

Optimização de Circuitos de Recolha de Lixos Domésticos em Zonas Urbanas

Manuel Joaquim da Silva Oliveira

Dissertação para obtenção do grau de Doutor em Ciências de Engenharia pela Faculdade de Engenharia da Universidade do Porto.



Universidade do Porto
Faculdade de Engenharia
FEUP

Esta tese foi orientada pelo
Professor Doutor Fernando Gomes Martins
Professor Auxiliar do Departamento de Química da Faculdade de
Engenharia da Universidade do Porto.

Março 2008

Agradecimentos

Nietzsche afirmou : “Fui eu que o fiz”, diz a minha memória. “Não posso ter feito isso”, diz o meu orgulho e mantém-se inflexível. Por fim é a memória que cede.

De facto a realização deste trabalho implicou a participação de várias pessoas, sem as quais a sua concretização estaria comprometida.

Gostaria de expressar a minha sincera gratidão e reconhecimento a todos que, de diferentes formas, contribuíram para a realização deste trabalho e especialmente, para aqueles que mais de perto o acompanharam.

Ao Professor Doutor Fernando Gomes Martins, pela coerência, clareza e total disponibilidade demonstradas na orientação deste trabalho.

Ao Professor Doutor Luís Filipe Malheiros, pela paciência, empenho e encorajamento que demonstrou.

Ao Departamento pela cedência de todas as condições adequadas ao desenvolvimento deste trabalho.

À Lipor, na pessoa do Dr. Fernando Leite, pela disponibilidade, confiança e colaboração, que contribuíram para a realização deste projecto.

Ao Engenheiro Paulo Sérgio Oliveira Martins pela colaboração prestada ao longo deste trabalho.

A todos os que me são mais próximos, pelo carinho, ajuda, preocupação e incentivo. Obrigado pela compreensão dos momentos de ausência.

A todos a minha gratidão.

Resumo

Este trabalho tem como objectivos principais o desenvolvimento de aplicações para determinação de percursos optimizados de recolha de resíduos sólidos em zonas urbanas (RSU) e a utilização de ferramentas de informação geográfica na definição do problema de recolha de RSU.

Na primeira parte deste estudo descreve-se o problema da recolha de RSU fazendo-se uma breve descrição da forma como ele é tratado nos dias de hoje. Apresentam-se as metodologias frequentemente usadas na gestão de rotas e de recolha dos resíduos, assim como os requisitos de pessoal e de equipamentos envolvidos. Referem-se de uma forma sucinta os sistemas de informação geográfica, assim como a sua evolução e os contributos para a formulação do problema de gestão de recolha. São mencionadas algumas destas ferramentas mais conhecidas e que podem ser usadas na automação de tarefas relacionadas com o problema em estudo.

A segunda parte concentra-se na descrição dos principais algoritmos de optimização destinados à resolução do problema em estudo. Apresentam-se várias formas de abordar o problema, assim como a formulação do problema recorrendo à abordagem por único caixeiro viajante e múltiplo caixeiro viajante.

De seguida faz-se uma formulação detalhada do problema da recolha de RSU que consiste na especificação de um conjunto de pontos de recolha, situados em diferentes locais, que devem ser visitados por veículos de recolha, partindo de um ponto comum (parque) e regressando a um ponto final (depósito). O objectivo é determinar os circuitos de recolha percorridos por cada veículo, minimizando a distância percorrida por cada um, utilizando o menor número de veículos necessários à recolha e optimizando os custos relacionados com o pessoal e veículos.

A aplicação desenvolvida para a resolução do problema da recolha de RSU é validada através de resolução de vários problemas teóricos. Apresenta-se uma forma recursiva de utilização do algoritmo de optimização, utilizando a noção de super-contentores e super-veículos, que permite transformar o problema inicial complexo em sub-problemas de dimensão mais reduzida.

Na última parte deste estudo adaptaram-se as aplicações desenvolvidas a um problema real. Foi abordado um sistema de recolha de RSU da Lipor, designado por serviço Ecofone, aplicado ao percurso realizado por uma equipa durante um dia.

São ainda apresentadas as conclusões retiradas dos estudos efectuados. Assim como linhas de orientação que podem ser seguidas para desenvolvimento posterior das matérias estudadas ao longo deste trabalho.

Palavras chave: percursos optimizados de recolha de RSU, sistemas de informação geográfica, algoritmos de optimização, super-contentores, super-veículos

Abstract

The main objectives of this work are the development of computer applications for determination of optimised circuits of waste collection in urban zones and the use of the potentialities of geographic information tools to define this problem.

In the first part of this study the problem of waste collection is described through a briefly description of how the problem is analysed nowadays. The often used methodologies to manage routes and to collect waste are presented, as well as the requirements of staff and equipment involved. The geographic information systems are mentioned in a summarized way, as well as its evolution and its contributions for problem formulation. Some of these more known tools are mentioned that can be used in tasks automation related with the study problem.

The second part is concentrated in the description of the main algorithms to optimise the problem in study. Some forms are presented to approach the problem, as well as the formulation of the problem considering one travelling salesman and multiple travelling salesman.

A detailed formulation of the problem of waste collection is followed. This approach consists in the specification of a set of collected points, situated in different places that must be visited by trucks, leaving of a common point (park) and returning to an end point (deposit). The objective is to determine the circuits covered by each truck, minimising the distance covered by each one, using the lesser number of necessary trucks for waste collection and improving the costs related with the staff and vehicles.

The application developed for the resolution of the problem of waste collection is validated evaluating some theoretical problems. A recursive form of the algorithm is presented, using the notion of super-containers and super-trucks that allows transforming the initial complex problem into sub-problems of reduced dimension.

In the last part of this study, a real problem was evaluated for a Lipor system of waste collection, assigned by Ecofone service. This problem analysed the circuit made by a team during one day.

Keywords: Optimised circuits of waste collection, geographic information systems, optimisation algorithms, super-containers, super-trucks.

Résumé

Ce travail a comme objectifs principaux le développement des applications pour la détermination des parcours optimisés de collecte de déchets dans les zones urbaines et l'utilisation d'outils d'information géographique dans la définition du problème de la collecte de déchets.

Dans la première partie de cet étude on décrit le problème de la collecte des déchets en faisant une brève description de la façon laquelle il est traité dans nous jours. On présente les méthodologies fréquemment utilisées dans la gestion des routes et de la collecte des résidus, bien comme les nécessités de personnel et des outils utilisés. On parle d'une façon synthétique des systèmes d'information géographique, bien comme son évolution et les contributions pour la formulation du problème de la gestion de collecte. On mentionne quelques outils plus connus et qui peuvent être utilisés dans l'automatisation de tâches relationnées avec les problèmes en étude.

La deuxième partie se concentre dans la description des principaux algorithmes d'optimisation destinés à la résolution du problème en étude. On présente des formes variées d'aborder le problème, bien comme la formulation du problème en utilisant l'abordage par un seul voyageant et multiple voyageant.

Après on fait une formulation détaillée du problème de la collecte des déchets qui consiste dans la spécification d'un group de points de collecte, situés en des différents lieux, qui doivent être visités par des voitures de collecte, qui partent d'un point commun (parking) et qui reviennent pour un point final (dépôt). L'objectif est déterminer les circuits de collecte parcourus pour chaque voiture, en minimisant la distance parcourue par chacune, en utilisant le plus petit nombre de voitures nécessaires à la collecte et optimisant les coûts relationnés avec le personnel et les voitures.

L'application développée pour la résolution du problème des collectes des déchets est validée par la résolution de plusieurs problèmes théoriques. On présente une façon récursive de l'utilisation de l'algorithme d'optimisation, en utilisant la notion de super conteneurs et de super camions, ce qui permet transformer le problème initial complexe dans des sub-problèmes d'une dimension plus réduite.

Dans la dernière partie de cette étude on a adapté les applications développées à un problème réel. On a abordé un système de collecte de déchets de Lipor, désigné par Service Ecofone.

Mots Clé : parcours optimisés de collecte de déchets, systèmes d'information géographique, algorithmes d'optimisation, super conteneurs, super camions.

Índice

Lista de Figuras	i
Lista de Tabelas	iv
Capítulo 1 - Introdução	1
1.1 Motivação	2
1.2 Objectivo da tese	3
1.3 Estrutura da tese	4
Capítulo 2 - Noções Gerais do Problema de Recolha de Resíduos Sólidos Urbanos / Sistemas de Informação Geográfica	5
2.1 Recolha e transporte de resíduos sólidos urbanos (RSU)	6
2.1.1 Definição	6
2.1.2 Regularidade	6
2.1.3 Frequência de recolha	7
2.1.4 Horários de recolha	8
2.1.5 Redimensionamento de rotas	9
2.1.5.1 Número de elementos que compõem cada equipa	9
2.1.5.2 Equidade de rotas	9
2.1.5.3 Ponto de referência para início da recolha	10
2.1.5.4 Avaliação dos pontos de recolha de RSU	10
2.1.5.5 Pesagem dos RSU	11
2.1.5.6 Definição de rotas	12

2.1.5.7 Método de redimensionamento de rotas	13
2.1.6 Veículos e recursos humanos para a recolha de RSU	16
2.2 SIG - Sistemas de informação geográfica	18
2.2.1 Definição	18
2.2.2 Evolução histórica	19
2.3 Aplicações existentes	21
2.4 Referências do capítulo	23
Capítulo 3 - Métodos de Cálculo	24
3.1 Abordagem geral sobre os problemas de optimização	25
3.2 O problema do caixeiro viajante (PCV)	26
3.3 O Problema da recolha	28
3.3.1 Definição matemática do problema do PCV	28
3.3.2 Métodos exactos	29
3.3.2.1 Algoritmo Dijkstra	29
3.3.2.2 Algoritmo de Floyd	36
3.3.2.3 Método Húngaro	40
3.3.2.4 Programação linear inteira	43
3.3.2.5 Programação dinâmica	45
3.3.3 Métodos heurísticos	46
3.3.3.1 Métodos de construção de rotas	47

3.3.3.1.1 Método de Clarke and Wright	48
3.3.3.1.2 Método da inserção	49
3.3.3.1.3 Obtenção de sementes	51
3.3.3.2 Métodos de melhoria de rota	52
3.3.3.2.1 K-opt	52
3.3.3.2.2 Recozimento simulado	54
3.3.3.2.3 Método de Monte Carlo	56
3.3.3.2.4 Algoritmos genéticos	56
3.3.3.2.5 Busca tabu	58
3.3.3.3 Algoritmos compostos	60
3.4 O problema dos m - caixeiros viajantes	61
3.5 Formulação clássica de atribuição de rotas	63
3.6 Abordagem ao problema de recolha de RSU	65
3.6.1 Introdução	65
3.6.2 Definição	66
3.6.3 Formulação do problema usando programação linear	67
3.6.4 Programação linear binária	69
3.7 Referências do capítulo	74
Capítulo 4 - Aplicação da Programação Linear Binária ao Problema da Recolha de RSU	82

4.1 Casos de estudo simples	83
4.1.1 Introdução	83
4.1.2 Resultados	85
4.1.3 Comentários	90
4.2 Casos de estudo mais complexos	90
4.2.1 Introdução	90
4.2.2 Criação de sub-áreas	91
4.2.3 Aplicação do algoritmo de programação binária	92
4.2.4 Aplicação dum método para a resolução do problema do caixeiro viajante	98
4.2.5 Conclusão	100
4.3 Método alternativo usando apenas programação linear binária	101
4.3.1 Introdução	101
4.3.2 Descrição do método	101
4.3.3 Aplicação do método	103
4.3.4 Conclusão	106
Capítulo 5 - Aplicação Informática MVBAGIS e Caso de Estudo Real	107
5.1 Aplicação informática MVBAGIS	108
5.2 Aplicação ao caso do programa ECOFONE da LIPOR	118
Capítulo 6 - Conclusões e Trabalhos Futuros	132
6.1 Síntese final	133

