Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática

SI - Computação Natural

TP3
Algoritmos Genéticos

Grupo 2:
André Gonçalves, A75625
Miguel Miranda, A74726
Rogério Moreira, A74634
Tiago Sá, A71835

Braga, 6 de Maio de 2018

Conteúdo

Resumo

O design de sistemas de distribuição gera problemas de otimização com um elevado conjunto de soluções válidas, que crescem exponencialmente à medida que mais restrições ou pontos de entrega entram no problema.

Estes contextos, enquadrados em problemas de localização e alocação de veículos (LRP), são assim cenários de relevo, uma vez que a necessidade de redução de custos, associados com a logística e entrega de produtos aos clientes, está presente em qualquer cadeia de distribuição.

Neste sentido, o presente projeto aborda a resolução de *LRPs* explorando as vantagens dos algoritmos genéticos na obtenção de soluções viáveis. As características evolutivas destes algoritmo, permitem a partir de uma população de soluções inicial aleatórias, aprimorar e evoluir para novas gerações de soluções válidas e otimizadas para o domínio do problema.

Como resultados, serão apresentadas as soluções encontradas para os conjuntos de instâncias em estudo, apresentado qual a melhor topologia e valores para as variáveis do problema. O algoritmo implementado, além de genérico para qualquer outro contexto de *LRP* com a mesma estrutura de variáveis, permite obter uma população de soluções válidas em tempos de processamento aceitáveis, podendo assim ser aplicado em domínios práticos de utilização.

1. Introdução

Algoritmos Genéticos apresentam-se como métodos de procura de soluções, construidos sobre os conceitos chave da teoria evolutiva, procurando assim implementar num algoritmo computacional as etapas que ao longo dos séculos permitem a evolução e adaptação de uma espécie biológica.

Estes algoritmos apresentam como vantagens o facto de operarem sobre uma população de soluções válidas, que são adaptadas e melhoradas ao longo das sucessivas iterações. No final das várias gerações, é possível obter não apenas uma melhor solução, mas sim uma população final de soluções válidas para o problema.

Focando o contexto de problemas LRP *Location Routing Problems*, pretende-se encontrar um sistema de distribuição capaz satisfazer a procura de um conjunto de clientes conhecidos. Contudo, a solução deve utilizar o menor número de depósitos e veículos possível, de forma a minimizar os custos associados com a logística e transporte do stock até aos clientes.

Devido ao número de variáveis e restrições envolvidas, o número de soluções total destes problemas cresce exponencialmente, sendo geralmente impossível (em tempos de processamento aceitáveis) computar todas as rotas e alternativas possíveis, no sentido de encontrar a melhor solução.

Uma vantagem da utilização de algoritmos genéticos na resolução de problema LRP recai exatamente na existência de uma população de soluções. Ao serem inicialmente criadas de forma aleatória, os indivíduos da população representam um conjunto de rotas distintas e válidas, que quando combinadas ao longos das gerações permitem derivar soluções gradualmente melhores.

Eventualmente, estas soluções serão suficientemente próximas da solução ótima, com a vantagem de serem obtidas com recursos de processamento e tempo aceitáveis. Apesar de não se garantir que a solução seja a ótima, o seu desempenho é suficientemente bom para ser aceite como uma solução válida.

Como estrutura do presente relatório: O capitulo 2 apresenta um introdução inicial aos conceitos base relacionados com a temática de algoritmos genéticos; o capítulo 3 apresenta uma descrição detalhado do problema *LRP* clássico, assim como uma descrição do contexto e restrições das instâncias em análise; o capitulo 4 descreve numa vertente conceptual a modelação de um cromossoma e da função de *fitness* para o contexto do problema LRP e, o capitulo 5 realiza a apresentação dos resultados obtidos, descrevendo os custos e rotas descritas pelas melhores soluções obtidas.

Por fim, o capitulo 7 apresenta a conclusões finais do projeto desenvolvido.

2. Algoritmos Genéticos

Os algoritmos genéticos representam métodos de procura de soluções, geralmente aplicados em problemas de otimização ou aprendizagem. Estes processos baseiam-se nos conceitos da teoria da evolução, apresentada por Darwin.

De uma forma geral, a partir de uma população inicial que representa um conjunto de possíveis soluções para o problema, os elementos com maior probabilidade de maximizar a solução são selecionados. A partir destes são geradas novas gerações de soluções, cada vez melhores e mais adaptadas ao contexto do problema [4, 3].

À medida que novas gerações são criadas, as principais características de uma geração são combinadas e passadas à seguinte, criando assim novos indivíduos, com melhor adaptação ao problema.

Quando o algoritmo convergir, será possível obter não só uma solução para o problema, mas sim um conjunto de soluções válidas para o problema. Cada elemento da geração final representa assim uma possível solução válida no contexto.

Sendo estes algoritmos regidos por probabilidades, repetições da execução do algoritmo podem gerar soluções diferentes. Apesar de não ser possível garantir que as soluções obtidas sejam uma solução ótima para o problema, pode assumir-se que para o tempo de convergência do algoritmo, as soluções encontradas serão pelo menos próximas da solução ótima.

O seguinte excerto de pseudo-código procura esboçar as fases de um algoritmo genético.

```
timer t = 0;                                    //Iniciar contador tempo a 0
P(t) = inicia_populacao(t);                     //Gerar aleatoriamente a população inicial
cond_paragem = avaliar_populacao( P(t) );       //Avaliar a população inicial

while( !cond_paragem ){                         //Enquando nao se verifica o critério paragem...
    P(t+1) = seleciona_pais( P(t) );            //Selecionar os elementos com maior pontuação
    P(t+1) = reproduzir_e_mutar( P(t+1) );      //Aplicar operadores genéticos para gerar a nova geração
    t = t++;                                    //Incrementar o contador de tempo
    cond_paragem = avaliar_populacao( P(t+1) );
}
```

Figura 2.1: Esboço da estrutura do código de um algoritmo genético

2.1 Codificação de soluções

Considerando que um algoritmo genético é independente da área e contexto dos problemas onde é aplicado, a forma das soluções por si encontradas é uma representação dos parâmetros que caracterizam uma solução real para um problema. Esta codificação das variáveis de uma solução num elemento capaz de ser interpretado e processado pelo algoritmo, dá origem a um cromossoma.

Cada cromossoma contem assim informação relativa à solução candidata que representa. Esta codificação num cromossoma é geralmente uma cadeia binária, de 0s ou 1s, podendo em algumas adaptações mais recentes do algoritmo ser também uma sequencia de números inteiros, strings ou até mesmo árvores de decisões. Tal como na biologia, o conteúdo de um cromossoma é designado por genótipo e a interpretação semântica do seu conteúdo designa-se por fenótipo. Esta interpretação semântica será uma combinação do conteúdo de um cromossoma com o contexto do problema [1, 2].

No caso de uma representação binária, um bit ou subconjuntos de bits podem representar alternativas sobre um determinado parâmetro da solução. Considerando como exemplo um problema de otimização no embalamento de produtos, podemos ter quatro variáveis na decisão:

Material a embalar	Espessura (cm)	Material lacrar caixa	Pegas laterais
00 – Ferro 01 – Alumínio 10 – Madeira 11 – Papel	$L \in [1,7], L\ inteiro$ 001 – 1cm 010 – 2cm ... 110 – 6cm 111 – 7cm	00 – Cola 01 – Fita cola 10 – Plástico	1 – Sim 0 – Não

Figura 2.2: Exemplo de parâmetros a decidir no embalamento de produtos.

Com o exemplo apresentado, qualquer cromossoma que represente uma solução candidata será uma sequencia binária de 8 bits. Por exemplo, o embalamento de uma placa de alumínio, com 5cm de espessura, cuja caixa é fechada com fita cola e apresenta pegas laterais dará origem ao cromossoma cuja sequencia binária é **01 011 01 1**.

Novamente, cabe posteriormente à função de *fitness* ser capaz de "partir"e interpretar individualmente cada sub parte do cromossoma com informação relativa a uma determinada propriedade de decisão.

2.2 População Inicial

O primeiro passo na implementação do algoritmo é a criação de uma população inicial. Os elementos que constituam esta população inicial devem ser gerados de forma aleatória, criando assim alguma diversidade de soluções [1]. Tal como no suporte da teoria da evolução biológica, este fator de diversidade entre as soluções é essencial para descrever diferentes hipóteses de solução e criar futuras gerações que gradualmente se vão adaptando e especializando como solução do problema. Se a dimensão da população inicial for reduzida, é cortada a hipótese de criar sucessivas gerações e o algoritmo converge prematuramente, gerando soluções limitadas e pouco ótimas. Em oposição, populações iniciais vastas e com muitos casos aumentam consideravelmente o tempo de execução do algoritmo.

Tendo um conjunto de soluções iniciais, o algoritmo pode assim aplicar os métodos de evolução: seleção e reprodução dos elementos mais favoráveis e aplicação de mutações para manter a diversidade entre gerações.

2.3 Seleção de Soluções

A cada iteração, cada um dos elementos da população é analisado e avaliado a nível de qualidade enquanto possível solução para o problema a tratar. Esta medida é obtida através de uma função de avaliação, que deve ter em consideração o contexto específico do problema e privilegia os indivíduos melhor adaptados, ou seja, quanto mais próximo for um individuo de uma solução ótima, melhor é a avaliação atribuída pela função de avaliação a esse indivíduo.

Esta função objetivo, também designada por função de *fitness*, pode envolver o uso de simuladores com a solução em análise e é geralmente complexa de calcular.

Os indivíduos com maiores medidas, têm assim maior probabilidade de serem escolhidos e se "reproduzirem", originando uma descendência que combina as suas características.

Dos diversos métodos de seleção, destacam-se [2]:

- **Método da roleta** : Depois de calculado o valor da função *fitness* para cada um dos cromossomas da população, a cada um deles é atribuída uma porção da roleta, conforme a proporção da sua pontuação com a soma de todos as pontuações da geração. Para que cada indivíduo receba uma porção da roleta, é necessário que a função de *fitness* retorne sempre um valor positivo superior a zero.

  Neste método, a probabilidade de reprodução será assim tanto maior quanto o valor dado pela função de *fitness*. No caso de um cromossoma ter uma classificação muito elevada, pode levar a que o mesmo ocupe a maior parte de roleta e, portanto, seja sempre o escolhido para reprodução. Esta situação levaria à perda de diversidade genética.

- **Método Elitista** : Devido aos operadores genéticos que se vão aplicar para gerar novos cromossomas, pode acontecer que as melhores características dos parentes da geração a ser criada sejam perdidos na sua descendência.

  Para contornar este fator e evitar a perda de diversidade genética, existem métodos de seleção que transportam para a nova geração alguns dos melhores cromossomas da geração anterior, garantindo assim a presença das melhores soluções nas gerações futuras.

