58
Capacidade ocupada	Mínima	197	202	219	189
	Média	269,5	265,8	269,8	195,8
	Máxima	337	362	298	200

Das experiências computacionais realizadas, pode-se verificar que a diminuição da capacidade dos veículos permite, globalmente, o aumento do seu aproveitamento em termos de capacidade de carga. No entanto, deve-se ter cuidado com a diminuição da capacidade dos veículos para um valor muito baixo porque isso leva a que as restrições de capacidade não permitam visitar muitos clientes, por isso é importante contrabalançar estes dois fatores.

4.4.3 Variação das horas extra para os motoristas

Pretende-se estudar influência da variação das horas extra, para os motoristas, sobre as soluções. Realizaram-se testes sobre a instância utilizada na Figura 23, com 10 veículos e

capacidade de 500, em que num teste tem 1 hora extra (9 horas de serviço) associado à disponibilidade dos veículos e a noutra tem 2 horas extra (10 horas de serviço). Os resultados encontram-se apresentados nas Figuras Figura 29 e Figura 30 e a Tabela 5 apresenta os valores mínimos, máximos e médios obtidos.

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 13962 of 14851
ROUTE[1]:46->76->83->30->61->91->86->51->95->5 Total vertices = 10 Travel Time = 08h:56m Capacity:  317 of 500
ROUTE[2]:6->19->29->82->77->11->36->50->52 Total vertices = 9 Travel Time = 08h:38m Capacity:  269 of 500
ROUTE[3]:54->81->79->18->42->70->59->49->89->39 Total vertices = 10 Travel Time = 09h:00m Capacity:  308 of 500
ROUTE[4]:87->2->10->17->23->60->62->55->88 Total vertices = 9 Travel Time = 08h:58m Capacity:  238 of 500
ROUTE[5]:13->35->37->94->96->33->92->16->48->64 Total vertices = 10 Travel Time = 08h:53m Capacity:  300 of 500
ROUTE[6]:74->58->63->97->9->31->85 Total vertices = 7 Travel Time = 08h:58m Capacity:  192 of 500
ROUTE[7]:45->21->98->101->93->44->69->8 Total vertices = 8 Travel Time = 08h:55m Capacity:  234 of 500
ROUTE[8]:80->20->47->3->22->4->32->38->57->15->65 Total vertices = 11 Travel Time = 08h:55m Capacity:  336 of 500
ROUTE[9]:27->14->84->100->7->66->26->25->99->73 Total vertices = 10 Travel Time = 09h:00m Capacity:  341 of 500
ROUTE[10]:68->56->90->28->43->12->75->78->71 Total vertices = 9 Travel Time = 08h:03m Capacity:  300 of 500
Vertices visited: 93 of 100 [93.00%]
Total Travel Time: 88h:16m
```

Figura 29 - Resultados obtidos para 1 hora extra

```
Allocation Mode = Low Cost
Population Size = 20
Evolutions = 100
Best Fitness = 14443 of 14851
ROUTE[1]:92->24->72->4->34->9->57->7->74->51->29->78 Total vertices = 12 Travel Time = 09h:57m Capacity:  363 of 500
ROUTE[2]:6->39->61->36->12->53->31->54->33->70 Total vertices = 10 Travel Time = 08h:33m Capacity:  328 of 500
ROUTE[3]:44->55->8->69->30->50->67->19->52->101 Total vertices = 10 Travel Time = 09h:57m Capacity:  282 of 500
ROUTE[4]:68->76->48->47->32->2->22->43->94->38->85 Total vertices = 11 Travel Time = 09h:56m Capacity:  350 of 500
ROUTE[5]:3->10->63->18->20->65->15->71->17->73->49 Total vertices = 11 Travel Time = 09h:17m Capacity:  310 of 500
ROUTE[6]:93->87->96->25->13->90->11->35->98 Total vertices = 9 Travel Time = 09h:36m Capacity:  281 of 500
ROUTE[7]:14->82->81->62->16->99->23->95->79->5 Total vertices = 10 Travel Time = 09h:31m Capacity:  280 of 500
ROUTE[8]:84->42->64->88->75->83->100->37->59->77 Total vertices = 10 Travel Time = 08h:24m Capacity:  310 of 500
ROUTE[9]:91->40->66->26->58->21->56->41->60 Total vertices = 9 Travel Time = 09h:17m Capacity:  258 of 500
ROUTE[10]:80->45->27->86->28 Total vertices = 5 Travel Time = 06h:05m Capacity:  166 of 500
Vertices visited: 97 of 100 [97.00%]
Total Travel Time: 90h:33m
```

Figura 30 - Resultados obtidos para 2 horas extra

Tabela 5 - Resultados para a atribuição de horas extra

		Horas Extra		
		0	1	2
% Vértices visitados		89	93	97
Vértices visitados	Mínimo	7	7	5
	Média	8,9	9,3	9,7
	Máximo	12	11	12
Tempo de viagens	Mínimo	07:49	08:03	06:05
	Média	07:55	08:49	09:03
	Máximo	08:00	09:00	09:57
Capacidade ocupada	Mínima	197	192	166
	Média	269,5	283,5	292,8
	Máxima	337	341	363

O aumento do número de horas extra para os motoristas melhorou globalmente os resultados na medida em que se consegue visitar maior número de clientes e também a capacidade dos veículos é, globalmente, melhor aproveitada do que em relação aos resultados obtidos para a mesma instância sem a existência de horas extra (Figura 23).