6.2 Trabalhos futuros	133
6.2.1 Algoritmos de optimização	134
6.2.2 Integração com sistemas de informação geográfica	134
Anexo A - Noções Gerais sobre Grafos	136
A.1 Noções básicas	136
A.2 Representação matricial de um grafo	138
A.3 Referências do capítulo	139

Lista de Figuras

Figura 2.1 - Volume de RSU produzido em cada rua.	12
Figura 2.2 - Percurso obtido por um método heurístico.	12
Figura 2.3 - Cidade dividida em três sub-áreas.	13
Figura 3.1 - Passo na intersecção k.	31
Figura 3.2 - Caminhos mais curtos.	32
Figura 3.3 - O último vértice antes de w é x.	33
Figura 3.4 - Grafo direccionado.	35
Figura 3.5 - Aplicação do algoritmo de Floyd a um grafo direccionado.	39
Figura 3.6 - Troca 2-Opt. - Esquerda: rota original; Direita: rota resultante da troca.	53
Figura 3.7 - Troca 3-Opt. - Esquerda: rota original; Direita: rotas resultantes das trocas.	53
Figura 3.8 - Percursos diferentes para cada caixeiro.	61
Figura 3.9 - Rede contraída.	62
Figura 3.10 - Rede expandida.	62
Figura 4.1 - Janela inicial do programa implementado para validação do modelo de optimização.	84
Figura 4.2 - Ligações possíveis e custos entre os seis pontos.	84
Figura 4.3 - Distribuição do volume de RSU por super-contentores.	92
Figura 4.4 - Distribuição do volume de RSU por super-contentores no concelho de Gondomar.	94

Figura 4.5 - Distribuição dos pontos de recolha para o veículo 1.	99
Figura 4.6 - Representação gráfica de alguns percursos entre os vários pontos de recolha.	99
Figura 4.7 - Percurso optimizado do veículo 1 na recolha dos super-contentores que lhe foram afectados.	100
Figura 4.8 - Exemplo de formação de super-contentores e super-veículos.	102
Figura 4.9 - Distribuição dos pontos de recolha dos super-contentores 1 e 2 afectos ao super-veículo 0.	105
Figura 5.1 - Janela inicial da aplicação MVBAGIS.	108
Figura 5.2 - Janela de edição e de criação de ficheiros <i>Shape</i> .	109
Figura 5.3 - Fluxograma relativo à edição e criação dum ficheiro <i>Shape</i> .	110
Figura 5.4 - Fluxograma de inserção de caminhos, dados geográficos e logísticos.	113
Figura 5.5 - Fluxograma 1.1 (parte do fluxograma da Figura 5.4).	114
Figura 5.6 - Fluxograma 1.2 (parte do fluxograma da Figura 5.4).	115
Figura 5.7 - Continuação do fluxograma 1.2 (parte do fluxograma da Figura 5.4).	116
Figura 5.8 - Janela de introdução das características da frota.	116
Figura 5.9 - Janela de introdução das características dos pontos de recolha.	117
Figura 5.10 - Exemplo de um ponto super contentor.	118
Figura 5.11 - Área de aplicação do algoritmo de Floyd para obtenção dos caminhos entre os quatro pontos definidos.	119
Figura 5.12 - Percurso do veículo XP-00-00, distância percorrida e respectivos custos.	120

Figura 5.13 - Relatório do percurso do veículo XP-00-00 e distância percorrida.	121
Figura 5.14 - Percurso do veículo XP-00-00, distância percorrida e respectivos custos.	122
Figura 5.15 - Percurso do veículo XP-00-01, distância percorrida e respectivos custos.	122
Figura 5.16 - Relatório do percurso do veículo XP-00-00 e distância percorrida.	123
Figura 5.17 - Relatório do percurso do veículo XP-00-01 e distância percorrida.	124
Figura 5.18 - Percurso do veículo XP-00-01, distância percorrida e respectivos custos.	126
Figura 5.19 - Relatório do percurso do veículo XP-00-01 e distância percorrida.	127
Figura 5.20 - Percurso do veículo XP-00-00, distância percorrida e respectivos custos.	128
Figura 5.21 - Percurso do veículo XP-00-01, distância percorrida e respectivos custos.	128
Figura 5.22 - Relatório do percurso do veículo XP-00-00 e distância percorrida.	129
Figura 5.23 - Relatório do percurso do veículo XP-00-01 e distância percorrida.	130
Figura 5.24 - Trajecto do veículo entre Lipor-ponto de partida e ponto final-Lipor.	131
Figura A.1 - Grafo direccionado.	136
Figura A.2 - Grafo não direccionado.	136
Figura A.3 - Grafo misto.	137
Figura A.4 - Ilustração de um ciclo (a_1, a_3, a_4, a_7 formam um ciclo).	138

Lista de Tabelas

Tabela 2.1 - Percursos actuais e suas especificações.	14
Tabela 3.1 - Matriz de custos.	35
Tabela 3.2 - Resultados da aplicação do algoritmo de Dijkstra.	36
Tabela 3.3 - Melhores caminhos encontrados.	36
Tabela 3.4 - Exemplo de aplicação do algoritmo de Floyd.	39
Tabela 3.5 - Tabela de custos do exemplo em causa.	42
Tabela 3.6 - Tabela obtida após passo 2.	42
Tabela 3.7 - Tabela obtida após passo 3.	42
Tabela 3.8 - Resultados da aplicação da programação dinâmica ao grafo da Figura 3.5.	46
Tabela 3.9 - Exemplo de aplicação da programação linear binária.	72
Tabela 4.1 - Resultados da aplicação do programa a diferentes casos simples.	85
Tabela 4.2 - Resultados da aplicação do programa aos super-contentores.	93
Tabela 4.3 - Resultados da aplicação informática aos super-contentores nas freguesias do concelho usando uma carga máxima de duas unidades.	95
Tabela 4.4 - Resultados da aplicação informática aos super-contentores nas freguesias do concelho usando uma carga máxima de quatro unidades.	96
Tabela 4.5 - Resultados da aplicação informática aos super-contentores nas freguesias do concelho usando uma carga máxima de duas unidades e apenas uma ligação possível entre os super-contentores.	97

Tabela 4.6 - Resultados da aplicação do programa aos super-contentores nas freguesias do concelho usando uma carga máxima de quatro unidades e apenas uma ligação possível entre os super-contentores.	98
Tabela 4.7 - Resultados da aplicação do programa aos super-contentores e super-veículos nas freguesias do concelho de Gondomar.	103
Tabela 4.8 - Resultados da aplicação do programa aos contentores correspondentes aos super-contentores 1 e 2.	105
Tabela 5.1 - Pontos de recolha do programa Ecofone de uma equipa durante um dia.	125

Capítulo 1 - Introdução

Sumário do Capítulo

Este capítulo tem por objectivo introduzir o problema da recolha de resíduos sólidos em zonas urbanas, bem como descrever o objectivo e estrutura deste trabalho.

1.1 Motivação

Nos últimos anos, a densidade populacional em zonas urbanas tem crescido duma forma desmesurada, aumentando a produção de resíduos sólidos urbanos. Assim, o serviço de recolha cresce a cada dia que passa. Acresce ainda o facto dos resíduos sólidos urbanos serem depositados diariamente em contentores, não podendo permanecer ali sem serem recolhidos, dado todos os riscos inerentes à sua permanência.

Devido ao grande número de pontos de recolha espalhados pelas zonas urbanas, bem como ao número de veículos disponíveis para recolha, torna-se necessário determinar circuitos óptimos de recolha com vista a reduzir os tempos de rota dos veículos e os custos inerentes.

O problema de optimização de rotas de veículos é um problema de optimização importante sendo a sua aplicação muito vasta. Nalgumas empresas de recolha de resíduos, o custo da recolha chega a representar 80% do custo total da empresa, uma percentagem elevada, motivo pelo qual as empresas estão cada vez mais interessadas em aumentar a sua produtividade, evitando ao máximo a sua inércia.

Já existem vários estudos sobre o problema da optimização de rotas de veículos, no entanto o grau da sua implementação a casos práticos ainda não é elevado. As particularidades que cada empresa apresenta torna os problemas de optimização de rotas de difícil resolução, fazendo que cada caso tenha que ser estudado separadamente.

Com o desenvolvimento da tecnologia existem actualmente ferramentas que dão a possibilidade às empresas de possuir mapas digitais das cidades, permitindo com isto a localização dos pontos e também a viabilização do processo de optimização das rotas.

No problema da optimização de rotas de veículos, o conjunto de pontos a visitar deve ser dividido em áreas ou regiões de atendimento, formando agrupamentos de pontos de visita por forma a poder decidir quantos veículos serão necessários para atender cada região e posteriormente como serão estabelecidas as rotas.

O objectivo principal dos problemas da optimização de rotas é a obtenção da distância total mínima percorrida, assim como a frota mínima de veículos utilizados para efectuar a recolha pretendida. Pretende-se obter o custo total mínimo e maximizar a produtividade.

Os veículos possuem restrições em termos de capacidade e de tempo, podendo iniciar e terminar as suas rotas num mesmo depósito central, o que constitui um problema característico de múltiplos caixeiros-viajantes, ou ainda ter o ponto inicial distinto do ponto final.

Existem vários algoritmos para a construção de rotas alguns dos quais são descritos e utilizados na resolução deste problema.

1.2 Objectivo da tese

O objectivo da presente tese é o desenvolvimento de aplicações informáticas, suportadas em sistemas de informação geográfica, que permitam a determinação de percursos de recolha de resíduos sólidos urbanos, numa dada região, com a subsequente redução dos custos inerentes à recolha.

O trabalho subjacente a esta tese evoluiu em 4 fases:

1. Identificação de todos os pontos de recolha (contentores), definindo as informações de cada um, nomeadamente: a localização, a capacidade, o volume diário, bem como a informação relativa à necessidade da recolha ser efectuada com maior ou menor prioridade;
2. Introdução da informação descrita anteriormente num sistema de informação geográfica;
3. Definição, no sistema de informação geográfica, das possíveis ligações entre os vários contentores bem como das características de cada ligação;
4. Identificação da frota disponível com vista à optimização dos circuitos de recolha. É necessário fornecer todas as informações sobre a frota, no que se refere às características técnicas de cada veículo, assim como as necessidades humanas para que esse veículo possa operar.

1.3 Estrutura da tese

A tese encontra-se dividida em seis capítulos e um anexo.

No Capítulo 1 são apresentadas as motivações e os objectivos do presente trabalho.

No Capítulo 2 faz-se uma descrição das noções básicas necessárias à formulação do problema de recolha de resíduos sólidos urbanos e aborda-se uma das formas como as entidades responsáveis pela recolha podem manualmente ultrapassar o problema. Além disso, referem-se também as particularidades e as dificuldades encontradas na realização deste serviço. Neste capítulo é ainda feita uma abordagem aos sistemas de informação geográfica (SIG), a sua definição e implementação. São igualmente apresentadas as ferramentas informáticas actuais mais comuns que podem ser usadas para ultrapassar o problema em causa.

O Capítulo 3 começa com o estudo do problema da optimização, passando para o estudo do método do caixeiro viajante e dos métodos aplicáveis à sua resolução. Em seguida, é feita a abordagem ao problema do múltiplo caixeiro viajante e sua resolução. Finalmente, faz-se a descrição matemática do problema em causa, sua resolução e implementação usando programação linear binária.

No Capítulo 4 apresenta-se a validação do algoritmo implementado no Capítulo 3 aplicando o algoritmo à resolução de casos simples, bem como as metodologias utilizadas para reduzir o tempo de execução no caso de problemas mais elaborados ou mais específicos, apresentados no Capítulo 5.