- **Método Torneio** : Perante a população de soluções, o método de seleção por torneio escolhe um conjunto de **N** cromossomas e é escolhido para reprodução o indivíduo desse sub grupo com melhor pontuação segundo a função de avaliação.

  Depois de selecionado o melhor indivíduo, os elementos são "repostos"na geração atual e outros subconjuntos de **N** elementos são retirados aleatoriamente, até se ter o número suficiente de cromossomas para a fase seguinte de reprodução.

  Se este parâmetro **N** for demasiado grande, a seleção de um grande número de elementos da população em simultâneo, pode levar à perda de variabi-

lidade genética, porque o indivíduo com a maior pontuação do subconjunto vai-se sobrepor a outros elementos vantajosos desse sub conjunto.

2.4 Reprodução

Organizando os cromossomas selecionados por pares para reprodução, é aplicada a técnica de *crossover* através da definição de pelo menos um ponto de corte. A posição de corte representa assim a zona onde os dois cromossomas irão trocar as suas características, gerando assim uma descendência que os combine.

No caso de cromossomas com representação binária, a posição de corte pode tirar significado ao cromossoma, sendo assim um processo de verificação e correção dos cromossomas gerados.

A figura seguinte procura esquematizar a criação de duas gerações através da técnica de reprodução por *crossover*. No primeiro esquema a descendência é criada por um plano de corte e, no segundo exemplo, por dois planos de corte.

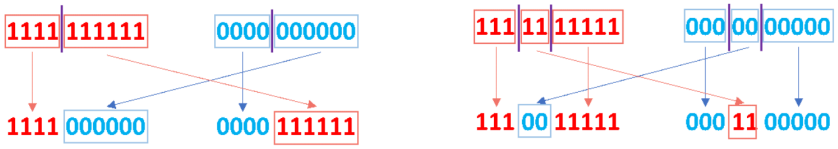

Figura 2.3: – Exemplos da técnica de reprodução por crossover.

2.5 Mutação

Além da diversidade gerada pelo cruzamento dos melhores cromossomas na fase de reprodução, é ainda aplicada uma fase de mutação a alguns elementos da nova população. Tal como no contexto biológico, as mutações ocorrem de forma extraordinária e rara e procuram dotar os indivíduos de novas características, possivelmente vantajosas para a sua adaptação ao meio.

No contexto dos algoritmos genéticos, a aplicação de mutações a cada cromossoma tem uma probabilidade baixa de acontecer, porque elevadas taxas de mutação poderiam levar à perda de boas características herdadas previamente pelos cromossomas progenitores.

No caso de uma codificação binária, uma mutação corresponde à alteração aleatória do valor de um dos seus bits. Por exemplo um cromossoma com a sequencia binária **11010110** pode mutar-se para **11010010**. Cada bit de um determinado cromossoma tem a mesma probabilidade de ser mutado.

2.6 Capacidade de Aprendizagem

A capacidade de aprendizagem nos algoritmos genéticos pode ser analisada na fase de reprodução do mesmo. Considerando que só os melhores indivíduos são selecionados para esta fase, um par de cromossomas selecionado para o cruzamento vai dar origem a duas novas soluções que irão pertencer à geração futura.

Estas novas soluções geradas, vão herdar uma fusão das características dos seus progenitores, adquirindo dos melhores indivíduos das gerações passadas aquilo que pode ser visto como o seu conhecimento.

Com este processo, as soluções que se encontram na última iteração do algoritmo, vão então conter as principais e melhores características de todos os cromossomas que fizeram parte da sua antecedência, adquirindo e aprimorando o seu conhecimento ao longo das gerações.

3. Analise do Problema

3.1 Location Routing Problem

Contextos relacionados com a alocação de veículos em rotas de distribuição de produtos (LRP, do inglês *Location Routing Problem*), revelam-se como uns dos problemas habituais no estudo de processos de otimização baseada em combinações.

Nestes problemas, a rede de distribuição encontrada deve ser capaz de realizar o atendimento de um conjunto de clientes, por intermédio de uma frota de veículos que apresentam como ponto de partida e término um determinado depósito.

As decisões passam assim por determinar quais os depósitos que efetivamente devem ser abertos e, nesses, decidir qual o número necessário de veículos que o mesmo deve possuir, assim como a rota que cada veículo realiza no atendimento aos clientes. Apesar do enunciado de um problema de *LRP* parecer simples, um problema com apenas estes três domínios de decisão (número de depósitos, número de veículos e rota dos veículos pelos clientes) apresenta um nível de soluções que cresce exponencialmente com o número máximo de cada um destes parametros referidos.

Como consequência, na grande maioria dos contextos práticos é impossível descobrir uma solução ótima para estes problemas, uma vez que não é possível computar e comparar todas as soluções possíveis no domínio (pelo menos num tempo e com recursos computacionais aceitáveis).

Por este motivo, diversas abordagens heurísticas ganham relevo na tentativa de resolução destes problemas. Sem precisarem de explorar todo o espectro de soluções possível, estas técnicas são capazes de apresentar uma solução suficientemente próxima da ótima, em tempos de processamento reduzidos. Algoritmos como *Ants Colony, Simulated Annealing, Tabu Search, GRASP search* e algoritmos genéticos são alguns exemplos de procedimentos capazes de resolver estes problemas.

Sendo o foco deste relatório, os algoritmos genéticos revelam-se um bom método para a resolução deste tipo de problemas, uma vez que os seus métodos de criação, mutação e cruzamento de soluções se baseiam sobre procedimentos e decisões probabilísticas. A evolução das soluções a cada geração permite rapidamente encontrar soluções próximas da ótima, em tempo de execução bastante aceitáveis.

3.2 Problema em Estudo

No sentido de aplicar as características de um algoritmo genético num problema de otimização, foi explorada uma variante do problema de localização e encaminhamento de veículo *Location routing problem - LRP*.

No problema em estudo, o desafio passa por descobrir uma rede de distribuição de stocks entre depósitos e clientes, de tal forma que a procura de todos os clientes seja satisfeita e a empresa acarrete o menor custo possível na distribuição dos produtos. A nível de comparação, para uma situação limite com apenas um depósito, um camião e N clientes, o problema seria reduzido a uma variante do caixeiro viajante, onde o percurso do camião (caixeiro) representa um percurso próximo do ótimo, capaz de passar por todos os clientes (cidades).

De uma forma geral, procura-se encontrar um conjunto de rotas que garantam que:

- Seja aberto o menor número de depósitos (armazéns stock) possíveis, para satisfazer a procura dos clientes;
- Cada deposito aberto utilize o menor número de veículos na sua frota;
- Os percursos percorridos pelos veículos utilizados (de todos os depósitos abertos) têm que ser capazes de:
  - Passar por todos os clientes uma única vez;
  - Satisfazer a procura de todos os clientes;

Nas restantes restrições do problema, entram ainda parâmetros simplificados como:

- A abertura de um depósito apresenta um custo de abertura do mesmo.

  Este custos podem, por exemplo, ser associados à construção do deposito, deslocação de mercadorias para o deposito, etc;

- A utilização efetiva de um veículo numa rota de distribuição acarreta um custo de uso do veículo.

  Estes custos podem ser associados, por exemplo, a custos de combustível, manutenção veículo, guias de transporte de mercadoria, etc;

- Na realização de uma rota, qualquer percurso apresenta o mesmo custo.

  Ou seja, além da distância entre dois pontos do mapa ser simplesmente a distância euclidiana entre os pontos, o custo por distância percorrida é o mesmo entre qualquer ponto dos grafos finais (Na realidade rotas diferentes têm custos distintos, como por exemplo percurso por auto-estradas *vs* caminhos rurais).

  Por não haver informações sobre o custo por "quilometro"este foi assumido como 1U.M. Assim, se um camião percorrer um percurso com distância 25.4, o custo desta deslocação po veículo será de 25.4U.M.

- Na avaliação de uma rota, caso o veículo passe por um cliente tendo já esgotado a sua capacidade de transporte, a solução incorre de um custo adicional

ϵ, usado como forma de penalizar a solução na função de fitness do algoritmo genético.

Apesar da descrição do problema ser uma modelação bastante simplificada da realidade, a sua resolução apresenta ainda alguma complexidade. Esta complexidade não está intrinsecamente ligada com as restrições do problema em si, mas com o crescimento exponencial de hipóteses possíveis, à medida que o número de depósitos e clientes aumenta.

Como consequência, na grande maioria dos problemas práticos que procuram resolver *LRP* não é viável a nível de tempo e recursos computacionais computar todas as soluções possíveis, de forma a saber qual a que apresenta menores custos de gestão. O interesse em explorar algoritmos genéticos nestes problemas recai assim neste fator, como forma de encontrar uma solução eventualmente próxima da ótima, num tempo de execução aceitável do algoritmo.

Tal como referido na secção 2.1, a codificação da estrutura de um cromossoma e a construção da função de *fitness* são geralmente as únicas etapas de um algoritmo genético que se relacionam com o contexto do problema. Nas restantes fases do algoritmo, processos como a mutação ou *crossover* estão abstraídos acima da estrutura do cromossoma, funcionando para qualquer novo problema com as mesmas restrições e estruturas.

Uma vez que os conjuntos de instâncias em estudo estão igualmente estruturadas, o significado atribuído à estrutura do cromossoma e as restrições avaliadas na função de *fitness*, são assim iguais para qualquer enunciado. O algoritmo genético construido é assim modular para qualquer um destes datasets, sem necessitar de nenhum alteração.

Como datasets, foram realizados testes sobre três *datasets* vulgarmente utilizados em problemas no domínio de *LRP*:

- ***Prins et al - Prodhon***, com 30 variantes do problema, indicando a capacidade dos veículos e depósitos;
- ***Tuzun and Burke***, com 36 variantes apenas indicando a capacidade dos veículos. A capacidade de cada depósito é assumida como igual ao sumatorio da procura de todos os clientes;
- ***Barreto***, com 14 variantes do problema, indicando a capacidade dos veículos e depósitos.

Cada um dos ficheiros (variantes) apresenta um topologia distinta para o número e as posições dos clientes e depósitos, variando também parâmetros como a procura de cada cliente, capacidades dos depósitos e capacidades dos veículos.

3.3 Pré-Processamento dos Dados

Como em qualquer projeto em que sejam usados dados externos, é necessária uma fase de estudo inicial dos mesmos, de modo a compreender a sua estrutura e significado dos seus atributos. Neste sentido, foi realizada uma etapa inicial de pré-processamento, para garantir que os dados contidos no formato *txt* pudessem ser convertidos numa estrutura *json* mais consistente e versátil. Este novo formato

é assim possível de ser reutilizado em qualquer ambiente de programação, sem necessitar de novo pré-processamento ou adaptações, tendo ainda a vantagens de já com o nome das variáveis incluído dentro de si.