Por outro lado, os resultados permitem observar que a opção de horas extra pode ser vantajosa só para algumas rotas, sendo necessário negociar um horário mais flexível.

4.4.4 Análise global dos cenários

Num contexto real, em que há possibilidade de calcular os custos em termos monetários, os responsáveis pela tomada de decisão têm o papel de criar e analisar diversos cenários para servir de apoio à tomada da decisão.

Essas decisões podem passar por aumentar ou diminuir o tamanho da frota como ilustrado na secção 4.4.1 ou por comprar veículos com maior ou menor capacidade (secção 4.4.2) ou por atribuir horas extra aos motoristas (secção 4.4.3) ou uma combinação destes, para reduzir os custos e otimizar o transporte de leite cru.

Separadamente, cada um dos cenários permite obter ganhos pontuais, porém para alcançar ganhos globais reduzindo custos dos transportes de leite cru é necessário conciliar os diferentes cenários de acordo com a realidade da empresa, conjugando o tamanho da frota com a capacidade dos veículos utilizados na recolha bem como a atribuição de horas extra a alguns motoristas para atender clientes não visitados.

Em casos em que nem todos os clientes são visitados, o decisor também deve analisar se é melhor atribuir um veículo de propósito para realizar essas recolhas isoladas, uma vez que o leite cru possui muitas restrições associadas que tornam imperativo a sua recolha numa janela temporal muito pequena, ou se é preferível perder algum cliente que, por exemplo a quantidade de leite cru a ser recolhido não compensa o gasto de deslocar uma viatura para fazer a recolha todos os dias.

5. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho realizou-se um estudo do comportamento dos modelos baseados nos problemas de orientação de equipas com restrições de capacidade e janelas temporais e na estrutura algorítmica de algoritmos genéticos de chave aleatória, aplicados ao problema da recolha e transporte de leite cru.

A revisão bibliográfica efetuada no âmbito deste trabalho incidiu sobre o estado da arte dos algoritmos genéticos, da *framework* BRKGA e dos problemas de orientação, nomeadamente as variantes do problema de orientação de equipas. As variantes do TOP estudadas foram os problemas de orientação de equipas com janelas temporais, problemas de orientação de equipas com restrições de capacidade e problemas de orientação de equipas com restrições de capacidade e janelas temporais.

Após o estudo do problema da recolha e transporte de leite cru e das restrições associadas ao mesmo, decidiu-se desenvolver um modelo matemático baseado nos problemas de orientação de equipas com restrições de capacidade e janelas temporais. O método de solução desenvolvido foi implementado em linguagem C++, por ser esta uma linguagem poderosa e robusta e também pelo fato de que já havia disponível a *framework* BRKGA, para esta linguagem, que lida automaticamente com a grande parte dos módulos independentes do problema, incluindo a gestão da população e a dinâmica evolutiva, deixando ao utilizador a tarefa de implementar um procedimento dependente do problema para converter um vetor de chaves aleatórias em uma solução para o problema de otimização subjacente.

Foi desenvolvida uma aplicação *standalone* para gerar e visualizar instâncias bem como para realizar testes às instâncias de forma a analisar o desempenho dos algoritmos desenvolvidos. A aplicação foi desenvolvida em linguagem C++ usando o IDE NetBeans 8.1 e o Qt Creator 3.6.1 para o desenvolvimento da interface gráfica.