No Capítulo 5 descreve-se a aplicação informática desenvolvida que serve de suporte a esta tese e testa-se a aplicação informática com um caso de estudo real relativo ao programa ECOFONE da LIPOR.

No Capítulo 6 apresentam-se as principais conclusões desta tese e enumeram-se algumas sugestões para trabalho futuro.

No Anexo A é feita uma abordagem à teoria de grafos, importante para o conhecimento de alguns conceitos básicos que são usados ao longo deste trabalho.

Capítulo 2 - Noções Gerais do Problema de Recolha de Resíduos Sólidos Urbanos / Sistemas de Informação Geográfica

Sumário do Capítulo

Este capítulo tem por objectivo descrever o problema da recolha de resíduos sólidos urbanos (RSU), vulgarmente denominados de lixos, e a forma como são tratados nos dias de hoje na quase totalidade das zonas urbanas. Apresenta ainda uma gestão manual das rotas (também designados de percursos) e da recolha dos resíduos, assim como o pessoal e equipamentos envolvidos nesta operação.

Por fim, apresentam-se os SIG (Sistemas de Informação Geográfica), assim como a sua evolução e os contributos para o problema em causa. São apresentadas algumas das ferramentas informáticas mais conhecidas, associadas aos sistemas de informação geográfica e que servem de base de desenvolvimento para a automatização de tarefas nas áreas em estudo.

2.1 Recolha e transporte de resíduos sólidos urbanos (RSU)

2.1.1 Definição

Recolher os RSU não é mais do que proceder ao levantamento e transporte dos RSU a partir da sua origem, mediante transporte adequado, para uma possível estação de transferência, um local de tratamento ou ainda para um depósito final. A recolha dos RSU realiza-se devido a questões ambientais e de saúde pública.

A recolha de RSU produzidos em zonas residenciais, estabelecimentos públicos e comerciais é da responsabilidade do pelouro municipal encarregado da limpeza urbana. Nesses serviços, podem usar-se recursos próprios do município, de empresas contratadas, ou sistemas mistos, como por exemplo o aluguer de viaturas e a utilização de mão-de-obra do município.

2.1.2 Regularidade

A recolha dos RSU domiciliário deve ser efectuada porta a porta ou a partir de depósitos colocados em locais estratégicos de acordo com as zonas. Esta deverá ser efectuada regularmente nos mesmos dias e horários, por forma a criar hábitos nas pessoas no que se refere à colocação dos RSU nos locais de recolha. Desta forma evitará que este se aglomere e fiquem expostos a não ser durante o tempo necessário à sua recolha.

Os RSU deverão ser colocados em locais ou depósitos predefinidos devido a questões ambientais e de saúde pública. O tempo de exposição dos RSU é uma questão importante devido a todos os aspectos ambientais, odores, aspectos sanitários. Isto justifica que, a recolha regular é um dos aspectos mais importantes do serviço.

Em qualquer cidade onde se proceda à pesagem dos RSU recolhidos, é possível verificar se a recolha é regular, comparando os pesos dos RSU recolhidos, nos mesmos dias da semana, para duas ou mais semanas consecutivas. Outra forma de

avaliar a regularidade da recolha é através dos quilómetros percorridos pelos veículos. Estes devem ser semelhantes pois os trajectos serão os mesmos.

Além disso, a ocorrência de pontos de acumulação de RSU e a existência de reclamações reflectem claramente qualquer irregularidade da recolha. O ideal, seria portanto, um sistema de recolha, com dias e horários determinados e de pleno conhecimento da população, de forma a criar uma imagem credível da recolha. Desta forma evitava-se que os RSU fossem colocados em locais impróprios. Os RSU passavam a ser acondicionados em embalagens adequadas e colocados, nos dias e horários marcados, daí resultando grandes benefícios em termos de higiene ambiental, saúde pública, limpeza e preservação da imagem dos locais públicos.

2.1.3 Frequência de recolha

Por razões óbvias, o tempo decorrido entre a produção dos RSU e o seu destino final não deve exceder uma semana para evitar, para além dos maus cheiros, a proliferação de moscas, roedores, insectos e outros animais (IBAM, 2001).

Em cidades que dispõem de estações de transferência, os RSU permanecem armazenados um ou dois dias até serem transportados para o aterro, onde são depositados. Por exemplo, se a recolha de RSU for efectuada três vezes por semana, os RSU produzidos no sábado só irão ser recolhidos na terça-feira seguinte (três dias depois). Se se considerar mais dois dias para serem transferidos e mais um dia para serem depositados no aterro, o total de dias decorridos entre a formação e o destino final pode atingir os seis dias. A frequência mínima de recolha admissível num país de clima quente é, portanto, de três vezes por semana.

Outros pontos a ter em atenção é a capacidade de armazenamento dos resíduos nos domicílios, nos pontos de recolha e em pontos onde não é possível o armazenamento dos RSU por mais de um dia, como é o caso dos centros das cidades, onde os estabelecimentos comerciais e de serviços, além da falta de local apropriado para o armazenamento, produzem RSU em quantidades consideráveis. Em todas as situações, é conveniente estabelecer que a recolha deverá ser feita diariamente.

2.1.4 Horários de recolha

Por uma questão económica, as recolhas devem ser realizadas em dois turnos (IBAM, 2001):

1º Turno - 2ª, 4ª e 6ª - $\frac{1}{4}$ dos percursos

3ª, 5ª e sábados - $\frac{1}{4}$ dos percursos

2º Turno - 2ª, 4ª e 6ª - $\frac{1}{4}$ dos percursos

3ª, 5ª e sábados - $\frac{1}{4}$ dos percursos

Por exemplo, se forem projectados 12 percursos de recolha, três vezes por semana, deve ser utilizada uma frota de $24/4 = 6$ veículos, mantendo uma reserva de pelo menos 10% da frota.

É conveniente definir turnos de 12 horas (dividindo-se o dia em dois turnos, mas trabalhando na prática cerca de oito horas por turno). Tem-se então, por exemplo, o primeiro turno a iniciar às sete horas e o segundo turno às 19 horas, sendo o tempo não utilizado, usado na reparação e manutenção dos veículos.

Nas zonas comerciais, a recolha deverá ser nocturna devido à grande afluência de pessoas e veículos. Já em zonas habitacionais, a recolha deverá ser efectuada em período diurno, uma vez que estas encontram-se menos sobrecarregadas de pessoas e veículos. Sempre que possível, principalmente em zonas de recolha porta a porta, deve proceder-se ao varrimento das ruas para recolher os eventuais resíduos deixados durante a recolha.

O controlo dos ruídos na recolha nocturna é uma das questões a ter em atenção. O volume das vozes, o comando de pára/arranca do veículo, deve ser efectuada através de interruptor luminoso, accionado na traseira do veículo e o silenciador deve estar em perfeito estado. O motor não deve funcionar a altas rotações na compactação, devendo existir um dispositivo automático de aceleração.

2.1.5 Redimensionamento de rotas

O aumento ou diminuição da população, as mudanças de características das zonas habitacionais e a não regularidade na recolha dos resíduos são factores que influenciam o redimensionamento das rotas de recolha dos resíduos sólidos urbanos. Deve-se ter em atenção os seguintes parâmetros:

- Número de elementos que compõem cada equipa;
- Equidade de rotas;
- Ponto de referência para o início da recolha;
- Avaliação dos pontos de recolha de RSU;
- Pesagem dos RSU;
- Definição das rotas.

2.1.5.1 Número de elementos que compõem cada equipa

Na maioria das cidades, o número de elementos que compõem cada equipa de recolha varia de dois a cinco trabalhadores por veículo, no caso de a recolha ser feita pelos municípios, e equipas de três trabalhadores, no caso de empresas privadas (IBAM, 2001).

2.1.5.2 Equidade de rotas

Cada equipa de recolha deve executar um mesmo volume de trabalho, que resulte num esforço físico equivalente. Nas áreas com RSU concentrados, os elementos carregam muito peso e percorrem uma extensão de rua menor.

Inversamente, em áreas com pequena concentração de RSU, cada equipa carrega pouco peso e percorre extensões maiores. Em ambos os casos, o trabalho realizado deverá ser sensivelmente o mesmo.

O método de redimensionamento que se descreve a seguir prevê a divisão da área a ser coberta em sub-áreas com densidades demográficas semelhantes, nas quais as concentrações de RSU (peso de RSU recolhido por distância percorrida, kg/m) variam

pouco. Nessas sub-áreas, deve-se fixar um tempo de trabalho igual. É também importante ter em conta o vigor físico dos elementos que compõem a equipa, devendo procurar-se um certo equilíbrio relativamente a este aspecto.

2.1.5.3 Ponto de referência para início da recolha

As rotas devem ser planeadas por forma a que as recolhas tenham como ponto inicial o ponto mais distante do local de destino dos RSU, e que o seu movimento seja feito na direcção deste, encurtando as distâncias (e o tempo) do percurso.

2.1.5.4 Avaliação dos pontos de recolha de RSU

É importante avaliar os pontos de recolha de RSU, habitações, estabelecimentos comerciais, etc. Estes dados são utilizados no dimensionamento das rotas de recolha regular. A pesquisa deve ser efectuada com base nos censos disponíveis, podendo-se calcular a quantidade média dos RSU gerados por pessoa e por dia.

O índice de produção individual de RSU deve ser determinado com certo rigor técnico. Estima-se que este valor situa-se entre 0.35 a 1.00kg por pessoa e por dia (IBAM, 2001).

Considere-se o exemplo em que os RSU gerados por pessoa/dia é de 0.70kg, e a população é de 50 mil habitantes; o peso dos RSU a ser recolhido é:

$$50 \text{ mil hab.} \times 0.70 \text{ kg/hab./dia} = 35\,000 \text{ kg/dia}$$

Este valor torna-se importante na definição do número de veículos a utilizar.

Na prática, o redimensionamento das rotas poderá ter em conta outros parâmetros, tendo em consideração, por exemplo, a dispersão dos RSU, o tipo de veículos, etc.

Após o redimensionamento, as novas rotas devem ser implementadas e ajustadas, passadas algumas semanas, em relação a alguns aspectos que necessitem de ser

alterados. Alguns exemplos são a redefinição de trajectos, o número de pessoas afectas a cada rota, o tipo de veículo, etc.

2.1.5.5 Pesagem dos RSU

A pesagem dos RSU deve ser efectuada num recipiente uniforme com capacidade conhecida e para o qual deve ser vazado todos os RSU recolhidos em cada ponto.

Estes devem ser depositados no contentor do veículo, contando o número de recipientes necessários para recolher todo os RSU de uma rua.

A pesagem deverá ser efectuada da seguinte forma (IBAM, 2001):

- Efectuar as pesagens nos dias de maior produção de RSU (normalmente segundas e terças feiras);
- Anotar em planta topográfica o volume de RSU produzidos em cada rua;
- Somar sucessivamente o número de recipientes em cada rua conforme a rota definida até que a capacidade do veículo seja atingida em cada viagem de cada turno. A avaliação da capacidade do veículo em termos de recipientes permite definir o número de recipientes que enche um veículo bem como o número de viagens de cada veículo por turno. Devem evitar-se, na determinação das rotas, percursos improdutivos do veículo ao longo dos quais não haja recolha de RSU. A escolha do percurso deve ser efectuada de forma a minimizar a energia despendida pelos elementos da equipa bem como os gastos do veículo;
- Colocar em prática as novas rotas, medindo os tempos, a fim de proceder aos ajustes necessários.

Conforme ilustrado na Figura 2.1, é possível criar um mapa com indicação dos valores volumétricos da distribuição dos RSU pelas várias ruas da cidade (IBAM, 2001).