Apesar das três instancias fornecidas (ProdHon, Barreto e Tuzun) se encontrarem devidamente documentadas a nível da sua estrutura, o formato *txt* obrigaria a desenvolver um parsing/interpretador que provavelmente não seria genérico entre linguagens. Ao usar o formato *json*, qualquer ficheiro pode ser lido em praticamente todos os ambientes de programação, uniformizando assim o acesso aos dados.

No projeto em estudo, foi tomada a decisão de desenvolver o problema de algoritmos genéticos na resolução de *LRP* na plataforma *R*, sendo usado o package *jsonlite* para ler e aceder facilmente aos campos destes ficheiros *json* criados.

A nível da estrutura dos ficheiros *json*, cada um identifica claramente as três principais categorias de informação dos ficheiros (Figura 3.1):

- Os clientes, onde para cada um está descrita a sua posição (x,y) e a respetiva quantidade de procura;
- Os depósitos, em que para cada depósito está descrito a sua posição (x,y), a sua capacidade e o seu custo de abertura ;
- Restantes dados genéricos aos problema: número de clientes, número depósitos, capacidade de cada veículo e o custo de cada veículo.

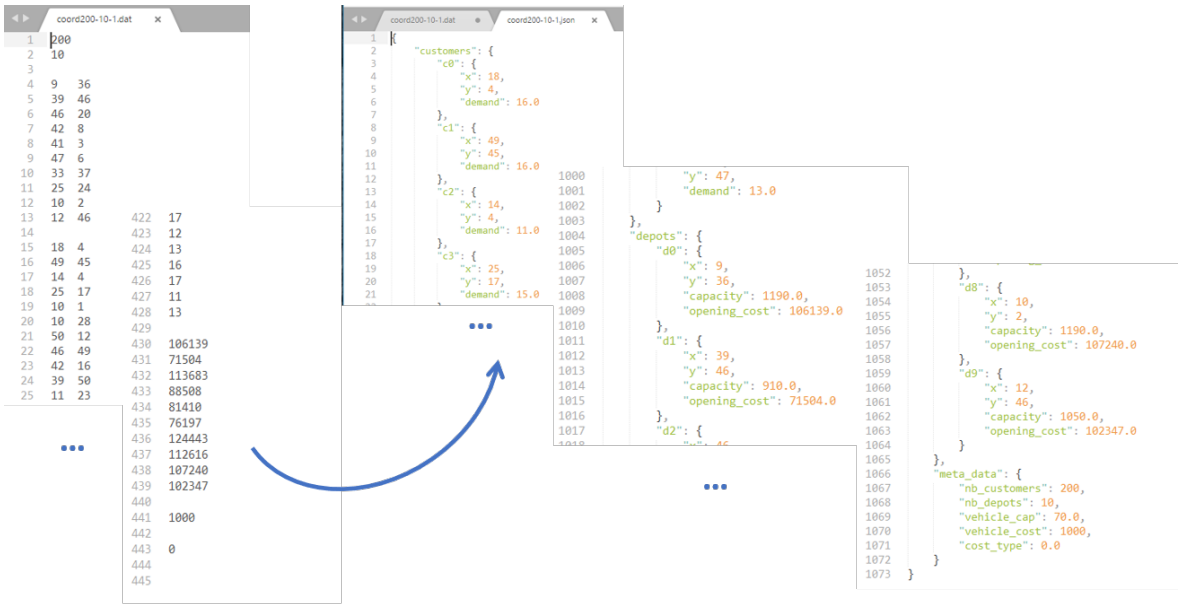

Figura 3.1: Exemplo do formato *json* após transformação de um ficheiro *Prodhon coord200101*.

Esta fase de pré-processamento foi automatizada num scrip desenvolvido em *python*, onde o ficheiro *.txt* de input é transformado numa estrutura *json* com os campos referidos. Para além da transformação referida, o script cria ainda um gráfico com as posições dos depósitos e dos clientes assinaladas, como forma de

observar num referencial as posições e a densidades dos clientes nas regiões do mapa (Figura 3.2).

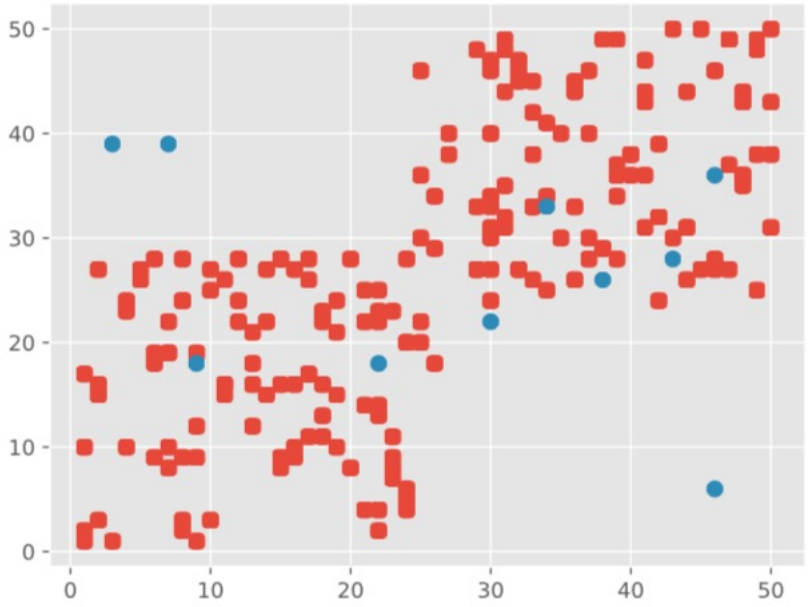

Figura 3.2: Exemplo do plot projetado, com a posição dos clientes (a vermelho) e depósitos (a azul).

4. Aplicação de Alg. Genéticos em LRP

4.1 Estrutura do cromossoma

Como referido na secção 2.1 a estrutura de um cromossoma é geralmente representada na forma de uma cadeia binária ou numérica. Por este motivo, a função de *fitness* deve ter em conta a estrutura abstrata atribuída a cada posição (ou subconjunto de posições) dos cromossomas de uma população.

Nesse sentido, é necessário "partir"e interpretar individualmente cada sub parte do cromossoma, que contem efetivamente informação relativa a uma determinada propriedade de decisão do problema. Para a estruturação de um cromossoma no problema de *LRP* em estudo, é tida como base a forma de um cromossoma num problema semanalmente de *vehicle routing* - *VRP*, com apenas 1 depósito e N clientes (Figura 4.1). Nesta codificação (para o exemplo com 1 deposito e N Clientes):

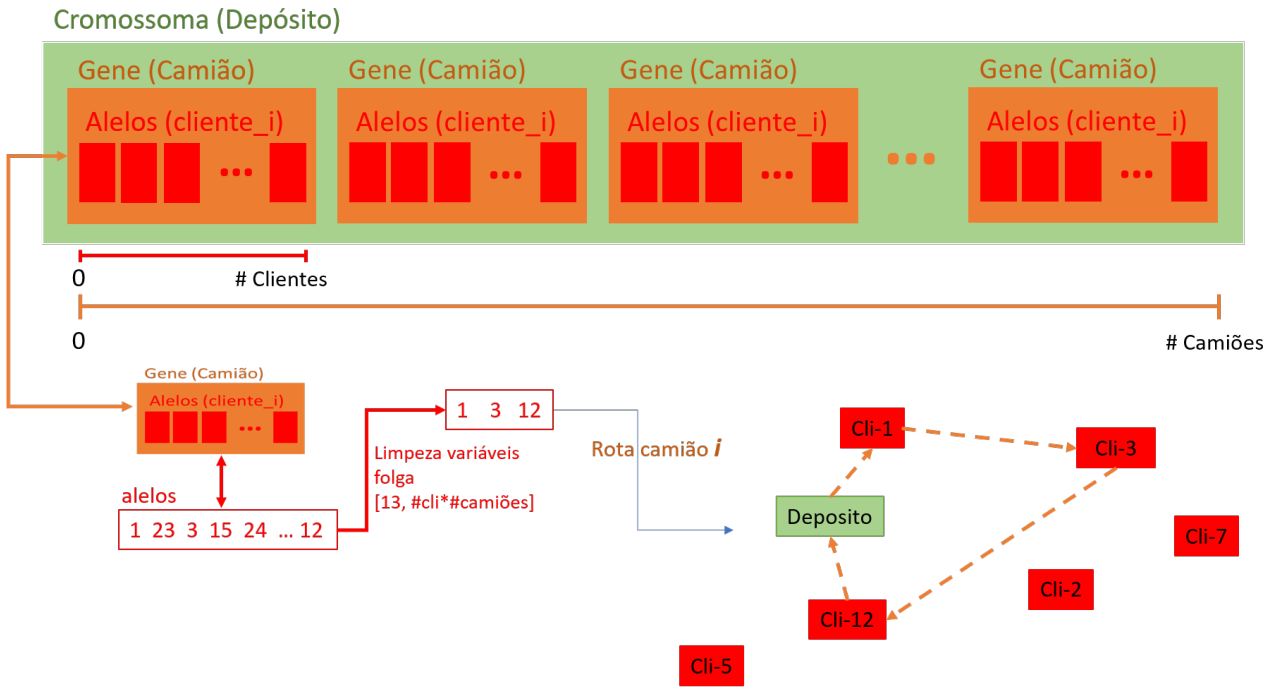

Figura 4.1: Esquema conceptual de um cromossoma para um problema VRP, com 1 deposito e N clientes.

- Um cromossoma representa as decisões sobre um deposito;
- O cromossoma terá tantos genes quanto o número de camiões a utilizar.

  No pior cenário, existiriam tantos camiões quanto o número de clientes, onde

cada cliente é atendido por um camião exclusivo a si e o camião realiza um percurso que passa apenas por um único cliente.

Contudo, existindo informação sobre a capacidade de stock do camião e da procura total dos clientes, pode ser calculado o número mínimo de camiões necessários para satisfazer a procura. Assim, o número suficiente de camiões é dado pela formula 4.1. Ao reduzir o número de camiões, cada camião vai (como seria de esperar) passar idealmente por vários clientes, tirando assim partido da quantidade de carga que consegue transportar.

$$nr\_ideal\_veculos = ceiling(\frac{\sum_{i=1}^{nr_clientes} ProcuraCliente\_i}{capacidadeVeiculo}) \tag{4.1}$$

- Cada gene, que representa o percurso de um camião, terá na pior hipótese tantos alelos quanto o número de clientes.

  Os alelos que estiverem corretamente marcados (com indices válidos), indicam a ordem e os clientes pelos quais o camião irá passar, sendo que um camião não tem necessariamente que passar por todos os clientes. Se nenhum dos alelos for válido, então o camião não é utilizado e não acarreta custos para a solução final.