Com o intuito de abranger as diversas situações reais, optou-se por se desenvolver instâncias euclidianas e assimétricas para efeito de testes. As instâncias euclidianas são as instâncias onde as distâncias entre os vértices são simétricas e respeitam a desigualdade triangular enquanto as instâncias assimétricas são instâncias, como o próprio nome indica, em que as distâncias entre os vértices são assimétricas e podem não respeitar a desigualdade triangular.

Foram desenvolvidos três métodos de alocação dos vértices, o método de inserção em série, inserção em paralelo e inserção de menor custo. Para a comparação dos métodos de inserção foram feitos testes a 20 instâncias, 10 assimétricas e 10 euclidianas, com diferentes números de evoluções. O método de inserção de menor custo obteve, globalmente, melhores resultados em relação ao *fitness* da solução, à percentagem dos vértices visitados e ao tempo total de viagens realizado pelas frotas.

A análise dos resultados permitiu verificar que as melhores soluções encontradas pertenciam a instâncias euclidianas. O que explica esses resultados é o fato de que nas instâncias euclidianas foi aplicado o algoritmo de Floyd para corrigir as distâncias entre todos os clientes, fazendo com que os veículos possam visitar mais clientes do que em relação a instâncias assimétricas, onde as distâncias podem ser diferentes para ir e voltar de um vértice, aumentando as distâncias percorridas pelos veículos numa visita a um cliente.

Após se concluir qual o método que melhores soluções obtém, fez-se uma análise de sensibilidade do ponto de vista da gestão do processo de recolha para três cenários distintos, utilizando o método de inserção de menor custo para instâncias do tipo euclidianas. O cenário 1 examina a influência da variação do número de veículos na solução, onde à medida que o número de veículos aumenta, maior é a percentagem de clientes visitados. Porém, os últimos veículos a alocar vértices visitam poucos clientes o que muitas vezes não compensa realizar essas recolhas tendo em conta os custos envolvidos. O cenário 2 observa como a variação da capacidade dos veículos influencia a solução. Neste estudo verificou-se que a diminuição da capacidade dos veículos permite, globalmente, o aumento do seu aproveitamento em termos de capacidade de carga. No entanto a diminuição da capacidade dos veículos para um valor muito baixo leva a que as restrições de capacidade não permitam visitar muitos clientes. O cenário 3 analisa influência da variação das horas extra para os motoristas. Neste cenário observou-se que a atribuição de horas extra aos motoristas melhorou globalmente os resultados na medida em que se consegue visitar maior número de clientes e também a capacidade dos veículos é melhor aproveitada. No entanto conclui-se que o recurso a horas extra só se justifica em algumas rotas, sendo necessário negociar um horário mais flexível.

Cada um dos diferentes cenários obtive ganhos em pontos específicos. Porém para alcançar ganhos globais reduzindo os custos dos transportes de leite cru é necessário conciliar os diferentes cenários de acordo com a realidade da empresa que se pretende estudar. Conjugando o tamanho da frota com a capacidade dos veículos usados no transporte, bem como, decidir se é necessário atribuir horas extra a alguns motoristas para atender clientes não

visitados são alternativas possíveis para melhorar o desempenho do processo de recolha de leite cru, usando esta ferramenta de apoio à decisão. Outra abordagem que esta ferramenta permite efetuar é decidir o envio de apenas um veículo propositadamente para realizar recolhas de vértices isolados, uma vez que o leite cru possui muitas restrições associadas à sua conservação, o que torna imperativo a sua recolha numa janela temporal muito pequena e também porque o custo de manter o leite cru a uma temperatura abaixo de 10ºC é elevado para ser suportado por pequenas e ou médias empresas do setor leiteiro. A ferramenta permite analisar o ajustamento da janela temporal da recolha de um cliente de modo a permitir a sua visita em conjunto com outros clientes.

5.1 Trabalhos futuros

Como perspetivas de trabalhos futuros, a aplicação *standalone* pode integrar uma funcionalidade onde é possível gerar instâncias com localizações reais de clientes, através do uso de *API's* como por exemplo Google Maps.

Dado que o uso de horas extraordinárias só interessa ser usado em algumas das rotas, será conveniente estudar o problema com rotas com limites temporais variáveis.

Também seria interessante testar os algoritmos para instâncias reais com um número elevado de clientes e evoluções e instâncias com múltiplas origens e destinos.