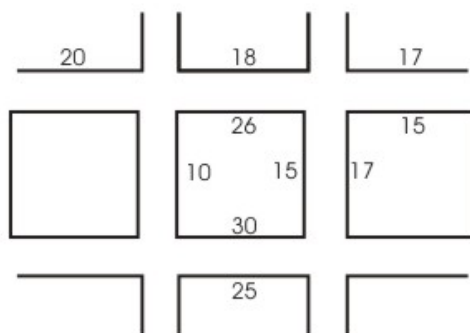


Figura 2.1 - Volume de RSU produzido em cada rua.

2.1.5.6 Definição de rotas

Uma rota deverá ser definida por um processo iterativo em que a solução óptima tenha em conta as várias implicações tais como o sentido do tráfego das ruas, os percursos duplicados e improdutivos. Na definição das rotas é habitual a utilização de métodos heurísticos.

A Figura 2.2 exemplifica um percurso razoável (obtido por um método heurístico) para um dado percurso:

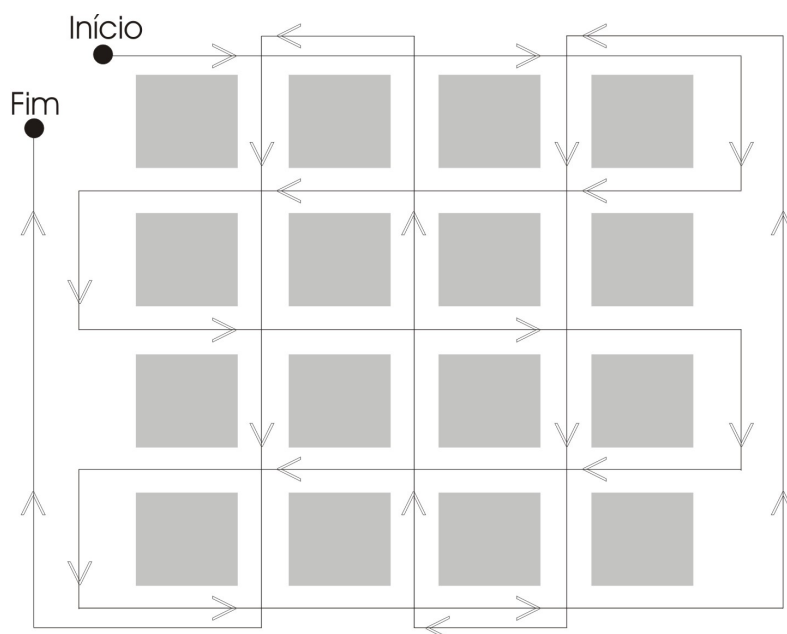


Figura 2.2 - Percurso obtido por um método heurístico.

2.1.5.7 Método de redimensionamento de rotas

Um método para o redimensionamento consiste em:

- Dividir a cidade em sub-áreas;
- Efectuar o levantamento e sistematizar as características de cada rota;
- Analisar as informações obtidas;
- Redimensionar as rotas, tendo em conta:
 - As horas extras de trabalho;
 - A definição das novas pesagens por turno;
 - As quantidades de RSU em cada área.

A divisão da cidade em sub-áreas é feita com base na densidade demográfica e por forma a que estas possuam densidades semelhantes. Por exemplo, nas sub-áreas I, II e III (Figura 2.3). Admita-se que a sub-área II contém 8 rotas de recolha, efectuadas em 2 turnos, 3 vezes por semana, pelo que serão necessários 2 veículos = $8/4$.

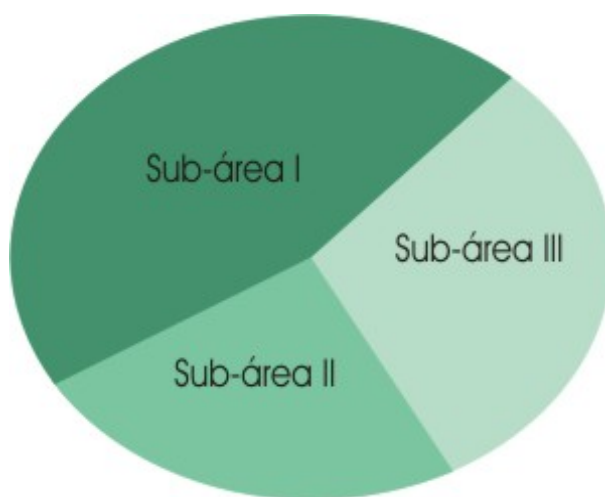


Figura 2.3 - Cidade dividida em três sub-áreas.

Considere-se a Tabela 2.1 com um exemplo para a definição anterior.

Tabela 2.1 - Percursos actuais e suas especificações.

Percursos actuais - segundas e terças-feiras							
Rota	Distância percorrida (m) (1)	Peso de RSU (Kg) (2)	Tempo médio de trabalho (h) (3)	Elementos da equipa (pessoa) (4)	Índices		
					kg/h (2)/(3)	kg/m* (2)/(1)	kg/pessoa (2)/(4)
01	14 250	16 400	8.20	4	2 000	1.15	4 100
02	13 180	14 200	7.72	4	1 839	1.08	3 550
03	14 600	17 300	8.75	4	1 977	1.18	4 325
04	16 410	19 500	8.99	4	2 169	1.19	4 875
05	15 120	18 100	9.78	4	1 851	1.20	4 525
06	18 040	17 400	8.65	4	2 012	0.96	4 350
07	13 870	15 600	9.36	4	1 667	1.12	3 900
08	15 660	18 300	10.01	4	1 828	1.17	4 575
Médias	15 141	17 100	8.93	4	1 915	1.13	4 275
Totais	-	153 900	-	-	-	-	-

* kg/m = concentração de RSU.

Verifica-se que os tempos de trabalho são elevados, resultando em horas extras.

Tendo em conta que se pretende efectuar a recolha em 8 horas de trabalho, evitando desta forma as horas extras. Assim, pode-se calcular os novos pesos a serem recolhidos por turno de trabalho, mantendo a mesma produtividade de cada equipa:

$$P = (\text{Peso de RSU recolhido por hora}) \times T_c \quad (2.1)$$

em que T_c o tempo escolhido para cada turno de trabalho, igual a 8 horas para o exemplo descrito a seguir e sendo P01, ..., P08, respectivamente os novos pesos recolhidos nas rotas 01, ..., 08, para os dados da Tabela 2.1.

Portanto:

$$P01 = 2\,000 \times 8 = 16\,000\text{kg}$$

$$P02 = 1\,839 \times 8 = 14\,712\text{kg}$$

$$P03 = 1\,977 \times 8 = 15\,816\text{kg}$$

$$P04 = 2\,169 \times 8 = 17\,352\text{kg}$$

$$P05 = 1\,851 \times 8 = 14\,808\text{kg}$$

$$P06 = 2\,012 \times 8 = 16\,096\text{kg}$$

$$P07 = 1\,667 \times 8 = 13\,336\text{kg}$$

$$P08 = 1\,828 \times 8 = 14\,624\text{kg}$$

Peso total recolhido = 112 744kg

Peso médio recolhido = 15 343kg

Sendo o peso médio dos RSU a serem recolhidos de 15 343kg, restarão desta forma:

$$153\,900 - 112\,744 = 31\,156\text{kg}$$

Considerando que o peso médio dos RSU recolhidos, com as novas rotas é de 15 343 kg/rota, serão necessários:

$$31\,156\text{kg}/15\,343\text{kg} = 2.03$$

2.03 novos percursos ou, aproximadamente dois percursos a mais, sendo um nas segundas, quartas e sextas, e o outro nas terças, quintas e sábados.

Desta forma obtêm-se dez percursos, a média de peso por percurso passará a ser:

$$153\,900\text{ kg}/10\text{ percursos} = 15\,390\text{ kg/percurso futuro}$$

Os percursos futuros deverão ser assinalados no mapa tendo em consideração a concentração de RSU em cada área (expressa em kg/m).

Assim, aplica-se a expressão:

$$L \times C \approx P \quad (2.2)$$

com:

L = comprimento das ruas (m)

C = concentração de RSU (kg/m)

P = peso médio associado aos percursos futuros (kg)

até obter pesos de aproximadamente 15 390 kg/percurso.

Desta forma, o número de viaturas será igual ao n° percursos / 4 = 2.5 viaturas por turno (as mesmas do primeiro turno, ficando uma de reserva).

Podem-se assim usar três viaturas no primeiro turno e duas viaturas no segundo turno.

O tipo e a capacidade das três viaturas a serem utilizadas dependerão do número de viagens possíveis ao local de deposição final. Se, por exemplo, forem possíveis duas viagens às segundas e terças, a carga média por viagem será de $15\,390/2 = 7\,695\text{kg}$.

Supondo que, numa dada região, ocorre um aumento médio de 20% na quantidade de RSU no verão, pode-se avaliar um acréscimo na produção de RSU de cerca de 40%. Cada rota teria, portanto, no verão, $15\,390 \times 1.4/2 = 10\,773\text{kg}$. Se forem utilizados veículos com capacidade para 12 toneladas/viagem, a recolha poderá ser realizada com folga e regularidade.

2.1.6 Veículos e recursos humanos para a recolha de RSU

As viaturas de recolha e transporte de RSU podem ser de dois tipos:

- Compactadoras: possuem equipamentos de compactação de carga;
- Sem compactação: conhecidos por veículos de recolha hermética.

Os veículos de recolha devem possuir as seguintes características (Higiene e Segurança no Trabalho, 2004):

- Não permitir o derramamento dos RSU na via pública;
- Possuir, no mínimo, uma taxa de compactação de 3:1, ou seja, cada 3 m³ de resíduos ficarão reduzidos, por compactação, a 1 m³;
- Apresentar a altura de carregamento ao nível da cintura dos elementos da equipa, ou seja, no máximo a 1.20 m de altura em relação ao solo;
- O carregamento deverá ser feito na parte traseira;
- Dispor de local adequado para transporte dos trabalhadores;
- Apresentar descarga rápida dos RSU no destino;
- Possuir compartimento de carga com capacidade para no mínimo 1.5 m³;
- Possibilitar carregamento de contentores de diversos tipos;
- Distribuir adequadamente a carga no chassis do veículo de transporte;
- Apresentar capacidade adequada para o menor número de viagens ao destino, nas condições de cada área.

No que se refere aos recursos humanos que efectuem a recolha, deverão possuir características físicas capazes, gozar de perfeita saúde e utilizar todos os meios de protecção individual capazes de assegurar a sua integridade. Os riscos biológicos, que se relacionam com a presença no ambiente de trabalho de microrganismos (tais como bactérias, vírus, fungos e bacilos), que podem penetrar no organismo do homem por via digestiva, respiratória, olhos e pele, são responsáveis por algumas doenças profissionais, podendo dar origem a doenças menos graves como infecções intestinais ou a simples gripe, ou mais graves como a hepatite, meningite ou sida (Higiene e Segurança no Trabalho, 2004).

Como estes microrganismos se adaptam melhor e se reproduzem mais em ambientes sujos, as medidas preventivas a tomar terão de ser relacionadas com:

- A rigorosa higiene de mecanismos de trabalho,
- A rigorosa higiene de corpo e das roupas;
- Destruição por processos de elevação da temperatura (esterilização) ou uso de cloro;
- Uso de equipamentos individuais para evitar contacto directo com os microrganismos;
- Ventilação permanente e adequada;
- Controlo médico constante;
- Vacinação sempre que seja necessário.

2.2 SIG - Sistemas de informação geográfica

2.2.1 Definição

Os sistemas de informação geográfica (SIG) têm sido objecto de variadas definições. Alguns autores baseiam-se nas características tecnológicas e na funcionalidade genérica destes sistemas para os distinguir de outros tipos de sistemas. Assim, Burrough (1986) define SIG como sendo um conjunto de ferramentas para recolher, armazenar, aceder, transformar e visualizar dados espaciais do mundo real. Esta definição caracteriza de uma forma genérica os produtos que se destinam ao suporte de aplicações envolvendo dados geográficos.