  Da mesma forma que é possível otimizar o número de camiões, um procedimento semelhante pode ser feito para determinar o número de alelos ótimo (formula 4.2).

$$nr\_ideal\_alelos = floor(\frac{capacidadeVeiculo}{ceiling(mean(ProcuraCliente\_i))} \tag{4.2}$$

  Neste caso, a partir da capacidade do camião (constante entre camiões) e a procura média dos clientes, é possível saber qual o número máximo de clientes que o camião consegue satisfazer no seu percurso, até esgotar a sua carga. Deste modo, se a carga do camião só conseguir satisfazer em média 5 clientes, não faz sentido o seu percurso ter uma passagem por mais do que 5 clientes.

  Em lugar da função média, podem ainda ser utilizadas eventualmente funções como o mínimo, máximo ou mediana da procura dos clientes.

- Cada alelo representa a possível passagem por um cliente e será um valor inteiro entre [1, nr_alelos * nr_camiões] (no exemplo da Figura 4.1).

  Ao utilizar um valor inteiro dentro do intervalo de valores referido:

  - Os valores entre [1, nr_clientes] identificam os clientes.
    Se o alelo tiver um valor dentro do intervalo acima referido, então o camião passa pelo cliente representado por esse valor inteiro;
  - Os valores entre [nr_clientes, nr_alelos * nr_camiões] podem ser vistos como variáveis de folga, que identificam a **não** passagem do camião pelo cliente do índice/ alelo_i;

- Garante-se que se servem todos os clientes na totalidade, porque todos os valores que identificam os clientes ([1, nr_clientes]) encontram-se no cromossoma;
- Garante-se que nenhum camião passa por um cliente já atendido por outro camião, porque se um valor aparecer num determinado gene, não se repete em alelos (clientes) de mais nenhum outro gene (camião);
- Não existe a possibilidade de um camião terminar ou passar o seu percurso por outro depósito, dado que essa informação/possibilidade não está sequer incluída nos índices representados pelos alelos.

Relativamente à informação completa de um cromossoma, o mesmo pode ser visto como uma lista de valores, que é reordenada e combinada de forma a criar novas soluções. Por este motivo, na utilização do algoritmo genético do package *GA* para *R* é utilizado o tipo *Permutation* que lida com todas as questões associadas com a mutação e crossover deste tipo de cromossomas.

Como referido na listagem anterior, o valor de um alelo indica a passagem, ou não, de um camião pelo cliente representando pelo índice do respetivo alelo.

Conceptualmente a não passagem por um cliente poderia ser marcada com o valor (0), ou (-1). Contudo, isto iria criar vários valores repetidos dentro do cromossoma o que, para as funções de mutação e crossover *default* do método *GA* iria levar à possível remoção de todas as ocorrência do valor (0) ou (-1) dentro de uma sub parte do cromossoma. Isto iria causar com que, na fase de criação de uma nova geração, um cromossoma pudesse reduzir de tamanho (devido aos valores repetidos removidos) e com isso perder significado no contexto do problema.

Ao utilizar variáveis de folga dentro da gama $[-\infty, 0]$ ou $[nr\_clientes, \infty]$, é assim possível marcar a não passagem do camião por clientes, sem ser necessário alterar os métodos de mutação e crossover oferecidos pelo método *GA*.

O esquema inferior da Figura 4.1 procura esquematizar de que forma é realizada a limpeza das informações dos alelos, retirando as variáveis de folga e ficando apenas com as informações relativas aos clientes efetivamente visitados pelo camião representado pelo gene em análise.

Tendo descrito a estrutura a considerações utilizadas para modelar um cromossoma no caso de existir apenas 1 depósito, estes conceitos podem ser generalizados para N depósitos. Para isso, basta imaginar um conjunto de N cromossomas iguais aos apresentados na descrição anterior, cada um referente a um depósito que pode, ou não, ser aberto.

A nível de nomenclatura, apesar dos algoritmos genéticos operarem sobre cromossomas, foi tomada a decisão de designar por cromossoma à informação relativa a um depósito. A informação relativa a todos os depósitos, forma aquilo que foi designado como **genoma**, ilustrado na Figura 4.1

Será sobre a estrutura chamada de genoma (conjunto de cromossomas referentes a N depósitos) que o algoritmo irá operar. Na secção seguinte será abordado de forma genérica de que forma se interpretam as informações de cromossoma, na perspetiva da função de fitness.

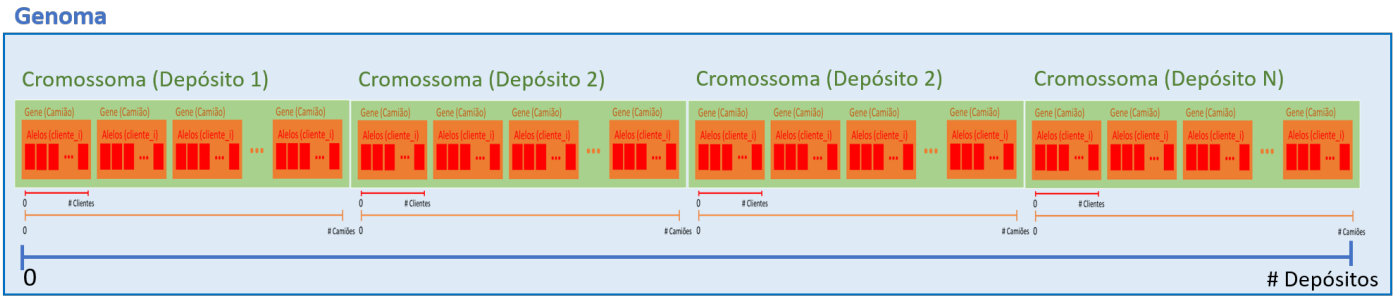

Figura 4.2: Esquema conceptual de um cromossoma no problema LRP em análise

4.2 Função de Fitness

Tendo sido abordado com detalhe a estrutura atribuída ao genoma -conjunto de cromossomas (depósitos)- que o algoritmo genético irá evoluir, será nesta secção abordado de que forma é avaliado o desempenho (fitness) de cada solução. Apesar do problema ser implementado de forma genérica, de tal como que os métodos de geração, mutação e crossover de soluções não sejam necessários de adaptar ao contexto, a função de avaliação deve ter em conta o formato abstrato atribuído ao genoma.

Num sentido prático, a função de fitness indica uma expressão cujo valor se procura maximizar, uma vez que a equação se encontra escrita na forma $\frac{1}{custo\_solucao}$. À medida que os indivíduos das novas gerações da população apresentam novas alternativas de rotas, o valor desta função deve ser preferencialmente menor.

```
# Fitness function to be minimized
lrpFitness <- function(genoma, customerDistances,
↪  customerDemand, depositosDistances) {

    1 / CustoTotal(genoma, customerDistances, customerDemand,
    ↪  depositosDistances);
}
```

Relativamente à função que calcula o custo total de um cromossoma, a sua definição baseia-se na iteração de três ciclos, que "partem"o genoma respetivamente em cromossomas (depósitos), genes (camiões) e alelos (clientes). Conforme o nível do ciclo, são introduzidas as restrições/custos associados aos atributos processados nesse ciclo.

A seguinte listagem procura esquematizar os principais passos deste processo.

1 Partir o genoma em cromossomas (matriz depósitos);
2 Iniciar o custo a zero;
3 **1º Ciclo**: Iterar cada um dos cromossomas (depósitos):
  3.1 Se depósito for aberto:
    (i) Incrementa custo com custo de abertura de depósito;
    (ii) Partir cromossoma em genes (matriz camiões deposito_**i**)
    (iii) **2º Ciclo**: Iterar cada um dos genes (camiões):
      A Calcular o percurso efetivamente feito pelo camião;
      B Se houver percurso, aumenta custo com custo de abertura de rota / uso de camião;
      C Partir gene em alelos (matriz clientes do percurso)
      D **3º Ciclo**: Iterar cada alelo (cliente do percurso):
        D.1 Se a carga atual do veiculo for suficiente, satisfaz procura do cliente;
        D.2 Caso contrário, penaliza o custo com valor ε, porque visita um cliente sem stock para o satisfazer, fazendo um percurso desnecessário.
    (iv) Incrementa o custo com o custo da distância total percorrida ao visitar os clientes;
4 Se a procura de todos os clientes não foi satisfeita, penaliza a solução final, pq apesar de válida nao satisfaz a procura total;
4.1 Caso contrário, devolve o custo calculado até então.

5. Apresentação Resultados

Neste capitulo serão apresentados os resultados obtidos com a execução do algoritmo construido nos datasets apresentados na Secção 3.2

No sentido de uniformizar a descoberta de soluções, foi construido um scrip que itera o algoritmo genético sobre cada ficheiro encontrado numa diretoria. Após a convergência do algoritmo genético, o melhor genoma da população (Secção 4.1) é avaliado por uma função criada para recolher as seguintes informações:

- Nome do ficheiro com o problema analisado;
- Tempo de execução do alg. genético;
- Número de clientes;
- Número de depósitos (possíveis de abrir);
- Número máximo de veículos disponíveis por depósito;
- Com base no melhor genoma da população:
  - Número de depósitos utilizados;
  - Número de veículos utilizado por cada depósito (na prática, se o índice **i** for zero indica que o deposito_**i** não é aberto);
  - Custo da solução, com base nas restrições do problema e percurso dos veículos;
  - Um gráfico onde são assinalados os clientes, depósitos e rotas dos veículos, extrapoladas a partir das informações do genoma.

A estas informações são registados num ficheiro *csv* que posteriormente serve como base para a construção das tabelas de resultados apresentadas neste capitulo.

De uma forma geral, a obtenção de resultados com qualidade neste tipo de algoritmos encontra-se intrinsecamente ligada com a regulação dos parâmetros do algoritmo, para casa caso em estudo. Como no âmbito deste projeto todos os datasets relacionam-se com o mesmo domínio e problema, foi tomada a decisão de realizar todos os teste com a mesma parametrização do algoritmo genético.

Contudo, foi definida uma parametrização suficientemente ampla para possibilitar a evolução dos cromossomas ao longo de várias iterações, explorando assim todos os principais atributos possíveis de regular na função *ga* da linguagem *R*. Como parâmetros principais:

- São realizadas 250 iterações, alterando a geração anterior a cada nova iteração;
- O tamanho da população é de 75 indivíduos;
- A probabilidade mutação é de 30%. Apesar deste valor ser geralmente baixo, no contexto do problema é interessante a introdução de novas características (rotas) no cromossoma, que possam ser relevantes para a construção de uma solução próxima da ótima;
- A probabilidade de cruzamento é de 85%;

- A taxa de elitismo foi definida como 20%, ou seja, 20% das melhores soluções da geração são mantidas na geração seguinte.
  Este fator é útil no sentido em que, caso a fase mutação ou crossover "estraguem"boas soluções, as melhores soluções da geração anterior são mantidas na geração seguinte. Ao garantir que existe sempre um conjunto de melhores soluções na população, permite encaminhar a evolução do algoritmo numa convergência no sentido de soluções próximas da ótima.