REFERÊNCIAS BIBLIOGRÁFICAS

Abreu, M. J. M. de. (2014). *Problema de orientação de equipas (TOP) aplicado às linhas de engarrafamento móveis*. University of Minho. Retrieved from https://repositorium.sdum.uminho.pt/handle/1822/33425

APROLEP. (2015). Produtores de leite nº 12, Outono/Inverno 2015, 1–48. Retrieved from https://aprolep.files.wordpress.com/2010/10/produtores-de-leite_12.pdf

Archetti, C., Feillet, D., Hertz, A., & Speranza, M. G. (2007). The Capacitated Team Orienteering and Profitable Tour Problems - 2007. Retrieved February 25, 2016, from https://www.gerad.ca/~alainh/G-2007-31.pdf

Archetti, C., Feillet, D., Hertz, A., & Speranza, M. G. (2009). The capacitated team orienteering and profitable tour problem. Retrieved February 14, 2016, from http://www.uv.es/route2011/Speranza.pdf

Archetti, C., Hertz, A., & Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. Retrieved February 14, 2016, from https://www.researchgate.net/publication/220403497_Metaheuristics_for_the_team_orienteering_problem_J_Heur

Boussier, S., Feillet, D., & Gendreau, M. (2006). An exact algorithm for team orienteering problems. *4OR*, *5*(3), 211–230. http://doi.org/10.1007/s10288-006-0009-1

Brito, D., Dias, G., Ferreira, I., Sousa, P., & Esteves, V. (2015). *Processamento de leite UHT*. Retrieved from http://www.esac.pt/noronha/pga/0708/trabalhos/Leite UHT - parte teórica PGA0708.pdf

Butt, S. E., & Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, *21*(1), 101–111. http://doi.org/10.1016/0305-0548(94)90065-5

Butt, S. E., & Ryan, D. M. (1999). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, *26*(4), 427–441. http://doi.org/10.1016/S0305-0548(98)00071-9

Carvalho, A. C. P. de L. F. de, Braga, A. de P., & Ludermir, T. B. (2003). Computação Evolutiva. In *Sistemas Inteligentes: Fundamentos e Aplicações*. Editora Manole Ltda.

Chao, I.-M., Golden, B. L., & Wasil, E. A. (1996). The team orienteering problem. *European Journal of Operational Research*, *88*(3), 464–474. http://doi.org/10.1016/0377-2217(94)00289-4

Gabinete de Planeamento e Politica - GPP. (2015). *Leite*. *Comissão Consultiva Setorial*. Retrieved from http://www.draplvt.mamaot.pt/Documents/Destaques/Apresenta%C3%A7%C3%A3o GPP - LEITE 2015 - CCS - 5_jun_FINAL.pdf

Golden, B. L., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, *34*(3), 307–318. http://doi.org/10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D

Gonçalves, J. F., & Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, *17*(5), 487–525. http://doi.org/10.1007/s10732-010-9143-1

Gunawan, A., Lau, H. C., & Lu, K. (2015). Well-Tuned ILS for Extended Team Orienteering Problem with Time Windows. Retrieved February 22, 2016, from https://centres.smu.edu.sg/larc/files/2015/09/Well-Tuned-ILS-for-Extended-Team-Orienteering-Problem-with-Time-WindowsTR-01-15.pdf

Holland, J. H. (1994). *Adaptation in Natural and Artificial Systems*. ((A Bradford Book), Ed.). Cambridge, Massachusetts.

Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, *54*(3), 648–665. http://doi.org/10.1016/j.cie.2007.10.001

Lemos, M. E. (2013). A Ovinicultura de leite /queijo Relação produção e Industria. Retrieved April 20, 2016, from http://www.drapc.min-agricultura.pt/base/geral/files/Eugenia_Lemos_higiene_refrigeracao_lite_ovelha.pdf

Li, Z., & Hu, X. (2011). The Team Orienteering Problem with Capacity Constraint and Time Window. Retrieved February 29, 2016, from http://www.aporc.org/LNOR/14/ISORA2011F18.pdf

Luo, Z., Cheang, B., Lim, A., & Zhu, W. (2013). An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem. *European Journal of Operational Research*, *229*(3), 673–682. http://doi.org/10.1016/j.ejor.2012.12.020

Montemanni, R., & Gambardella, L. M. (2009). An ant colony system for team orienteering problems with time windows. Retrieved February 16, 2016, from https://www.academia.edu/3931875/An_ant_colony_system_for_team_orienteering_problems_with_time_windows

Oliveira, J. A., Ferreira, J. A. O., Figueiredo, M., Dias, L. M. S., & Pereira, G. (2013, October 24). Comparação de dois algoritmos genéticos aplicados ao TOP. SGAPEIO. Retrieved from http://repositorium.sdum.uminho.pt/handle/1822/26198

Oliveira, J. A., Ferreira, J., Figueiredo, M., Dias, L., & Pereira, G. (2014). Sistema de Apoio à Decisão para o Transporte Não Urgente de Doentes em Veículo Partilhado. *Iberian Journal of Information Systems and Technologies*, (13), 17–33. http://doi.org/10.4304/risti.13.17-33

Resende, M. G. C., Toso, R. F., Gonçalves, J. F., & Silva, R. M. A. (2012). A biased random-key genetic algorithm for the Steiner triple covering problem. *Optimization Letters*, *6*(4), 605–619. http://doi.org/10.1007/s11590-011-0285-3

Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., & Van Oudheusden, D. (2010). A Path Relinking approach for the Team Orienteering Problem. *Computers & Operations Research*, *37*(11), 1853–1859. http://doi.org/10.1016/j.cor.2009.05.002

Tang, H., & Miller-Hooks, E. (2005). A TABU search heuristic for the team orienteering problem. *Computers & Operations Research*, *32*(6), 1379–1407. http://doi.org/10.1016/j.cor.2003.11.008

Toso, R. F., & Resende, M. G. C. (2014). A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, *30*(1), 1–15. http://doi.org/10.1080/10556788.2014.890197

Tricoire, F., Romauch, M., Doerner, K. F., & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, *37*(2), 351–367. http://doi.org/10.1016/j.cor.2009.05.012

Vansteenwegen, P., Souffriau, W., Berghe, G. Vanden, & Oudheusden, D. Van. (2009). A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, *196*(1), 118–127. http://doi.org/10.1016/j.ejor.2008.02.037

Vansteenwegen, P., Souffriau, W., & Oudheusden, D. Van. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, *209*(1), 1–10. http://doi.org/10.1016/j.ejor.2010.03.045

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers &*

Operations Research, 36(12), 3281–3290. http://doi.org/10.1016/j.cor.2009.03.008

ANEXO I – RESULTADOS DOS TESTES PARA INSTÂNCIAS ASSIMÉTRICAS

Anexo 1 - Resultados dos testes para instâncias assimétricas

Tabela 6 – Testes a instâncias assimétricas – Inserção em Série

	Serial																			
Instância	Run 1				Run 2				Run 3				Run 4				Run 5			
	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU
RD_100C_6	8228	54	75:15:00	00:03:00	8431	56	75:15:00	00:03:00	8704	58	76:02:00	00:03:00	8911	60	74:51:00	00:03:00	8535	56	74:52:00	00:03:00
RD_100C_7	9912	64	75:21:00	00:03:00	9392	59	73:52:00	00:03:00	8819	55	74:09:00	00:03:00	9883	62	75:04:00	00:03:00	8565	55	75:29:00	00:03:00
RD_100C_8	8661	59	72:14:00	00:04:00	7704	52	70:52:00	00:03:00	8231	56	72:37:00	00:03:00	8165	55	71:29:00	00:04:00	8326	56	72:24:00	00:03:00
RD_100C_9	8915	56	72:44:00	00:04:00	9114	58	72:24:00	00:03:00	9006	58	72:48:00	00:03:00	9145	57	74:48:00	00:04:00	8773	57	72:18:00	00:04:00
RD_100C_10	8292	53	73:20:00	00:04:00	8379	52	73:20:00	00:03:00	8273	54	73:04:00	00:04:00	8158	51	73:20:00	00:04:00	8759	55	73:40:00	00:04:00
RD_100C_11	9182	61	75:26:00	00:05:00	9042	61	74:58:00	00:04:00	8988	60	75:06:00	00:05:00	8575	56	73:46:00	00:05:00	8973	60	75:07:00	00:05:00
RD_100C_12	8581	56	73:21:00	00:05:00	8985	59	72:04:00	00:05:00	9002	61	72:14:00	00:05:00	8716	58	72:43:00	00:05:00	8912	59	71:41:00	00:06:00
RD_100C_13	8842	56	73:08:00	00:03:00	8920	57	73:48:00	00:03:00	8318	53	73:28:00	00:03:00	8779	58	72:20:00	00:03:00	9388	61	72:08:00	00:03:00
RD_100C_14	8697	56	74:20:00	00:04:00	8523	54	74:18:00	00:03:00	8703	56	74:25:00	00:03:00	8152	53	75:23:00	00:03:00	8422	54	74:32:00	00:03:00
RD_100C_15	8055	53	73:07:00	00:03:00	7981	53	73:07:00	00:04:00	7418	49	72:37:00	00:03:00	7723	52	72:09:00	00:04:00	8226	57	72:47:00	00:03:00

Tabela 7 – Testes a instâncias assimétricas – Inserção em Paralelo

	Parallel																			
Instância	Run 1				Run 2				Run 3				Run 4				Run 5			
	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU
RD_100C_6	8277	55	74:41:00	00:01:00	8526	57	76:17:00	00:01:00	9300	63	76:05:00	00:01:00	8795	58	74:31:00	00:01:00	8566	57	75:14:00	00:01:00
RD_100C_7	8817	55	74:17:00	00:01:00	10125	64	75:44:00	00:01:00	9061	57	73:20:00	00:01:00	9262	58	74:24:00	00:01:00	9130	58	73:26:00	00:01:00
RD_100C_8	8460	55	71:55:00	00:01:00	8035	55	71:13:00	00:01:00	8437	56	72:24:00	00:01:00	8001	53	70:52:00	00:01:00	8719	58	71:13:00	00:01:00
RD_100C_9	8973	56	73:50:00	00:01:00	9080	58	72:51:00	00:01:00	9048	58	74:02:00	00:01:00	9271	59	74:24:00	00:01:00	8856	56	71:11:00	00:01:00
RD_100C_10	9158	57	72:29:00	00:01:00	8218	51	73:03:00	00:01:00	8255	52	72:28:00	00:01:00	8343	53	71:31:00	00:01:00	8232	51	72:05:00	00:01:00
RD_100C_11	9012	59	75:02:00	00:02:00	8871	59	74:51:00	00:01:00	8777	57	75:18:00	00:01:00	8847	59	72:56:00	00:01:00	8936	58	75:00:00	00:01:00
RD_100C_12	8209	54	72:37:00	00:02:00	8615	58	73:38:00	00:02:00	9436	63	71:54:00	00:02:00	8653	57	72:43:00	00:02:00	8866	59	72:57:00	00:02:00
RD_100C_13	8308	54	72:29:00	00:01:00	8536	56	73:38:00	00:01:00	8843	56	74:06:00	00:01:00	8575	55	72:24:00	00:01:00	8484	54	72:06:00	00:01:00
RD_100C_14	9179	59	75:01:00	00:01:00	8368	55	75:35:00	00:01:00	8884	56	75:38:00	00:01:00	8702	55	74:40:00	00:01:00	8994	58	75:11:00	00:01:00
RD_100C_15	8216	54	73:20:00	00:01:00	8477	55	72:18:00	00:01:00	7769	53	73:53:00	00:01:00	7642	50	73:58:00	00:01:00	7049	46	72:40:00	00:01:00

Tabela 8 – Testes a instâncias assimétricas – Inserção de Menor Custo

	Low Cost																			
Instância	Run 1				Run 2				Run 3				Run 4				Run 5			
	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU
RD_100C_6	12036	80	75:28:00	00:11:00	12069	81	75:23:00	00:11:00	12255	81	74:50:00	00:10:00	11961	80	74:51:00	00:10:00	11905	80	74:24:00	00:10:00