Uma definição mais abrangente de SIG, não distinguindo sequer se o sistema se encontra automatizado ou não, é dada por Aronoff (1989) em que um SIG tem por objectivo a recolha, o armazenamento e a análise de objectos e fenómenos, cuja

localização geográfica constitui uma característica importante e fundamental para a análise do problema em causa.

Alguns autores definem SIG distinguindo fundamentalmente três tipos diferentes de perspectivas: a visão baseada em mapas, em bases de dados e na análise espacial (Maguire, 1991). Segundo este autor, a primeira perspectiva encara os SIG como sistemas para processamento e visualização de mapas (Tomlin, 1991). A segunda enfatiza a importância dos SIG terem subjacente uma base de dados bem desenhada e possuírem um sistema de gestão de bases de dados (SGBD) potente (Frank, 1988). A terceira distingue a capacidade dos SIG para efectuar análise espacial, defendendo a existência de uma ciência de informação espacial em alternativa à perspectiva tecnológica com que geralmente os SIG são abordados (Openshaw, 1991; Goodchild, 1992).

2.2.2 Evolução histórica

A recolha de informações sobre a distribuição espacial de propriedades significativas da superfície da Terra constitui, desde há muito, uma parte importante das actividades das sociedades organizadas (Burrough, 1986).

Efectivamente, a informação geográfica, organizada por temas, tem sido tradicionalmente apresentada sob a forma de mapas, desde as civilizações mais antigas. Recorrendo apenas a processos manuais, foi possível representar em folhas de papel o resultado das observações efectuadas sobre algumas características da superfície terrestre. Estas eram representadas por meio de pontos, linhas e áreas aos quais eram associados símbolos, cores e padrões, cujo significado era explicado numa legenda ou num texto. Com base neste tipo de mapas, era possível realizar alguns tipos de análise.

As primeiras operações de análise efectuadas tinham um carácter essencialmente qualitativo, já que se baseavam na mera observação visual e na intuição de quem efectuava a análise. Com mapas baseados numa escala, era também possível realizar algumas operações de análise quantitativa, basicamente relativas ao cálculo de distâncias e áreas.

Contudo, a utilidade dos mapas desenhados manualmente mostrou-se limitada por diversos factores. Um factor importante foi o pouco detalhe que a generalidade dos mapas desse tipo possuía; o custo de produção dos mapas desenhados manualmente levou à adopção preferencial de uma escala pequena e, consequentemente, à representação da informação com um nível elevado de generalização.

Paralelamente, estes mapas rapidamente se encontravam desactualizados; as alterações da realidade são frequentes e é impossível redesenhar a totalidade de um mapa sempre que tal sucede. Finalmente, as operações de análise espacial envolvendo diferentes temas é difícil e apenas teoricamente é possível efectuar manualmente a análise do resultado da combinação de mapas relativos a temas diversos, o que, na prática, só é possível para pequenos volumes de informação.

O recurso a meios computacionais para suporte de informação espacial iniciou-se no princípio da década de 60, com a codificação digital da informação que, tradicionalmente, apenas era representada sob a forma de mapas. Contudo, só os avanços no campo da tecnologia informática alcançados no início da década de 70, particularmente os relacionados com o acesso directo a discos, permitiram obter resultados significativos. Posteriormente, o enorme aumento de eficiência do processamento computacional permitiu o recurso a diversos tipos de análise espacial.

Actualmente, os SIG são utilizados como ferramentas de análise geográfica por excelência, já que permitem a integração de grandes volumes de informação espacial e de outros tipos num mesmo sistema e o seu tratamento conjunto. Esta integração tornou-se possível como resultado da convergência de várias disciplinas e técnicas tradicionais. De entre estas, podem citar-se como especialmente relevantes a Geografia, Cartografia, Fotogrametria, Detecção Remota, Agrimensão, Geodesia, Engenharia Civil, Matemática, Estatística, Investigação Operacional, Informática, e, dentro desta, as áreas de CAD (*Computer Aided Design*), Computação Gráfica, Sistema de Gestão de Bases de Dados, Redes e Inteligência Artificial.

As empresas ligadas à gestão e aproveitamento de florestas, exploração de redes de transportes e telecomunicações e os organismos responsáveis por planeamento urbanístico, vias de comunicação e defesa do ambiente encontram-se entre os principais utilizadores de SIG.

A limitação na sua utilização reside essencialmente na necessidade de um grande investimento. Este deve-se, em parte, à aquisição do equipamento informático de suporte, mas sobretudo ao trabalho de recolha e introdução dos dados. Esta fase implica, por enquanto, o recurso intensivo a processos manuais que exigem uma elevada especialização por parte de quem os executa. A resolução de qualquer problema envolve sempre um grande volume de informação geográfica. Como foi já referido, a utilização crescente de técnicas de detecção remota a partir de satélites fornece, actualmente, o maior volume de informação. Contudo, este tipo de informação não é adaptado a todos os problemas e, mesmo nos casos em que é considerado adequado, exige várias fases de pré-processamento e o recurso a diversas técnicas de processamento para que os dados possam ser utilizados (Aronoff, 1989).

2.3 Aplicações existentes

Na indústria dos transportes, a análise geográfica é a chave para obter melhores decisões, seja a monitorizar sistemas de caminhos-de-ferro e condições das estradas, a descobrir a melhor rota para entregar mercadorias e serviços, gerir e monitorizar frotas, ou para fazer a manutenção de redes de transportes.

A percepção destes procedimentos numa perspectiva geográfica é crucial para a gestão de custos e de recursos, pois estes são a chave deste negócio.

Os profissionais do sector dos transportes e logística utilizam os Sistemas de Informação Geográfica para integrarem análises de mapas como suporte à decisão de planeamento e análise de redes de transportes, monitorização de veículos e optimização de rotas, inventário de monitorização, planeamento de rotas e análise, entre outros.

Como ajuda à resolução destes problemas, existem várias aplicações informáticas que servem de base de trabalho para o desenvolvimento de soluções específicas. No entanto, estas aplicações são genéricas e muitas vezes pouco específicas para as necessidades particulares de cada cliente. Abrangem aspectos genéricos e comuns aos mais variados problemas de optimização, esquecendo outros muito específicos do

problema que possa estar em causa, pois não se pode esquecer que cada caso é um caso.

Os mais importantes são:

- ArcGis - representado em Portugal pela Empresa ESRI, sobre o qual já desenvolveu várias aplicações de onde se pode destacar o ArcLogistics Route, que é uma solução para logística e que permite a definição optimizada de percursos de entregas por veículos, destinada a utilizadores finais. Com o ArcLogistics Route é possível alocar um veículo a cada ponto de entrega, e estabelecer a melhor sequência de paragens, tendo em conta as janelas temporais dos clientes, minimizando o tempo de percurso total. Esta optimização dos percursos de entrega pode reduzir entre 10 a 40% os custos associados aos processos de entregas.
- Autodesk - representado em Portugal pela Empresa MICROGRAF, sendo os mais importantes o AutoCAD Map 3D, o Autodesk MapGuide e o AutoCAD Raster.

2.4 Referências do capítulo

Aronoff, S., Geographic Information Systems: a management perspective, Ottawa, Canada: WDL Publications, 1989.

Burrough, P., Principles of Geographical Information Systems for Land Resources Assessment, Oxford: Clarendon Press, 1986.

Frank, A.U., Requirements of a database management system for a GIS. Photogrammetric Engineering and Remote Sensing, 54, 1557-1564, 1988.

Goodchild, M. F., Geographical Information Science. International Journal of Geographical Information Systems, 6, 31-45, 1992.

Higiene e Segurança no Trabalho - Manual de Formação - Programa Formação PME, 2004

IBAM, Gestão Integrada de Resíduos Sólidos - Manual de Gerenciamento de Resíduos Sólidos. Publicação elaborada pelo Instituto Brasileiro de Administração Municipal, sob o patrocínio da Secretaria Especial de Desenvolvimento Urbano da Presidência da República - SEDU/PR, 61-73, 2001.

Maguire, D. J., An overview and definition of GIS. Em Geographical Information Systems, 1 - Principles, editado por D. J. Maguire, M. F. Goodchild, D. W. Rhind, London: Longman, 9-20, 1991.

Openshaw, S., Developing appropriate spatial analysis methods for GIS. Em Geographical Information Systems, 1 - Principles, editado por D. J. Maguire, M. F. Goodchild, D. W. Rhind, London: Longman, 389-402, 1991.

Tomlin, C. D., Cartographic Modelling. Em Geographical Information Systems, 1 - Principles, editado por D. J. Maguire, M. F. Goodchild, D. W. Rhind, London: Longman, 361-374, 1991.

Capítulo 3 - Métodos de Cálculo

Sumário do Capítulo

Este capítulo descreve os principais métodos de resolução do problema de optimização da recolha de resíduos sólidos urbanos. Apresenta várias formas de atacar o problema, assim como uma abordagem com o problema do caixeiro viajante e múltiplo caixeiro viajante. Termina com a implementação de um algoritmo de determinação dos circuitos óptimos de recolha de RSU em zonas urbanas.

A formulação do problema consiste na especificação de um conjunto de pontos de recolha, situados em diferentes locais, que devem ser visitados por veículos de recolha, partindo de um ponto comum (parque) e regressando a um ponto final (depósito).

O objectivo principal é a determinação dos vários circuitos de recolha percorridos pelos vários veículos, minimizando a distância percorrida por cada um, utilizando o menor número de veículos necessários à recolha e optimizando os custos relacionados com o pessoal e veículos.

O Anexo A apresenta algumas noções sobre grafos, importantes para a compreensão de alguns métodos descritos neste capítulo.

3.1 Abordagem geral sobre os problemas de optimização

Os principais métodos usados na resolução dos problemas de optimização dividem-se em métodos exactos e heurísticos. Os métodos exactos, são aplicados a casos pequenos e com poucas restrições (Bondin et al.,1983). Por sua vez, os métodos heurísticos aplicam-se a casos mais complexos e com um número de restrições mais elevado (Golden et al.,1985).

A classificação dos algoritmos em termos de complexidade é feita utilizando-se o conceito de limitação polinomial. Algoritmos polinomiais (ou pertencentes à classe-P) são aqueles em que o número de operações elementares necessárias na obtenção da solução óptima de um dado problema é limitado, no pior dos casos, por uma função polinomial do tamanho do problema. Os problemas para os quais não se conhecem algoritmos polinomiais capazes de obter a solução exacta são classificados como NP-Completo (ou NP-Hard), e são considerados complexos e de difícil tratamento. O problema do caixeiro viajante (PCV) é um exemplo típico de problema NP-Completo. Portanto, os algoritmos exactos são usados apenas na obtenção de soluções de problemas de pequena dimensão. Para problemas de maior dimensão, utilizam-se métodos heurísticos.

Os algoritmos de *Branch and Bound*, a Programação Dinâmica e a Programação Inteira são os principais métodos exactos usados no problema do caixeiro viajante (Laporte, 1992; Nemhauser et al., 1988).

Os métodos heurísticos fornecem soluções aproximadas, fazendo com que a solução obtida necessite de um processo de verificação (Golden et al.,1985).

Para realçar a forma de funcionamento deste tipo de algoritmos, considere-se os seguintes exemplos de métodos heurísticos:

1. O algoritmo de *Sweep* (Billy et al., 1971) - Os pontos de passagem que formam cada rota são escolhidos de acordo com o ângulo da coordenada polar de cada local. São usados procedimentos iterativos para melhorar a distância total de todas as rotas. O número de cálculos a efectuar aumenta linearmente com o número de locais de recolha em cada rota.