5.1 Resultados Instância Barreto

Dados Ficheiro			Optimização Parametros			Detalhes Melhor solução				
File Name	Nr Clientes	Nr Depots	Nr Camioes	Nr Alelos	Size genoma	Tempo (min)	Custo (U.M.)	Nr Used Depots	Nr Used camioes per depot	Nr not used Depots
coordChrist100.json	100	10	8	13	1040	1.07	6297.8	10	6 6 5 3 2 4 4 7 4 5	0
coordChrist50.json	50	5	5	10	250	20.94	2485.5	5	5 3 2 3 4	0
coordChrist75.json	75	10	9	8	720	1.15	5351.32	10	4 4 4 5 4 3 4 3 7 2	0
coordDas150.json	150	10	10	15	1500	1.76	329851.07	10	8 8 7 6 7 6 5 6 8 6	0
coordDas88.json	88	8	5	17	680	49.79	3025.03	8	4 5 3 4 5 5 3 5	0
coordGaspelle.json	21	5	4	5	100	16.86	1317.26	5	1 3 1 2 2	0
coordGaspelle2.json	22	5	3	9	135	15.63	1575.14	4	0 1 2 2 3	1
coordGaspelle3.json	29	5	3	10	150	22.75	1758.15	5	3 2 1 1 3	0
coordGaspelle4.json	32	5	4	8	160	22.9	1795.44	3	0 3 3 4 0	2
coordGaspelle5.json	32	5	3	11	165	21.78	1907.32	5	3 1 2 2 2	0
coordGaspelle6.json	36	5	4	10	200	24.8	1556.92	5	2 3 4 2 2	0
coordMin134.json	134	8	10	14	1120	1.71	65194.03	8	9 8 8 7 8 7 3 5	0
coordMin27.json	27	5	4	8	160	22.46	10300.34	5	3 2 1 2 2	0
coordOr117.json	117	14	5	27	1890	2.88	85627.15	14	4 5 3 2 4 3 4 2 4 4 3 5 2 3	0

Execução com número de camiões e alelos optimizado

Figura 5.1: Resultados recolhidos após aplicação do algoritmo genético às instâncias do conjunto de dados Barreto (Número de camiões e alelos otimizado).

File Name	Nr Clientes	Nr Depots	Nr Camioes	Tempo (min)	Custo (U.M.)	Nr Used Depots	Nr Used camioes per depot	Nr not used Depots
coordChrist100.json	100	10	7	5.43	5757.11	10	4 3 3 2 5 3 6 4 5 4	0
coordChrist50.json	50	5	5	1.23	2304.48	5	4 4 2 4 3	0
coordChrist75.json	75	10	9	6.11	4646.79	10	1 3 4 4 7 2 4 5 2 4	0
coordDas150.json	150	10	10	13.24	331019.9	10	5 9 5 8 8 9 5 6 6 7	0
coordDas88.json	88	8	5	4.03	2963.18	8	4 4 5 3 5 3 4 4	0
coordGaspelle.json	21	5	4	30.81	1187.86	3	1 4 0 3 0	2
coordGaspelle2.json	22	5	2	19.67	1363.79	3	2 0 0 2 1	2
coordGaspelle3.json	29	5	3	34.46	1416.35	3	0 0 3 1 3	2
coordGaspelle4.json	32	5	4	50.27	1796.49	4	1 4 2 2 0	1
coordGaspelle5.json	32	5	3	38.73	1452.64	3	1 0 3 3 0	2
coordGaspelle6.json	36	5	4	53.09	1464.43	5	1 4 3 2 2	0
coordMin134.json	134	8	9	9.55	61943.1	8	7 6 8 6 6 4 6 7	0
coordMin27.json	27	5	3	38.46	7712.55	4	0 1 1 3 2	1
coordOr117.json	117	14	4	5.91	79281.98	14	3 3 4 3 1 3 3 3 3 2 3 2 4 4	0

Execução com número de camiões optimizado | nr alelos = nr_cliente (não optimizado)

Figura 5.2: Resultados recolhidos após aplicação do algoritmo genético às instâncias do conjunto de dados Barreto. (Número de camiões otimizado e número de alelos igual ao número de clientes - não otimizado)

De notar que nos testes apresentado na figura 5.2, por não se otimizar as dimensões do número de alelos o tamanho do cromossoma é maior. Como consequência, é possível observar o maior tempo de execução face às resultados da tabela da Figura 5.1.

Contudo, a nível de dos resultados obtidos para o custo, é possível observar que os valores se encontram próximos para a mesma instância. Como tal, a redução das dimensões do genoma não prejudicam o desempenho de otimização do algoritmo, embora tragam fortes vantagens a nível do tempo de processamento necessário para computar uma solução próxima da ótima.

5.2 Resultados Instância Prodhon (Prins et al)

File Name	Nr Clientes	Nr Depots	Nr Camioes	Tempo (min)	Custo (U.M.)	Nr Used Depots	Nr Used camioes per depot										Nr not used Depots
coord100-10-1.json	100	10	23	2.45	546965.21	9	7	10	7	8	6	0	10	4	8	5	1
coord100-10-1b.json	100	10	11	1.66	531510.66	9	5	7	4	5	6	0	6	6	6	6	1
coord100-10-2.json	100	10	22	2.28	528201.46	9	9	7	10	4	6	10	7	7	7	0	1
coord100-10-2b.json	100	10	11	1.64	510925.88	9	6	0	9	7	6	5	2	3	4	7	1
coord100-10-3.json	100	10	22	2.27	515578.33	9	6	7	9	5	0	11	6	6	6	10	1
coord100-10-3b.json	100	10	11	1.46	446452.11	8	7	9	3	7	0	6	8	0	4	7	2
coord100-5-1.json	100	5	23	1.06	293563.63	5	9	8	10	12	16						0
coord100-5-1b.json	100	5	11	41.78	272955.89	5	6	9	9	9	3						0
coord100-5-2.json	100	5	23	1.02	315618.33	5	7	9	11	12	14						0
coord100-5-2b.json	100	5	11	41.73	297533.99	5	6	7	7	8	8						0
coord100-5-3.json	100	5	23	1.03	296559.28	5	10	15	11	11	8						0
coord100-5-3b.json	100	5	11	41.47	273481.79	5	8	10	4	6	6						0
coord20-5-1.json	20	5	5	14.52	22445.7	2	0	0	4	4	0						3
coord20-5-1b.json	20	5	3	13.82	21135.59	2	0	0	3	2	0						3
coord20-5-2.json	20	5	5	14.08	21359.04	2	0	0	0	5	1						3
coord20-5-2b.json	20	5	3	12.33	20468.26	2	0	3	2	0	0						3
coord200-10-1.json	200	10	45	3.94	1136578.56	10	12	10	16	18	16	15	15	10	10	15	0
coord200-10-1b.json	200	10	21	2.53	1101496.89	10	11	10	9	15	11	15	7	13	5	9	0
coord200-10-2.json	200	10	45	3.94	1266951.26	10	13	12	15	13	13	18	14	12	15	15	0
coord200-10-2b.json	200	10	21	2.5	1231755.26	10	11	11	10	9	12	11	10	10	11	12	0
coord200-10-3.json	200	10	44	3.79	1150346.81	10	9	9	16	18	22	15	12	9	14	11	0
coord200-10-3b.json	200	10	21	2.7	1120681.99	10	11	15	9	14	10	12	11	8	13	4	0
coord50-5-1.json	50	5	11	32.46	57644.85	4	4	6	7	3	0						1
coord50-5-1b.json	50	5	6	25.18	56551.33	4	4	5	6	4	0						1
coord50-5-2.json	50	5	12	34.3	81007.91	5	4	3	4	6	5						0
coord50-5-2b.json	50	5	6	24.96	74373.73	5	3	4	3	2	4						0
coord50-5-2bBIS.json	50	5	6	25.35	43973.47	4	0	3	4	4	6						1
coord50-5-2BIS.json	50	5	11	33.53	57649.39	5	4	6	4	5	5						0
coord50-5-3.json	50	5	11	33.04	72940.84	5	6	3	3	7	3						0
coord50-5-3b.json	50	5	6	25.13	53610.75	4	3	0	5	5	4						1

Execução com número de camiões e alelos optimizado

Figura 5.3: Resultados recolhidos após aplicação do algoritmo genético às instâncias do conjunto de dados Prodhon (200 iterações e população com 75 indivíduos)

Dados Ficheiro			Optimização parâmetros			Detalhes Melhor solução															
File Name	Nr Clientes	Nr Depots	Nr Camioes	Nr Alelos	Size genoma	Tempo (min)	Custo (U.M.)	Nr Used Depots	Nr Used camioes per depot												Nr not Used Depots
coord100-10-1.json	100	10	23	4	920	1.99	552483.9	9	5	11	11	7	9	9	9	5	5	0			1
coord100-10-1b.json	100	10	11	8	880	1.39	532892.72	9	6	9	5	3	4	6	0	6	6	5			1
coord100-10-2.json	100	10	22	4	880	2.08	578278.4	10	5	5	7	7	6	9	6	6	5	5			0
coord100-10-2b.json	100	10	11	9	990	1.79	512657.3	9	6	6	4	10	5	6	5	5	6	0			1
coord100-10-3.json	100	10	22	4	880	2.71	575038.18	10	7	7	10	8	6	7	4	2	7	8			0
coord100-10-3b.json	100	10	11	9	990	2.16	555160.29	10	6	6	5	5	3	2	7	4	6	4			0
coord100-5-1.json	100	5	23	4	460	1.21	295608.4	5	9	11	13	13	11								0
coord100-5-1b.json	100	5	11	9	495	51.58	271626.73	5	6	8	6	9	6								0
coord100-5-2.json	100	5	23	4	460	1.2	318556.15	5	11	10	13	10	12								0
coord100-5-2b.json	100	5	11	9	495	48.89	299356.52	5	8	7	8	8	7								0
coord100-5-3.json	100	5	23	4	460	1.22	298070.83	5	10	11	10	13	13								0
coord100-5-3b.json	100	5	11	9	495	51.08	274548.08	5	6	7	8	7	7								0
coord20-5-1.json	20	5	5	4	100	17.59	29976.63	3	0	0	4	2	2								2
coord20-5-1b.json	20	5	3	9	135	15.64	21295.68	2	0	0	2	3	0								3
coord20-5-2.json	20	5	5	4	100	18.41	25408.61	2	0	0	0	5	5								3
coord20-5-2b.json	20	5	3	9	135	16.57	19429.52	2	0	3	0	2	0								3
coord200-10-1.json	200	10	45	4	1800	4.57	1136750.5	10	17	7	17	15	8	14	14	14	18	14			0
coord200-10-1b.json	200	10	21	9	1890	3.13	1103536	10	14	11	8	9	12	12	12	8	11	10			0
coord200-10-2.json	200	10	45	4	1800	4.6	1272474.52	10	13	15	15	20	13	10	14	16	18	11			0
coord200-10-2b.json	200	10	21	9	1890	3.25	1231631.28	10	10	11	14	10	10	6	13	15	7	11			0
coord200-10-3.json	200	10	44	4	1760	4.4	1154226.44	10	14	12	16	13	9	15	11	18	19	12			0
coord200-10-3b.json	200	10	21	9	1890	3.19	1123502.4	10	13	7	14	12	11	12	13	9	7	12			0
coord50-5-1.json	50	5	11	4	220	36.19	74429.67	5	5	5	3	5	5								0
coord50-5-1b.json	50	5	6	9	270	28.04	55462.43	4	4	6	3	5	0								1
coord50-5-2.json	50	5	12	4	240	38.69	84058.92	5	6	3	7	6	3								0
coord50-5-2b.json	50	5	6	9	270	27.4	61756.14	4	6	5	3	0	4								1
coord50-5-2bBIS.json	50	5	6	9	270	28.69	51459.6	5	5	4	2	4	2								0
coord50-5-2BIS.json	50	5	11	4	220	37.41	59043.42	5	5	6	4	7	3								0
coord50-5-3.json	50	5	11	4	220	37.42	76022.48	5	5	5	4	6	5								0
coord50-5-3b.json	50	5	6	9	270	28.01	53075.28	4	0	4	5	3	4								1