RD_100C_7	12585	80	75:18:00	00:10:00	12560	81	75:13:00	00:11:00	12568	80	75:38:00	00:12:00	12546	80	76:41:00	00:11:00	12529	80	76:17:00	00:10:00
RD_100C_8	11567	77	73:07:00	00:10:00	11999	80	74:35:00	00:10:00	11577	78	73:43:00	00:10:00	11563	78	74:06:00	00:10:00	11687	77	73:44:00	00:10:00
RD_100C_9	12406	79	74:02:00	00:10:00	12284	80	74:49:00	00:10:00	12393	80	74:51:00	00:10:00	12312	79	74:30:00	00:10:00	12427	80	74:23:00	00:10:00
RD_100C_10	12290	77	74:03:00	00:10:00	11988	77	74:39:00	00:10:00	12173	77	74:11:00	00:10:00	12410	77	74:05:00	00:10:00	12201	77	73:13:00	00:09:00
RD_100C_11	12459	81	75:20:00	00:11:00	12080	80	75:14:00	00:11:00	12274	81	75:41:00	00:10:00	11989	80	74:38:00	00:14:00	12172	79	73:49:00	00:15:00
RD_100C_12	11804	77	74:17:00	00:14:00	11828	79	74:21:00	00:14:00	12018	79	74:11:00	00:16:00	11761	79	73:49:00	00:15:00	12121	79	74:21:00	00:17:00
RD_100C_13	12132	79	73:31:00	00:16:00	12153	79	74:22:00	00:12:00	12122	78	72:51:00	00:15:00	11907	77	73:53:00	00:15:00	11958	78	74:06:00	00:09:00
RD_100C_14	11748	77	74:51:00	00:09:00	11645	76	75:25:00	00:08:00	11980	77	75:30:00	00:09:00	12107	78	75:12:00	00:10:00	12056	77	75:37:00	00:11:00
RD_100C_15	11411	76	74:30:00	00:10:00	11682	78	74:49:00	00:10:00	11435	77	74:43:00	00:10:00	11227	75	74:59:00	00:10:00	11512	78	75:17:00	00:10:00

ANEXO II – RESULTADOS DOS TESTES PARA INSTÂNCIAS EUCLIDIANAS

Anexo 2 - Resultados para as instâncias euclidianas – modo release

Tabela 9 - Testes a instâncias euclidianas – Inserção em Série

	Serial																			
Instância	Run 1				Run 2				Run 3				Run 4				Run 5			
	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU
RD_100E_100C_1	10680	68	70:31:00	00:03:00	11170	72	70:44:00	00:04:00	11008	72	70:43:00	00:03:00	10520	69	71:02:00	00:04:00	11003	71	70:24:00	00:04:00
RD_100E_100C_2	12502	83	78:47:00	00:03:00	12242	82	79:04:00	00:03:00	12048	78	78:49:00	00:03:00	12534	81	78:26:00	00:03:00	12100	81	78:13:00	00:03:00
RD_100E_100C_3	12752	84	78:37:00	00:03:00	12905	86	78:09:00	00:03:00	12450	83	78:00:00	00:03:00	12726	85	78:30:00	00:03:00	12697	84	78:46:00	00:03:00
RD_100E_100C_4	11050	75	78:38:00	00:03:00	10968	73	78:26:00	00:04:00	10915	75	79:06:00	00:04:00	11752	79	78:52:00	00:04:00	11810	80	78:55:00	00:03:00
RD_100E_100C_5	12763	84	78:56:00	00:03:00	12309	80	78:47:00	00:03:00	12574	83	78:47:00	00:03:00	12622	82	78:56:00	00:03:00	12691	83	79:10:00	00:03:00
RD_100E_100C_6	12087	80	78:54:00	00:05:00	12422	83	78:25:00	00:03:00	12890	87	78:28:00	00:03:00	12567	84	78:24:00	00:03:00	12235	81	78:52:00	00:03:00
RD_100E_100C_7	11937	78	78:05:00	00:04:00	11691	78	79:07:00	00:04:00	11496	76	79:18:00	00:06:00	11423	75	79:03:00	00:06:00	11236	74	78:32:00	00:05:00
RD_100E_100C_8	12259	83	78:19:00	00:06:00	11397	77	78:41:00	00:05:00	11734	80	78:43:00	00:06:00	11387	77	79:02:00	00:05:00	11898	80	78:28:00	00:05:00
RD_100E_100C_9	12368	80	78:34:00	00:04:00	11965	76	79:02:00	00:03:00	12511	79	78:19:00	00:03:00	11959	76	78:52:00	00:03:00	12232	78	78:51:00	00:03:00
RD_100E_100C_10	10945	72	78:08:00	00:03:00	11786	79	78:18:00	00:03:00	11445	76	78:24:00	00:03:00	11639	78	78:47:00	00:03:00	11348	76	78:04:00	00:03:00

Tabela 10 – Testes a instâncias euclidianas – Inserção em Paralelo

Instância	Parallel																			
	Run 1				Run 2				Run 3				Run 4				Run 5			