2. O algoritmo do vizinho mais próximo (Bodin et al., 1983) - O local de partida é escolhido aleatoriamente e vai-se adicionando à rota o local mais próximo do último, repetindo este processo até que um circuito Hamiltoniano seja formado.
3. O algoritmo de Inserção (Rosenkrantz et al., 1977) - Com base em sub-rotas iniciais, procura-se o local a ser inserido numa sub-rota, procurando a melhor posição para a inserção deste local. As sub-rotas iniciais podem ser os próprios locais. A forma de inserção dá origem a diferentes variantes, por exemplo inserção mais próxima, inserção arbitrária (Bodin et al., 1983).

3.2 O problema do caixeiro viajante (PCV)

O problema do caixeiro viajante é um exemplo clássico dos problemas de optimização combinatória (Karp, 1975). A primeira consideração que se pode pensar para resolver esse tipo de problema é reduzi-lo a um problema de enumeração: descobrem-se todas as rotas possíveis e, usando um computador, calcula-se o comprimento de cada uma delas, e então vê-se qual a menor. É claro que ao descobrir todas as rotas estar-se-á a enumerá-las, e daí poder dizer-se que se está a reduzir o problema de optimização a um de enumeração.

Com o objectivo de encontrar o número $R(n)$ de rotas para o caso de n cidades, basta fazer um raciocínio combinatório simples e clássico. Por exemplo, no caso das $n = 4$ cidades, a primeira e última posição são fixas, de modo que elas não afectam o cálculo; na segunda posição pode colocar-se qualquer uma das 3 cidades restantes B, C e D, e uma vez escolhida uma delas, pode colocar-se qualquer uma das 2 restantes na terceira posição; na quarta posição já não existiria nenhuma escolha, pois sobra apenas uma cidade; consequentemente, o número de rotas é $3 \times 2 \times 1 = 6$. De modo semelhante, para o caso de n cidades, como a primeira é fixa, o número total de escolhas que se pode fazer é $(n-1) \times (n-2) \times \dots \times 2 \times 1$. Usando a notação de factorial, obtém-se $R(n) = (n - 1)!$.

Assim, a estratégia consiste em gerar cada uma dessas $R(n) = (n - 1)!$ rotas, calcular o comprimento total das viagens de cada rota e ver qual delas tem o menor

comprimento total. Supõe-se um trabalho fácil para um computador. Bem, talvez não; veja-se porquê.

Suponha-se que existe um computador capaz de fazer 10^9 adições por segundo. Isso parece uma velocidade imensa, capaz de tudo. Com efeito, no caso de 20 cidades, o computador precisa apenas de 19 adições para dizer qual o comprimento de uma rota e então será capaz de calcular $10^9 / 19 = 53$ milhões de rotas por segundo. Contudo, essa imensa velocidade é pequena relativamente ao tamanho do número $19!$ de rotas que precisará de examinar. Com efeito, o valor de $19!$ é 121 645 100 408 832 000 (ou, aproximadamente, 1.2×10^{17} em notação científica). Consequentemente, ele precisará de $1.2 \times 10^{17} / (53 \times 10^6) = 2.3 \times 10^9$ segundos, o que equivale a cerca de 73 anos. O problema reside no facto de $(n - 1)!$ crescer de uma forma acentuada.

O termo “Caixeiro Viajante” foi usado pela primeira vez no início da década de 1930 por Merrill Flood, motivado pela proposta de um modelo matemático para a teoria dos jogos. O primeiro artigo que foca o problema do caixeiro viajante foi publicado por Dantzing, Fulkerson e Johnson, em 1954, no Jornal da Sociedade de Pesquisa Operacional da América, sendo então o principal propulsor da Optimização Combinatorial.

O problema do caixeiro viajante tem como base a formulação de um problema em que um veículo parte de um depósito central e visita n localidades, uma única vez, para regressar em seguida ao depósito central, minimizando a distância percorrida. O problema pode ser visto de uma forma simétrica (considerando o sentido e direcção das ruas) ou assimétrico (direcção e sentido das ruas são consideradas).

A relevância do PCV é devida a pelo menos três das suas características, que são: sua grande aplicação prática (Bellmore e Nemhauser, 1968; Burkard, 1979; Reinelt, 1994), uma enorme relação com outros modelos (Laporte et al., 1996) e a sua grande dificuldade de solução exacta (Papadimitriou e Steiglitz, 1982; Gould, 1991 e Zhang, 1997). Dessa forma, a importância do modelo é indiscutível, tanto sob o aspecto prático, como teórico. Foi considerado intratável por Garey e Johnson (1979) e classificado por Karp (1975) como NP-Difícil.

A formulação do problema recorre à teoria dos grafos (ver Apêndice A). Seja um grafo $G(N,A,C)$, onde N designa o número de nós, A o número de arcos e C a matriz

de custos. Pode desta forma ser visto como um circuito Hamiltoniano em G de mínimo custo total.

3.3 O problema da recolha

Descrevem-se em seguida alguns métodos que podem ser utilizados para a resolução do problema, baseando-se na formulação do problema do PCV.

3.3.1 Definição matemática do problema do PCV

A formulação matemática do problema (PCV) estabelecida da seguinte forma (Golden et al., 1977; Bodin et al., 1983):

$$\text{Min } z = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (3.1)$$

$$\text{sujeito a } \sum_{j=1}^n x_{ij} = 1, i=1, 2, \dots, n. \quad (3.2)$$

$$\sum_{i=1}^n x_{ij} = 1, j=1, 2, \dots, n. \quad (3.3)$$

onde c_{ij} representa o custo de viajar de i para j , supondo $c_{ij} = c_{ji}$; $c_{ii} = +\infty$.

Considerem-se n pontos, sendo c_{ij} o custo do percurso de i para j , x_{ij} tomará o valor 1 se o trajecto i, j for válido, e 0 caso contrário.

Os problemas de designação determinam uma matriz $X=(x_{ij})$ de variáveis de decisão com distância mínima; porém, não asseguram a formação de uma rota, resultando, frequentemente, em sub-rotas.

No entanto, ao associar-se ao problema um problema de atribuição, pode obter-se um limite inferior para o custo da solução do problema.

3.3.2 Métodos exactos

Os métodos denominados de exactos são aqueles que encontram a melhor solução para o problema (se esta existir), ou seja, a solução óptima. Esta solução deve satisfazer de forma óptima a função objectivo correspondente ao problema em questão, respeitando todos os parâmetros que se aplicam à resolução do mesmo (Andrade et al., 2004).

Segundo Cormen et al. (1999), um problema pode ou não possuir um algoritmo exacto para encontrar a sua solução. Existindo esse algoritmo, o mesmo pode não encontrar um óptimo em tempo satisfatório, ou seja, o algoritmo pode levar décadas para encontrar a solução desejada. Quando isso acontece, diz-se que o mesmo é inviável para o problema em estudo (Andrade et al., 2004).

A viabilidade do método escolhido é também uma questão que deve ser analisada na escolha da forma exacta de resolução dos problemas. Como se pode observar, nem sempre um método exacto consiste na melhor forma de abordagem para um problema. Aprofundando a questão de viabilidade, podem ser inviáveis tanto os algoritmos da classe P como algoritmos da classe NP. Tomando por exemplo um algoritmo exacto com ordem $O(n^{100})$, mesmo pertencendo à classe P, este algoritmo pode não ser viável computacionalmente para, por exemplo, $n = 10000$ (Andrade et. Al., 2004).

Especificamente, os problemas classificados como polinomiais (pertencentes à classe de problemas P) são, na sua maioria, viáveis para aplicação de algoritmos exactos.

Segue-se a descrição de alguns métodos exactos que podem ser usados na resolução do problema do caixeiro viajante.

3.3.2.1 Algoritmo Dijkstra

Generalidades

O algoritmo de Dijkstra (Dijkstra, 1959 e Cormen et al., 2001), identifica, a partir de um vértice de um grafo, o custo mínimo entre esse vértice e todos os outros desse

grafo. No início, o conjunto S contém somente esse vértice, chamado origem. A cada passo, selecciona-se dum conjunto de vértices sobranes, o que se situa mais próximo da origem. De seguida, para cada vértice sobranes, actualiza-se a sua distância em relação à origem. Se a distância do novo vértice ficar menor, a nova distância é memorizada.

Supondo que o grafo é representado por uma matriz de adjacência, onde o valor é infinito se não existir aresta entre dois vértices. Admitindo também que os vértices do grafo são enumerados de 1 até n , então, o conjunto de vértices é $N = \{1, 2, \dots, n\}$. O vector $D[2, \dots, n]$ contém a distância que separa todo vértice do vértice 1 (o vértice do grafo, o vértice 1, é escolhido arbitrariamente).

Procedimentos

Para aplicação de qualquer método exacto ou heurístico, devem-se conhecer as distâncias (custos) entre todos os pares de nós. Estas distâncias podem ser obtidas através do algoritmo de Floyd (Christofides, 1975). Este algoritmo fornece as distâncias mínimas entre dois quaisquer nós de uma rede.

Dado um grafo não dirigido $G = (V, E)$, a distância de um vértice $v_i \in V$ a um vértice $v_j \in V$ é definido como o comprimento do mais curto caminho entre v_i e v_j . Assim, tem-se:

$$d(v_i, v_j) = \begin{cases} 0, & \text{se } i = j, \\ \infty, & \text{se não existe } \mu(v_i, v_j) \\ l(\mu(v_i, v_j)), & \text{caso contrário.} \end{cases} \quad (3.4)$$

Com: $l(\mu)$ comprimento do caminho μ .

No algoritmo, considera-se um vértice v_i como fechado quando é obtido um caminho $\mu(v_o, v_i)$, de menor custo, e aberto, no caso contrário.

Passo 1 - Atribui-se um custo inicial a $d[v]$, $d[v_i] = w(v_o, v_i)$, $i = 1, \dots, n$.

Passo 2 - Fecha-se o vértice ainda aberto que tiver o menor custo.

Passo 3 - Compara-se o custo de cada vértice aberto com a soma do custo do último vértice fechado, com o custo do arco que une o último vértice fechado e cada um dos vértices abertos. Se o custo que inclui o último vértice fechado for menor que o actual, faz-se a actualização.

Passo 4 - Executam-se os passos 2 e 3, n vezes (sendo n o número de vértices).

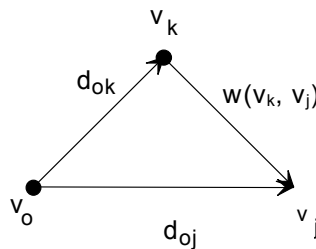


Figura 3.1 - Passo na intersecção k .

$$d(v_o, v_j) = \min[d(v_o, v_j), d(v_o, v_k) + w(v_k, v_j)] \quad (3.5)$$

Se $d(v_o, v_k) + w(v_k, v_j) < d(v_o, v_j)$, então v_k participa do menor caminho entre v_o e v_j .

Eficiência

A eficiência do algoritmo determina-se recorrendo ao conceito de caminho especial mais curto. Seja v_1 a origem, para qualquer vértice v_i , que não faz parte do conjunto S . O caminho especial mais curto é o caminho mais curto para alcançar v_i a partir de v_1 e que passa somente por vértices do conjunto S .

A determinação da eficiência do algoritmo baseia-se na prova do seguinte:

- 1- Se o vértice i pertence ao conjunto S , então $D[i]$ contém o comprimento do caminho mais curto entre a origem e i ;
- 2- Se o vértice i não pertence ao conjunto S , então $D[i]$ não contém o comprimento do caminho especial mais curto entre a origem e i ;

- 3- A prova faz-se por indução. Inicialmente, o vector D contém os valores das arestas que ligam cada vértice ao vértice 1 (valores infinitos se não tem aresta). Nenhum dos vértices, com a excepção do vértice 1, faz parte de S . Assim, a tarefa de conferir que as duas propriedades são verdadeiras é, nesse caso, muito simples e constitui a base da indução. A hipótese de indução supõe que as duas condições são verdadeiras imediatamente antes de acrescentar um novo vértice v no conjunto S . De seguida analisa-se o que acontece quando se acrescenta v ao conjunto S .
- 4- Em cada vértice do conjunto S , antes de acrescentar v , a primeira propriedade continua verdadeira. A hipótese de indução diz que $D[v]$ é a distância do caminho especial mais curto até v . Ao conseguir-se provar que o caminho mais curto até v passa somente por vértices do conjunto S , tem-se a prova que $D[v]$ é o valor do caminho mais curto e que, acrescentando v no conjunto S , respeita-se a primeira propriedade. Supõe-se que existe um outro caminho mais curto até v . Considera-se x , o primeiro vértice desse caminho alternativo que não pertence a S . A Figura 3.2 ilustra tal procedimento, onde a vermelho é indicado o caminho mais curto, e a preto o caminho especial mais curto.