Execução com número de camiões e alelos optimizado

Figura 5.4: Resultados recolhidos após aplicação do algoritmo genético às instâncias do conjunto de dados Prodhon (200 iterações e população com 60 indivíduos)

5.3 Resultados Instância Tuzun and Burke

Dados Ficheiro			Optimização Parametros			Detalhes Melhor solução																						
File Name	Nr Clientes	Nr Depots	Nr Camioes	Nr Alelos	Size genoma	Tempo (min)	Custo (U.M.)	Nr Used Depots	Nr Used camioes per depot																			
coordP111112.json	100	10	11	9	990	1.56	12584.69	10	7	4	6	4	4	6	8	4	8	4										
coordP111122.json	100	20	10	10	2000	2.59	13404.66	20	4	3	1	2	3	3	2	2	3	2	6	3	5	2	2	1	4	2	3	3
coordP111212.json	100	10	10	10	1000	1.68	10940.26	10	5	3	6	6	7	6	4	4	4	6										
coordP111222.json	100	20	11	9	1980	3.64	15138.32	19	3	2	5	4	0	2	2	5	2	2	4	2	3	2	3	5	3	4	5	3
coordP112112.json	100	10	11	9	990	2.05	15582	10	6	6	6	5	5	8	3	6	6	2										
coordP112122.json	100	20	10	10	2000	2.78	12861.7	20	5	3	2	2	7	2	2	2	2	5	4	2	2	3	2	2	4	1	4	3
coordP112212.json	100	10	11	9	990	1.5	8357.92	10	7	7	6	4	7	5	7	1	6	4										
coordP112222.json	100	20	10	10	2000	2.77	12680.2	20	3	2	5	1	5	2	2	2	4	3	4	3	2	5	2	1	3	4	5	1
coordP113112.json	100	10	10	10	1000	1.57	11750.71	10	5	4	6	2	6	5	5	8	6	6										
coordP113122.json	100	20	11	9	1980	2.82	15939.09	18	0	2	6	4	7	2	2	2	1	4	3	4	2	2	4	4	6	2	3	0
coordP113212.json	100	10	10	10	1000	1.52	11135.61	10	6	8	3	7	3	6	5	6	3	8										
coordP113222.json	100	20	10	10	2000	2.79	13410.36	17	5	4	3	3	3	1	1	5	4	5	1	0	4	4	2	0	5	2	0	2
coordP121112.json	200	10	20	10	2000	3.15	24804	10	10	10	8	8	14	15	9	12	12	11										
coordP121122.json	200	20	21	9	3780	5.92	33407.58	20	5	7	6	9	3	4	7	7	7	5	4	8	6	5	5	6	7	9	10	7
coordP121212.json	200	10	20	10	2000	3.02	23296.76	10	9	11	12	12	9	10	10	10	13	11										
coordP121222.json	200	20	20	10	4000	5.64	27163.03	20	8	11	9	6	6	8	6	7	4	6	7	5	5	5	9	7	8	3	5	4
coordP122112.json	200	10	21	9	1890	2.62	19870.76	10	5	14	13	14	12	14	13	8	12	10										
coordP122122.json	200	20	20	10	4000	4.7	24751.16	20	7	8	9	6	4	7	3	8	10	6	8	6	6	5	6	8	6	6	6	6
coordP122212.json	200	10	20	10	2000	2.47	19390.85	10	12	14	10	9	11	11	11	8	7	14										
coordP122222.json	200	20	21	9	3780	4.69	27923.73	19	7	7	7	6	8	5	8	12	6	11	9	5	4	7	6	6	6	8	6	0
coordP123112.json	200	10	21	9	1890	2.4	27355.75	10	12	15	11	12	8	9	9	14	8	13										
coordP123122.json	200	20	20	10	4000	4.79	34385.5	19	8	11	4	0	10	6	5	8	8	5	11	5	10	5	9	10	5	4	5	8
coordP123212.json	200	10	21	9	1890	2.33	19718.41	10	13	10	11	12	12	12	10	13	8	12										
coordP123222.json	200	20	21	9	3780	4.57	30360.95	20	8	7	5	8	10	5	6	5	6	6	6	3	11	5	8	6	12	7	6	6
coordP131112.json	150	10	15	10	1500	1.64	19637.19	10	7	9	6	9	8	5	9	9	11	7										
coordP131122.json	150	20	15	10	3000	2.81	21901.23	20	6	4	4	5	4	4	4	5	7	3	5	4	7	6	5	6	2	4	5	6
coordP131212.json	150	10	16	9	1440	1.55	16470.82	10	10	10	9	8	8	7	11	6	4	8										
coordP131222.json	150	20	15	10	3000	2.9	20760.95	19	4	9	6	3	0	2	7	5	5	7	6	3	5	6	5	4	3	7	3	3
coordP132112.json	150	10	15	10	1500	1.55	18766.63	10	7	7	8	9	8	8	9	10	6	9										
coordP132122.json	150	20	15	10	3000	3.03	21866.63	20	4	5	6	4	6	4	1	6	5	3	3	3	6	6	7	6	7	4	5	2
coordP132212.json	150	10	15	10	1500	1.67	15013.73	10	6	10	6	5	9	10	6	13	10	10										
coordP132222.json	150	20	16	9	2880	2.86	24384.24	20	6	5	6	5	7	4	5	3	3	2	2	5	5	3	5	5	6	5	6	5
coordP133112.json	150	10	15	10	1500	1.52	19706.93	10	10	6	9	8	9	9	10	10	3	6										
coordP133122.json	150	20	15	10	3000	2.77	20503.78	20	5	4	5	5	4	5	3	2	4	8	6	2	6	5	6	7	3	7	9	4
coordP133212.json	150	10	16	9	1440	1.54	11665.48	10	7	8	10	9	9	10	6	7	7	9										
coordP133222.json	150	20	15	10	3000	3.03	21101.37	19	3	6	3	3	4	5	5	4	7	7	6	7	2	4	7	9	0	5	6	4

Execução com número de camiões e alelos optimizado

Figura 5.5: Resultados recolhidos após aplicação do algoritmo genético às instâncias do conjunto de dados Tuzun. (200 iterações e população com 75 indivíduos)

File Name	Nr Clientes	Nr Depots	Nr Camioes	Tempo (min)	Custo (U.M.)	Nr Used Depots	Nr Used camioes per depot	Nr not used Depots
coordP111112.json	100	10	11	1.73	11778.4	10	6 6 5 6 6 6 6 4 4 1	0
coordP111122.json	100	20	10	2.81	13609.17	18	3 3 0 4 3 1 3 1 3 0 2 3 4 5 3 4 3 3 3 4	2
coordP111212.json	100	10	10	1.51	11289.14	10	4 3 7 7 6 2 5 6 6 5	0
coordP111222.json	100	20	11	3.02	13894.26	17	3 5 3 3 3 3 7 3 3 2 0 3 3 3 2 0 3 0 5 3	3
coordP112112.json	100	10	11	1.55	14898.19	10	5 4 5 7 4 5 6 5 7 2	0
coordP112122.json	100	20	10	2.38	11615.99	16	3 0 1 5 4 3 4 2 0 2 5 4 3 4 1 3 3 6 0 0	4
coordP112212.json	100	10	11	1.26	7712.38	10	7 4 7 5 4 5 8 5 7 4	0
coordP112222.json	100	20	10	2.33	13225.61	19	1 4 3 5 1 2 1 1 4 3 4 6 5 0 4 4 2 3 5 4	1
coordP113112.json	100	10	10	1.24	11224.07	10	6 7 2 4 4 7 4 7 3 6	0
coordP113122.json	100	20	11	2.43	15788.99	19	3 5 3 6 3 5 5 2 3 1 0 4 1 1 4 4 3 2 1 3	1
coordP113212.json	100	10	10	1.23	10629.11	10	4 4 5 3 5 4 4 9 6 8	0
coordP113222.json	100	20	10	2.35	13543.38	18	5 2 4 2 0 3 0 3 3 3 2 6 3 3 2 3 3 4 5 3	2
coordP121112.json	200	10	20	2.51	26066.28	10	12 11 13 11 10 10 14 11 12 10	0
coordP121122.json	200	20	21	4.81	33615.04	20	6 5 4 7 4 9 4 8 5 6 6 8 6 6 8 4 9 8 11 6	0
coordP121212.json	200	10	20	2.47	23308.48	10	10 14 10 12 7 13 13 8 11 8	0
coordP121222.json	200	20	20	4.93	26524.69	20	8 5 9 6 8 9 8 5 3 5 8 5 9 6 7 4 3 4 8 8	0
coordP122112.json	200	10	21	2.62	20142.74	10	13 9 11 13 9 10 12 14 8 11	0
coordP122122.json	200	20	20	4.43	24771.35	20	5 9 5 7 6 8 10 6 7 5 7 5 7 7 8 5 10 3 6 8	0
coordP122212.json	200	10	20	2.4	19271.58	10	10 10 12 11 13 12 4 5 12 16	0
coordP122222.json	200	20	21	4.45	29277.04	20	5 9 7 5 8 6 8 6 5 8 4 6 12 3 6 12 7 7 7 5	0
coordP123112.json	200	10	21	2.43	27585.31	10	13 12 12 12 12 10 10 11 10 8	0
coordP123122.json	200	20	20	4.46	31140.82	20	5 4 9 8 8 7 6 10 7 5 5 7 3 4 7 5 7 7 3 5	0
coordP123212.json	200	10	21	2.47	21048.61	10	9 14 11 15 13 10 16 11 8 10	0
coordP123222.json	200	20	21	4.62	31504.72	20	6 10 6 7 4 8 7 9 6 8 5 8 6 9 7 7 5 8 8 5	0
coordP131112.json	150	10	15	1.74	19172.06	10	9 7 7 8 9 7 6 7 10 9	0
coordP131122.json	150	20	15	3.12	22283.3	20	5 5 3 5 4 4 5 6 7 4 7 8 3 5 8 3 5 5 2 3	0
coordP131212.json	150	10	16	1.72	16148.18	10	9 11 6 9 9 5 6 12 9 6	0
coordP131222.json	150	20	15	3.3	20421.28	20	3 3 6 6 4 5 5 5 6 2 2 7 4 7 5 3 4 4 5 6	0
coordP132112.json	150	10	15	1.71	17218.88	10	6 5 12 6 8 9 7 10 6 8	0
coordP132122.json	150	20	15	3.35	21766.5	20	2 4 3 8 5 3 4 9 3 6 3 3 2 3 5 5 9 4 2 5	0
coordP132212.json	150	10	15	1.76	14791.38	10	5 11 9 8 9 4 7 9 10 8	0
coordP132222.json	150	20	16	3.08	24493.45	20	6 5 2 5 2 2 2 4 4 5 6 9 4 7 3 7 5 6 6 5	0
coordP133112.json	150	10	15	1.63	18735.13	10	8 6 9 8 5 7 11 10 6 6	0
coordP133122.json	150	20	15	2.9	19382.58	20	5 6 2 2 9 5 2 5 4 4 6 5 5 8 2 8 6 1 4 3	0
coordP133212.json	150	10	16	1.54	12012.11	10	10 4 9 7 10 7 8 8 9 7	0
coordP133222.json	150	20	15	2.9	20537.4	20	4 3 9 6 4 4 4 6 5 4 3 5 5 8 5 4 4 7 2 3	0