	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU
RD_100E_100C_1	10667	69	70:56:00	00:01:00	11095	72	70:53:00	00:01:00	10834	70	70:32:00	00:01:00	11693	76	70:49:00	00:01:00	11591	73	70:57:00	00:01:00
RD_100E_100C_2	12615	84	78:45:00	00:01:00	12352	81	78:35:00	00:01:00	12520	83	78:58:00	00:01:00	12361	80	78:38:00	00:02:00	12211	80	78:44:00	00:01:00
RD_100E_100C_3	12656	83	79:08:00	00:01:00	13456	89	78:51:00	00:02:00	12826	86	78:26:00	00:02:00	13160	88	78:56:00	00:01:00	12855	86	79:17:00	00:01:00
RD_100E_100C_4	11772	80	78:43:00	00:01:00	11812	79	78:29:00	00:02:00	11529	78	78:50:00	00:01:00	10854	72	78:42:00	00:01:00	11613	79	79:12:00	00:01:00
RD_100E_100C_5	12866	84	79:04:00	00:01:00	12773	83	78:56:00	00:01:00	12773	83	78:35:00	00:01:00	13060	84	79:31:00	00:01:00	12651	83	78:51:00	00:01:00
RD_100E_100C_6	12317	82	79:05:00	00:01:00	12004	79	78:51:00	00:01:00	12597	84	79:07:00	00:01:00	12456	82	78:37:00	00:01:00	12495	84	78:58:00	00:01:00
RD_100E_100C_7	11744	77	78:49:00	00:01:00	11876	78	78:57:00	00:01:00	11834	79	78:57:00	00:01:00	11256	73	78:51:00	00:01:00	11786	77	79:10:00	00:01:00
RD_100E_100C_8	12197	82	78:54:00	00:01:00	12965	87	78:42:00	00:01:00	12292	83	79:04:00	00:01:00	11869	80	78:42:00	00:01:00	11998	81	79:13:00	00:01:00
RD_100E_100C_9	12919	83	78:15:00	00:01:00	13225	85	77:52:00	00:01:00	12050	78	78:14:00	00:01:00	12419	80	77:32:00	00:01:00	11771	76	79:22:00	00:01:00
RD_100E_100C_10	11397	76	78:51:00	00:01:00	11635	78	78:43:00	00:01:00	11623	76	78:50:00	00:01:00	12098	80	79:08:00	00:01:00	11769	78	79:07:00	00:01:00

Tabela 11 – Testes a instâncias euclidianas – Inserção de Menor Custo

Instância	Low Cost																			
	Run 1				Run 2				Run 3				Run 4				Run 5			
	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU	Fitness	Vértices (%)	Viagens (h)	CPU
RD_100E_100C_1	11970	78	71:30:00	00:12:00	11789	76	71:16:00	00:11:00	11005	72	71:06:00	00:10:00	12163	80	71:41:00	00:11:00	11146	73	71:19:00	00:10:00
RD_100E_100C_2	12617	84	79:25:00	00:17:00	12801	85	79:10:00	00:16:00	13391	89	77:33:00	00:15:00	13213	87	79:24:00	00:14:00	12779	85	79:00:00	00:13:00
RD_100E_100C_3	13251	90	79:16:00	00:14:00	13338	89	78:49:00	00:14:00	13481	90	79:15:00	00:14:00	13026	87	79:28:00	00:16:00	13175	87	79:14:00	00:15:00
RD_100E_100C_4	11685	80	79:09:00	00:11:00	11744	79	79:18:00	00:16:00	12176	82	79:04:00	00:11:00	12011	83	79:16:00	00:10:00	11874	81	79:35:00	00:10:00
RD_100E_100C_5	13137	85	79:25:00	00:13:00	12607	82	79:35:00	00:13:00	12773	84	79:25:00	00:14:00	12480	82	79:42:00	00:13:00	12503	82	79:32:00	00:13:00
RD_100E_100C_6	12796	86	79:12:00	00:12:00	12582	82	79:33:00	00:12:00	12732	86	79:42:00	00:12:00	12718	86	79:22:00	00:12:00	12634	84	79:42:00	00:12:00
RD_100E_100C_7	11938	78	79:25:00	00:11:00	12187	81	79:21:00	00:11:00	12056	79	78:59:00	00:11:00	12407	82	79:03:00	00:11:00	12080	80	79:25:00	00:11:00
RD_100E_100C_8	12559	85	79:27:00	00:12:00	12398	84	79:30:00	00:12:00	12280	83	79:31:00	00:13:00	12411	84	78:40:00	00:13:00	12112	83	79:23:00	00:12:00
RD_100E_100C_9	13330	86	78:54:00	00:14:00	12511	81	79:18:00	00:15:00	13437	86	79:07:00	00:14:00	13110	84	79:31:00	00:13:00	12802	83	79:07:00	00:12:00
RD_100E_100C_10	12189	81	78:54:00	00:12:00	12321	82	79:06:00	00:12:00	12402	83	79:04:00	00:12:00	12565	83	78:37:00	00:12:00	12218	82	79:04:00	00:12:00