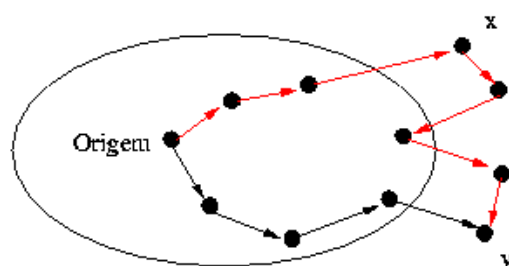


Figura 3.2 - Caminhos mais curtos.

- 5- Nesse caso, supondo que o grafo não contém nenhum peso negativo, a distância até v , passando por x , é maior ou igual à distância até x . Como esse caminho começa com um caminho especial até x , e que, por indução, o comprimento $D[x]$ dessa parte do caminho é mínima, sabe-se que a distância total até v passando por x é maior ou igual a $D[x]$. Como v foi seleccionado antes de x , deve-se ter também $D[x] \geq D[v]$. Isso implica que o caminho a

vermelho, supostamente mais curto, deve possuir um comprimento maior ou igual ao comprimento do caminho especial indicado a preto, o que contradiz a hipótese. Conclui-se então que o caminho especial até v é o mais curto.

- 6- Considera-se agora o que acontece com os outros vértices diferentes de v que ficam fora do conjunto S . Seja w tal vértice. Quando v é acrescentado no conjunto S , tem duas possibilidades no que diz respeito ao caminho especial mais curto até w : ou ele fica igual; ou passa agora por v . Considera-se então o segundo caso.
- 7- Parece que ainda existem duas possibilidades: ou o vértice v é o último no caminho até w , ou não é. Demonstra-se que o primeiro caso nunca acontece. Admite-se então que o último vértice antes de w é o vértice x , diferente de v , tal como ilustrado na Figura 3.3:

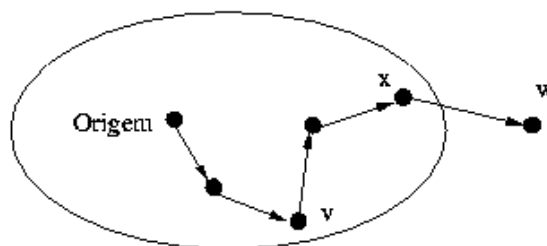


Figura 3.3 - O último vértice antes de w é x .

- 8- Esse caminho não pode ser mais curto pois já existe um caminho mais curto (ou igual) até x que não inclui v (isso porque x já estava em S antes da adição de v e, por indução, já se tinha um caminho mais curto). Então, o caminho especial mais curto é o menor entre o que passa por v antes de chegar a w e o que se tinha antes de acrescentar v . Esta é a prova final do algoritmo.

Os passos iniciais do algoritmo exigem um tempo de execução da ordem (n). O ciclo executa-se $n-2$ vezes e a cada iteração todos os vértices, não actualmente no conjunto S , são visitados. Na primeira iteração, percorrem-se $n-1$ vértices, na segunda, $n-2$ vértices, e assim por diante. Pode-se então deduzir que o tempo de execução é da ordem de n^2 .

Considere-se agora um grafo esparso. Mais especificamente, um grafo com a arestas e n vértices, onde $a \ll n$. Pode-se melhorar o desempenho do algoritmo utilizando uma estrutura de adjacência, e supondo que, para cada vértice, tem-se um campo que indica se ele pertence ao conjunto C , e que cada elemento da lista de adjacência associada a um vértice v indica não somente o vértice w ligado mas também o peso $d(v,w)$ de v para w .

Neste caso, o ciclo mais interno não vai percorrer todos os vértices do grafo mas somente os que são adjacentes ao último vértice acrescentado. No total, o teste interno será executado a vezes, o que é proporcional a n , no caso de um grafo esparso. Mas mesmo se existir uma melhoria na prática, o algoritmo ainda é da $O(n^2)$ porque devem-se ainda percorrer todos os elementos do conjunto C para retirar o vértice que apresenta o menor valor.

É possível melhorar ainda mais a eficiência do algoritmo usando a estrutura de adjacência e um ciclo que contenha um nodo para cada vértice v do conjunto C , ordenados por ordem crescente pelo valor $D[v]$. Considere-se agora o ciclo principal. Para retirar o vértice v do conjunto C que minimiza $D[v]$, simplesmente remove-se o vértice que está na raiz do ciclo e ajusta-se o ciclo. Sabe-se que isso exige um tempo em $O(\log n)$. No que diz respeito ao ciclo mais interno, devem considerar-se todos os vértices x adjacentes a v pertencentes ao conjunto C .

Neste caso, deve-se ajustar o valor $D[x]$. O ajuste necessário ao ciclo exige um tempo da ordem $O(\log n)$. Em suma, é preciso retirar por $n-2$ vezes um vértice do conjunto C (exigindo uma actualização do ciclo a cada vez), e rever o valor $D[v]$ no máximo igual a a vezes, o que exige uma actualização do ciclo. Portanto, com esta implementação, o tempo de execução será da ordem $(a+n)\log n$. Se o grafo é conexo, tem-se $a \geq n-1$ e então o tempo de execução é da ordem $a \log n$.

Exemplo

Seja o grafo direccionado $G = (V, E)$

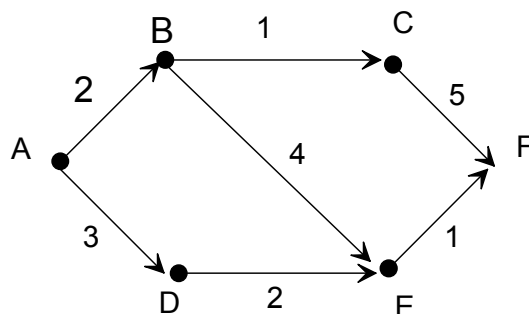


Figura 3.4 - Grafo direccionado.

Tem-se que:

Tabela 3.1 - Matriz de custos.

	A	B	C	D	E	F
A	0	2	∞	3	∞	∞
B	∞	0	1	∞	4	∞
C	∞	∞	0	∞	∞	5
D	∞	∞	∞	0	2	∞
E	∞	∞	∞	∞	0	1
F	∞	∞	∞	∞	∞	0

Aplicando o algoritmo descrito anteriormente, obtêm-se os resultados apresentados na Tabela 3.2.

Tabela 3.2 - Resultados da aplicação do algoritmo de Dijkstra.

Passos	Vértice fechado	Distância a partir do vértice raiz A					Vértices não fechados
		B	C	D	E	F	
0	A (raiz)	2	∞	3	∞	∞	B, C, D, E, F
1	B	*	3	3	6	∞	C, D, E, F
2	C		*	3	6	8	D, E, F
3	D			*	5	8	E, F
4	E				*	6	F
5	F					*	$\emptyset$

A Tabela 3.3 apresenta os melhores caminhos encontrados que correspondem às distâncias menores.

Tabela 3.3 - Melhores caminhos encontrados.

A a B:	A B	$l(\mu(A, B)) = 2$
A a C:	A B C	$l(\mu(A, C)) = 3$
A a D:	A D	$l(\mu(A, D)) = 3$
A a E:	A D E	$l(\mu(A, E)) = 5$
A a F:	A D E F	$l(\mu(A, F)) = 6$

3.3.2.2 Algoritmo de Floyd

Generalidades

Seja $G=(V,E)$ um grafo direccionado. Supõe-se que os vértices de G são enumerados de 1 até n . O grafo representa-se por uma matriz de adjacência D onde $D[i,j] = 0$ quando $i=j$, $D[i,j] \geq 0$ para $i \neq j$, e $D[i,j] = \infty$ quando não existe nenhuma aresta entre i e j .

O princípio do algoritmo que se vai apresentar de seguida é o seguinte (Cormen et al., 1990 e Hu, 1969): O algoritmo efectua n iterações. Na iteração k do algoritmo obtém-se uma matriz D cujos elementos na posição $D[i,j]$ representam o caminho mais curto de i até j , entre todos os que passam somente por vértices do conjunto $C = \{1,2,\dots,k\}$, isto é, um caminho que pode conter somente os vértices i e j e qualquer outro do conjunto $C = \{1,2,\dots,k\}$. Quando se acrescenta o vértice $k+1$, tem-se que actualizar todas as distâncias para ver se agora não se tem um caminho mais curto passando pelo vértice k . O valor na posição $D[i,j]$ muda somente se o valor $D[i,k] + D[k,j]$ for menor. Neste caso, coloca-se em $D[i,j]$ o valor $D[i,k] + D[k,j]$. Este procedimento é repetido até $k = n$, obtendo-se a matriz com os valores dos caminhos mais curtos entre qualquer par de vértices.

Procedimentos

Considere-se a matriz de custos definida no algoritmo anterior.

Passo 1 - Faça-se $k=0$.

Passo 2 - $k=k+1$.

Passo 3 - Para todo o $i \neq k$ tal que $c_{ij} \neq \infty$ e todo $j \neq k$ tal que $c_{kj} \neq \infty$. Tome-se c_{ij} como o mínimo entre c_{ij} e $c_{ik} + c_{kj}$.

Passo 4a - Se $k=n$ parar. A solução óptima foi obtida e $[c_{ij}]$ fornece os custos mínimos para cada par de nós.

Passo 4b - Se $k < n$, voltar ao passo 2.

O algoritmo de Floyd pode ser complementado de modo a apresentar, além da matriz de custos mínimos, os caminhos para os quais estes custos foram obtidos. Para isto Hu (Hu, 1969) sugere o armazenamento de uma segunda matriz $P=[p_{ij}]$, onde p_{ij} é o nó que precede o vértice v_j , no caminho mínimo entre os vértices v_i e v_j .

Eficiência

A prova do funcionamento correcto do algoritmo é feita também por indução. No início $D[i,j]$ contém o valor da aresta entre i e j se tal aresta existir, e ∞ no caso

contrário. Como inicialmente o conjunto C está vazio, não existe vértice intermediário entre i e j , e $D[i,j]$ apresenta o valor do caminho mais curto, supondo-se a hipótese verdadeira para a iteração k . Se o caminho entre i e j que passa por $k+1$ é mais curto, necessariamente os pedaços de i até k e de k até j são também os mais curtos. A matriz contém, nas posições $D[i,k]$ e $D[k,j]$, os valores desses caminhos mais curtos. A comparação da soma desses dois valores com o valor encontrado em $D[i,j]$ e a escolha do menor permitem obter, na etapa $k+1$, os caminhos mais curtos que passam pelos vértices $\{1,2,\dots,k+1\}$.