Execução com número de camiões e alelos optimizado

Figura 5.6: Resultados recolhidos após aplicação do algoritmo genético às instâncias do conjunto de dados Tuzun. (200 iterações e população com 60 indivíduos)

5.4 Análise de Resultados

Como análise detalhada de resultados será analisada a instância ***coord200-10-1*** do conjunto *Prodhon (Prins et al)*, por apresentar a maior dimensão a nível de número de clientes e depósitos. (Figura 5.3 e 5.4)

Nesta instância do dataset:

- Existem 200 Clientes;
- Existem 10 possíveis localizações para abrir depósitos;
- Cada veiculo apresenta uma capacidade máxima de carga de 70 unidades produto;
- O somatório da procura dos clientes é 3098 unidades produto;
- A média da procura por cliente é de 15.49 unidades produtos.

Recordando as considerações da Secção 4.1, no pior cenário é utilizado um camião por cliente e, cada gene (camião) tem tantos alelos quanto o número de clientes. Ou seja, numa abordagem primitiva para este problema, o genoma seria

uma sequência de valores inteiros com 400000 índices. (nr_depositos * nr_camioes * nr_clientes = 10*200*200 = 400000)

Introduzindo a otimização do número de camiões, segundo a formula 4.1 da secção 4.1, o número de camiões deixa de ser igual ao número de clientes e passa a ser 45 veículos (*ceiling*(3098 / 70) = 45). Desta forma, o tamanho do cromossoma é reduzido para 22.5% da dimensão inicial, passando a ter 90000 índices (10*45*2000).

De forma semelhante, aplicando a redução do número de alelos, com base na procura média dos clientes (4.2 da secção 4.1), o número de alelos deixa de ser igual ao número de clientes para passa a ser igual a 4 clientes. Com isto, o tamanho do genoma passa a ser de apenas 1800 índices (10*45*4), sendo 0.45% do tamanho inicial de 400000.

Estas reduções mostram-se significativas no impacto que causam no tempo de execução do algoritmo. Quanto menor as dimensões do genoma, mais eficientes se torna a computação dos processos de avaliação, mutação e cruzamento de soluções. No caso

5.5 Visualização de Rotas

Nesta secção são apresentados alguns gráficos criados pela função que permite visualizar a disposição das rotas de cada camião ao longo de um mapa, onde se encontram assinalados os depósitos e os clientes.

Para construir o plot:

- Cada percurso com saída num determinado depósito tem um cor especifica;
- Cada percurso de um determinado depósito é marcado com um traço diferente, distinguindo assim os percursos dos diferentes camiões;
- Os ícones que representam os clientes (quadrados azuis), depósitos (triangulos vermelhos) e respetivos nomes são adicionados apenas no final, de forma serem legíveis (não são sobrepostos pelas setas dos percursos)

Neste secção são apenas apresentados 3 exemplos de gráfios, respetivamente para um caso de cada dataset em análise. Devido à sua resolução, os gráficos são apenas ilustrativos da funcionalidade implementada. Para observar com detalhe os gráficos é necessário abrir as imagens criadas (enviadas junto com o presente documento).

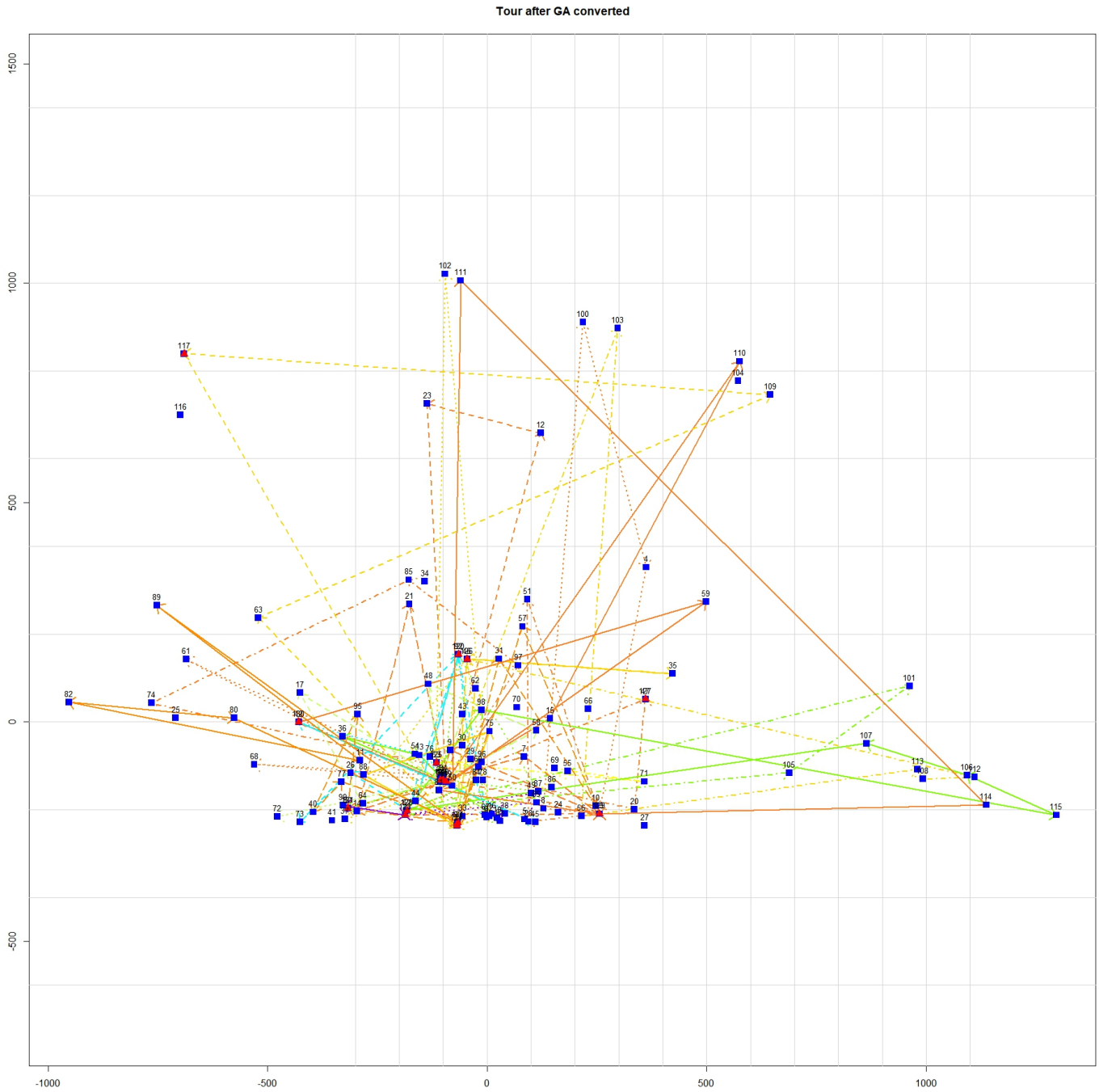

Figura 5.7: Representação visual dos tragetos encontrados na melhor solução do Algoritmo genético, para o ficheiro **coordOr117** do dataset Barreto.

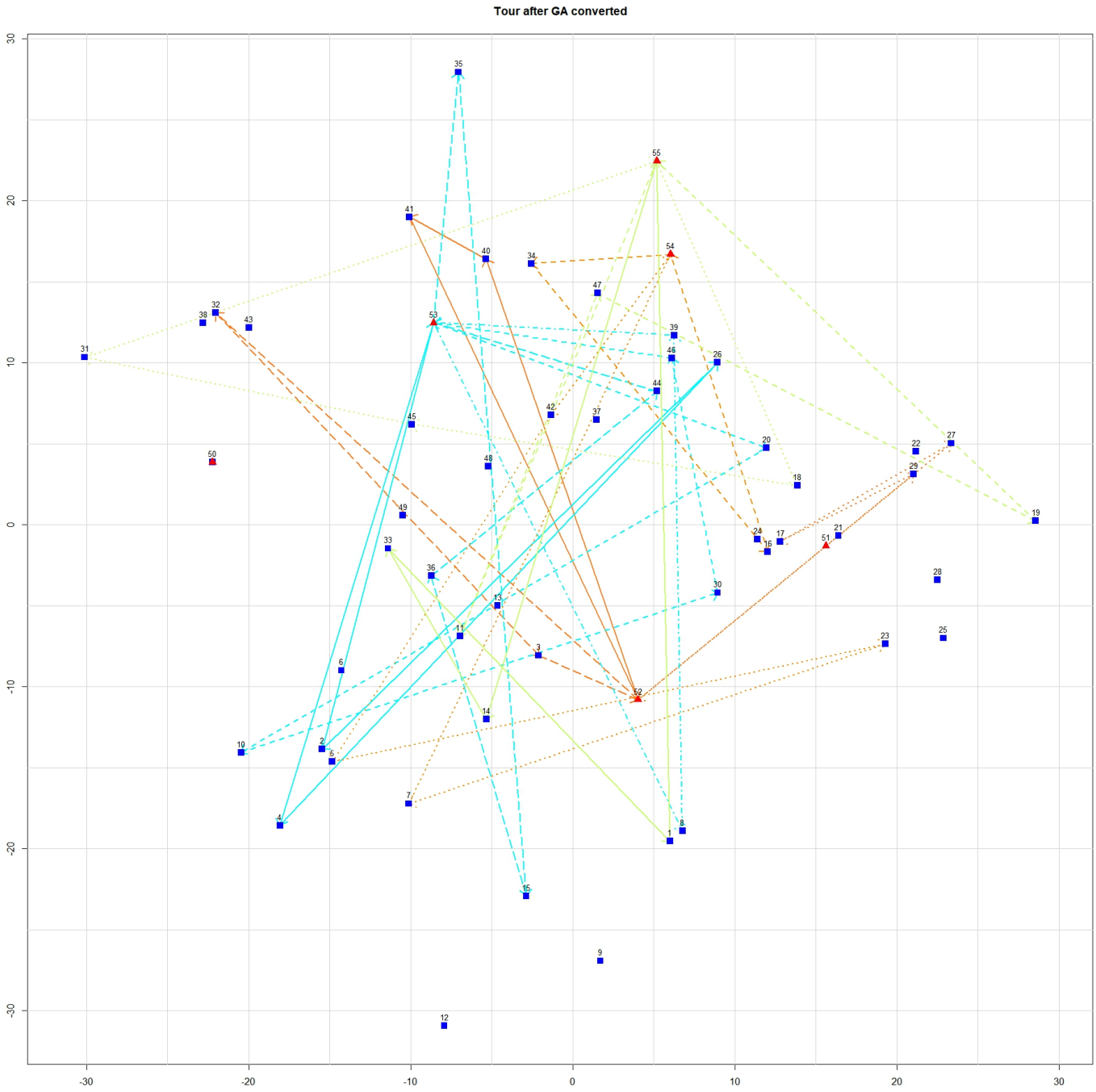

Figura 5.8: Representação visual dos tragetos encontrados na melhor solução do Algoritmo genético, para o ficheiro **coord50-5-3b** do dataset Prodhon.

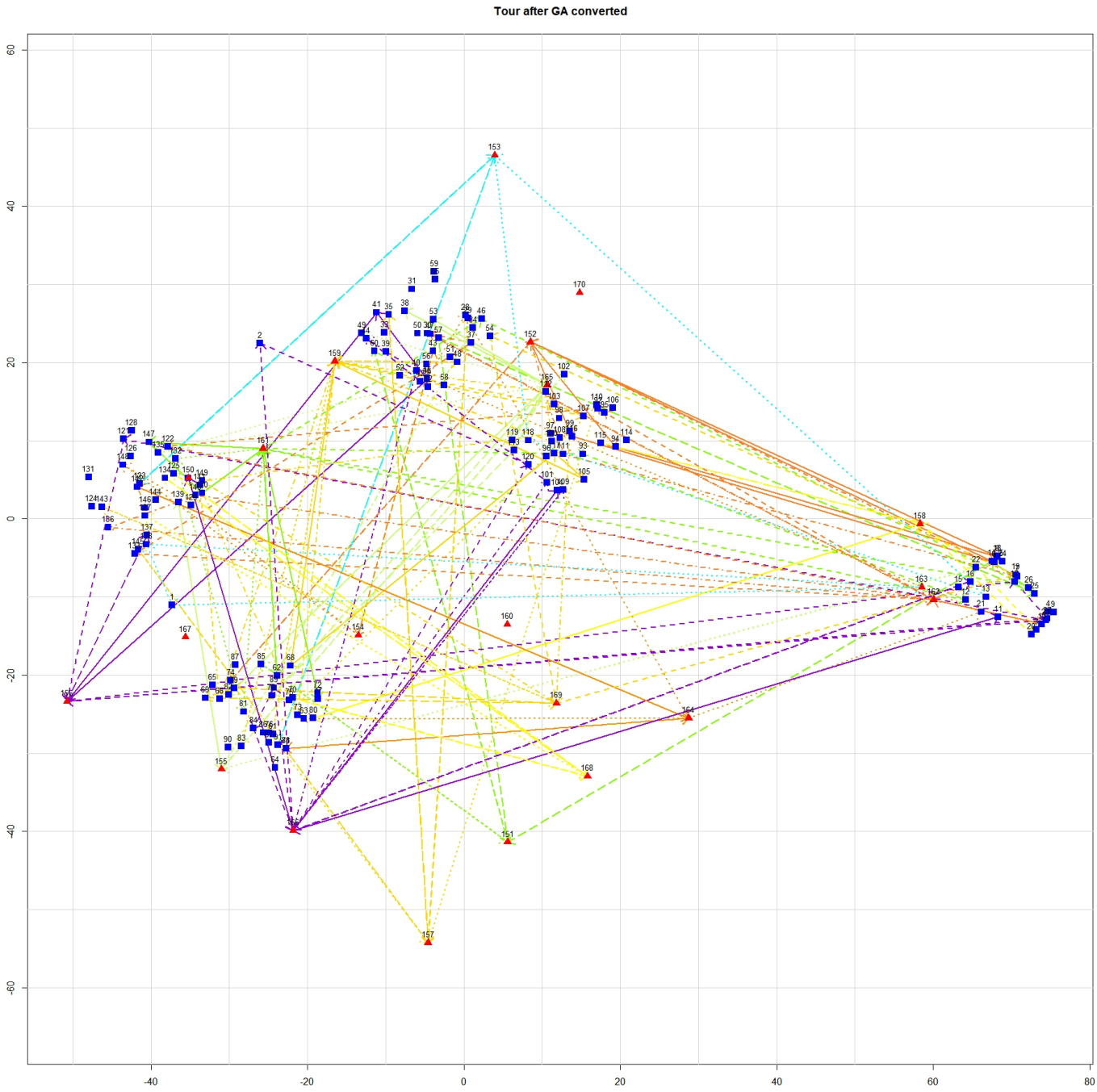

Figura 5.9: Representação visual dos tragetos encontrados na melhor solução do Algoritmo genético, para o ficheiro **coordP133222** do dataset Tuzun.

6. Considerações finais

- Garantindo que novos enunciados se encontrem com a mesma estrutura dos datasets utilizados, qualquer novo problema de *LRP* com novas localizações ou quantidades de procura, podem igualmente ser processados pelo algoritmo genético criado, sem qualquer necessidade de adaptações;

- Apesar deste tipo de problemas estar intrinsecamente ligado com o "tunning"dos parametros de aprendizagem, nas execuções do algoritmo foram utilizados valores constantes de iterações e de dimensão de população.

  Uma vez que nestes processos não se equaciona o problema de *overfitting*, o processamento de uma população maior, ao longo de várias iterações, permite aprimorar as soluções até atingir, eventualmente, resultados próximos da solução ótima.

  Em cenários em que a evolução de gerações fique estagnada a nível da minimização dos custos, o algoritmo apresenta mecanismos para terminar antecipadamente, sem que sejam necessárias executar todas as iterações definidas.

  Desta forma, como os tempos de execução são razoáveis, o número de iterações e da população inicial podem ser elevados (No limite, 500 iterações com 100 indivíduos na população leva a um tempo de execução de 1 hora, para o enunciado Prodhon com 200 cliente e 10 depósitos).

- Sendo algoritmos baseados em probabilidades, refere-se também a flutuação entre os tempos de execução e os custos obtidos nos vários cenários de teste realizados.

  Como estes fatores estão fortemente relacionados com variabilidade da população inicial, o aumento do número de iterações procura assim "dar tempo"ao algoritmo, para eventualmente conseguir extrair boas características de cada geração criada.

- A função responsável por criar um gráfico com os percursos dos trajetos apresenta, em alguns casos, clientes sem retas traçadas.

  Este aspeto está relacionado com uma falha na codificação da função que cria o plot. A ideia inicial seria utilizar setas (arrows) com cores distintas por depósitos e tipo de linha/traço distinto por cada percurso de camião. Contudo, por algum motivo existe conflitos com tipos de tracejados já utilizados, o que faz com que o método *arrows* não desenhe alguns dos percursos.

  Contudo, convém frisar que na função de fitness encontra-se explicitamente codificado uma restrição que verifica se toda a procura dos clientes foi satisfeita com as rotas do genoma. Caso tal não se verifique o custo é aumentando exponencialmente para que a solução seja penalizada em excesso e descartada na evolução do algoritmo.

Por este motivo, apesar do erro nos gráficos (que não são o propósito do projeto) as soluções fornecidas pelo algoritmo garantem a passagem e satisfação de todos os clientes.

7. Conclusão

No sentido de construir um algoritmo genético desenvolvido para ser aplicado em problemas de *Location Routing (LRPs)*, o presente relatório descreve todas as etapas efetuadas ao longo do projeto.

O capitulo inicial, onde se introduzem os principais conceitos associados com algoritmos genéticos, apresenta-se essencial, no sentido em que foca todos os aspetos necessários para compreender muitas das decisões tomadas ao longo do projeto, relacionadas com o modo de funcionamento e aprendizagem destes métodos.

Relativamente ao problema em estudo, destaca-se a versatilidade do algoritmo para qualquer uma das instâncias dos três datasets analisados. Esta característica de modularidade é garantida pela abstração dada ao conteúdo do cromossoma, sendo que apenas na função de *fitness* de soluções, se é tida em consideração a estrutura abstrata de cada cromossoma.

Embora o aumento do número de iterações e indivíduos da população leve a um aumento dos tempos de execução do algoritmo, esta desvantagem foi extremamente reduzida pela otimização aplicada à dimensão do cromossoma. As otimizações conseguidas focam tanto a redução do número de camiões necessários por depósito (com base na procura dos clientes), como a redução do número de cliente pelos quais o percurso do camião precisa de visitar (com base na sua carga máxima).

Relativamente à otimização do número de alelos (clientes pelos quais um camião pode passar no seu percurso), podiam ainda ser realizados testes que se baseassem em funções como o mínimo, máximo ou a mediana da procura dos clientes. Devido ao tempo de desenvolvimento do projeto, foi apenas explorada a média da procura dos clientes, para determinar quantos clientes a carga do camião pode satisfazer.

No geral, destaca-se também os vários detalhes recolhidos após a execução de cada algoritmo genético, sendo ainda possível criar uma visualização gráfica dos trajetos encontrados pela melhor solução do algoritmo genético.

Em suma, o presente projeto apresenta uma implementação bem sucessida de um algoritmo genético capaz de resolver em tempo de execução aceitáveis uma variação simples do problema de alocação e distribuição de produtos.

Bibliografia

[1] Marco Aurélio and Cavalcanti Pacheco. Algoritmos genÉticos: PrincÍpios e aplicaÇÕes. 05 2018.

[2] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1989.

[3] António Parra Truyol Jorge Arranz de la Peña. Algoritmos genéticos. *Departamento de Engenharia Telemática, Universidade Carlos III de Madrid*, 1993. [Online; Accessed 5 Maio 2018].

[4] Joilma Souza Santos. *Mineração de Dados Utilizando Algoritmos Genéticos*.