Por comparação com o algoritmo de Dijkstra, verifica-se que é da ordem $O(n^3)$. Note-se que este mesmo problema pode ser resolvido aplicando n vezes o algoritmo de Dijkstra, começando cada vez com um vértice diferente. Neste caso, o tempo de execução é também da ordem de $O(n^3)$. Mas tal não significa que ambos têm a mesma eficiência. O algoritmo de Floyd necessita de iniciar uma matriz de tamanho $n \times n$. O algoritmo de Dijkstra tem, dentro do primeiro ciclo, dois ciclos onde cada um exige um tempo de execução de ordem n . É conveniente lembrar que, com o uso de uma estrutura de ciclo, pode-se obter uma implementação do algoritmo de Dijkstra em que o tempo de execução é da ordem $O((a+n) \log n)$, onde a é o número de arestas. Aplicando n vezes o algoritmo, obtém-se um tempo de execução de $n \times O((a+n) \log n) = O((an + n^2) \log n)$. Se o grafo é esparso, pode ser mais adequado utilizar o algoritmo de Dijkstra. Se o número de arestas se aproxima do grafo completo, o algoritmo de Floyd será o mais indicado.

A descrição do algoritmo apresentado acima não permite identificar o caminho mais curto. Esse caminho obtém-se modificando o algoritmo da seguinte forma. Utiliza-se uma matriz P onde cada elemento $P[i,j]$ aponta para o último vértice (diferente de i) encontrado no caminho antes de chegar a j . A matriz P é actualizada cada vez que uma posição $D[i,j]$ é actualizada, colocando o último vértice k considerado. No início, toda a matriz P recebe o valor 0. No final, o caminho mais curto entre i e j , será obtido percorrendo os ponteiros a partir de $P[i,j]$ até chegar a uma entrada que tem valor 0.

Exemplo:

Apliquem-se agora as etapas da execução do algoritmo de Floyd com o grafo da Figura 3.5. Estas são representadas na Tabela 3.4.

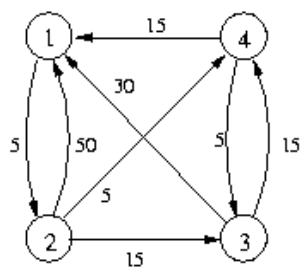


Figura 3.5 - Aplicação do algoritmo de Floyd a um grafo direccionado.

Tabela 3.4 - Exemplo de aplicação do algoritmo de Floyd.

	Iteração	Matriz D	Matriz P
Etapa 1:	Inicialização	0 5 ∞ ∞	0 0 0 0
		50 0 15 5	0 0 0 0
		30 ∞ 0 15	0 0 0 0
		15 ∞ 5 0	0 0 0 0
Etapa 2:	1	0 5 ∞ ∞	0 0 0 0
		50 0 15 5	0 0 0 0
		30 35 0 15	0 1 0 0
		15 20 5 0	0 1 0 0
Etapa 3:	2	0 5 20 10	0 0 2 2
		50 0 15 5	0 0 0 0
		30 35 0 15	0 1 0 0
		15 20 5 0	0 1 0 0
Etapa 4:	3	0 5 20 10	0 0 2 2
		45 0 15 5	3 0 0 0
		30 35 0 15	0 1 0 0
		15 20 5 0	0 1 0 0
Etapa 5:	4	0 5 15 10	0 0 4 2
		20 0 10 5	4 0 4 0
		30 35 0 15	0 1 0 0
		15 20 5 0	0 1 0 0

A matriz P obtida aplicando o algoritmo permite determinar que, por exemplo, o caminho mais curto de 1 até 3 passa pelo vértice 4. O valor 4 aparece na posição (1,3) da matriz. O conhecimento do caminho completo é determinado pela procura recursiva do caminho mais curto de 1 até 4 e de 4 até 3. Procurando novamente na matriz P , vê-se que na posição (4,3) há um 0, que indica que o caminho de 4 até 3 é directo. Na posição (1,4) da matriz aparece um 2, indicando que o caminho mais curto de 1 até 4 passa pelo vértice 2. Repetindo o processo, vê-se que os caminhos mais curtos de 1 até 2 e de 2 até 4 não contêm outros vértices intermediários. Portanto, o caminho mais curto entre 1 e 3 é a sequência dos vértices 1, 2, 4, 3.

3.3.2.3 Método Húngaro

Generalidades

O método Húngaro é um algoritmo de optimização que é aplicado na resolução de problemas de designação. A primeira versão, foi inventada e publicada por Harold Kuhn em 1955. Esta foi revista por James Munkres em 1957, passando-se a designar por algoritmo de designação de Munkres, ou algoritmo de Kuhn-Munkres. O algoritmo baseou-se nos primeiros trabalhos de outros dois matemáticos húngaros: Dénes König e Jenő Egerváry. O algoritmo constrói uma solução válida para o problema primal partindo duma solução não válida (que corresponde a uma solução válida do problema dual).

Procedimentos

O algoritmo modela um problema de designação como uma matriz de custos n por m , que corresponde à matriz de custos definida no algoritmo de Floyd (MacLaughlin, 1979 e Harold Kuhn, 1955). Por omissão, o algoritmo faz a minimização dos elementos da matriz; desta forma, em caso de se estar perante um problema de minimização dos custos, torna-se suficiente usar o método de eliminação de Gauss-Jordan para obtenção de zeros (pelo menos um zero por linha e por coluna). No caso de um problema de maximização, a matriz de custos deverá ser modificada de forma a que a minimização dos seus elementos se torne numa maximização do problema.

Passo 1 - Obtenha a matriz de custo pelo algoritmo de Floyd;

Passo 2 - Subtraia a cada linha ou a cada coluna o seu menor elemento por forma a gerar, no mínimo, um zero em cada linha e em cada coluna;

Passo 3 - Faça o máximo de atribuições às células zero ($\checkmark$ para atribuir e X para retirar):

- a) Em cada linha com apenas um zero, faça $\checkmark$ e X em todos os zeros daquela coluna;
- b) Em cada coluna com apenas um zero, faça $\checkmark$ e X em todos os zeros restantes daquela coluna e respectiva linha;
- c) Repita a) e b) até que todos os zeros da matriz estejam marcados, certificando-se que em cada linha e coluna não existe mais que um $\checkmark$.

Passo 4 - Se existir um $\checkmark$ em cada linha e coluna, então estas são atribuições mínimas. Marque as respectivas células no problema original obtendo o custo total. A solução óptima foi obtida.

Passo 5 - Se não existir um $\checkmark$ em cada linha e coluna, podem obter-se mais zeros da seguinte forma:

- a) Identifique todas as linhas sem atribuição;
- b) Identifique todas as colunas com X's nas linhas identificadas anteriormente;
- c) Identifique todas as linhas com $\checkmark$ nas colunas marcadas;
- d) Repita b) e c) até à exaustão.

Passo 6 - Alinhe todas as linhas não identificadas e colunas identificadas.

Passo 7 - Na nova matriz proceda da seguinte forma:

- a) Encontre o mínimo valor não alinhado (das linhas não mexidas no ponto 6);
- b) Subtraia este valor a todas as linhas identificadas no ponto 5;
- c) Adicione este valor a todas as colunas identificadas no ponto 5.

Passo 8 - Faça novamente as atribuições (ponto 3). Caso não exista um $\surd$ em cada linha e em cada coluna, devem-se repetir os passos 5, 6 e 2.

Nota: Se, no passo 3, as linhas e colunas com um único zero já terminaram e tem-se somente linhas e colunas com dois ou mais zeros, significa que existem atribuições alternativas ou seja percursos diferentes com custo mínimo.

Exemplo

Seja a Tabela 3.5 com a matriz de custos (passo 1).

Tabela 3.5 - Tabela de custos do exemplo em causa.

c_{ij}	1	2	3
1	900	750	750
2	350	850	550
3	1250	950	900

Subtrair 750 nas entradas da primeira linha, 350 na segunda e 900 na terceira (passo 2):

Tabela 3.6 - Tabela obtida após passo 2.

c_{ij}	1	2	3
1	150	0	0
2	0	500	200
3	350	50	0

Tabela 3.7 - Tabela obtida após passo 3.

c_{ij}	1	2	3
1	150	$\surd$	X
2	$\surd$	500	200
3	350	50	$\surd$

Como pode ver-se, existe $\surd$ em cada linha e cada coluna; logo está-se perante a solução óptima.

3.3.2.4 Programação linear inteira

Generalidades

A programação linear (PL) (Ramalhete et al., 1995 e Wosley, 1998) é uma importante área de investigação no campo da optimização. Muitos problemas práticos em investigação operacional podem ser expressos como problemas de programação linear. Certos casos especiais de programação linear, tais como problemas de fluxo de redes, são considerados suficientemente importantes para que se tenha gerado muita pesquisa em algoritmos especializados para obter as soluções. Vários algoritmos para outros tipos de problemas de optimização funcionam resolvendo problemas de PL como sub-problemas. Historicamente, as ideias da programação linear inspiraram muitos dos conceitos centrais de teoria da optimização, tais como dualidade, decomposição e a importância da convexidade e suas generalizações.

Procedimentos

Seja o problema da Secção 3.3.1, definido pelas equações:

$$\text{Min } z = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \text{ sujeito a } \sum_{j=1}^n x_{ij} = 1, i=1, 2, \dots, n. \quad (3.6)$$

$$\sum_{i=1}^n x_{ij} = 1, j=1, 2, \dots, n. \quad (3.7)$$

Na eliminação de sub-rotas devem estabelecer-se restrições em x_{ij} de acordo com a definição seguinte:

- 1- Todo o sub-conjunto de nós próprios P deve estar conectado aos outros nós do percurso na solução: $\sum_{i \in P} \sum_{j \in P} x_{ij} \geq 1$, sendo P todo o sub-conjunto próprio não vazio de N .

- 2- Seja R todo o sub-conjunto não vazio de $\{2, 3, \dots, n\}$. Um ciclo em R contém $|R|$ arcos; logo, a condição que se segue elimina a hipótese de que o conjunto

de arcos seleccionados contenha um ciclo: $\sum_{i \in R} \sum_{j \in R} x_{ij} \leq |R| - 1$.

Como se pode depreender, no caso de n pontos existem 2^n restrições só para eliminar a existência de sub-rotas, tornando-se o problema de resolução mais difícil.

Esta situação pode ser ultrapassada reformulando o problema da seguinte forma:

$$\text{Min } z = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \text{ sujeito a } \sum_{j=1}^n x_{ij} = 1, i=1, 2, \dots, n. \quad (3.8)$$

$$\sum_{i=1}^n x_{ij} = 1, j=1, 2, \dots, n. \quad (3.9)$$

Os passos subsequentes de resolução do problema são:

- 1- Obrigar que cada nó seja apenas um ponto de chegada e apenas um ponto de partida, com excepção do primeiro que será apenas um ponto de partida:

$$\sum_j x_{1j} = 1 \text{ e } \sum_j x_{ij} + \sum_k x_{ki} = 2.$$

- 2- Obrigar à existência de n arcos no percurso obtido: $\sum_{i=1}^n \sum_{j=1}^n x_{ij} = n$.

Estas restrições tornam o problema de difícil resolução quando o número de pontos n cresce substancialmente.

3.3.2.5 Programação dinâmica

Generalidades

A programação dinâmica é um método para a construção de algoritmos de resolução de problemas computacionais, em especial os de optimização combinatória (Bellman, 1957). É aplicável a problemas nos quais a solução óptima pode ser determinada a partir de uma solução óptima previamente calculada e memorizada de forma a evitar um novo cálculo de outros sub-problemas que, sobrepostos, compõem o problema original.

O algoritmo de Dijkstra permite identificar todos os caminhos mais curtos a partir de um único vértice (origem). Utilizando a programação dinâmica, pode obter-se o caminho mais curto entre qualquer par de vértices. Lembra-se que a programação dinâmica consiste essencialmente em produzir soluções intermediárias que são utilizadas incrementalmente para obter a solução final.

Procedimentos

Seja um grafo direccionado representado por uma matriz de adjacência $L[1, \dots, n; 1, \dots, n]$. Calcule-se em cada etapa do algoritmo uma matriz D onde cada elemento $D[i, j]$ representa o valor do caminho mais curto de k arestas entre i e j . Na etapa seguinte, os valores das matrizes D e L são utilizados no cálculo da nova matriz D que contém os valores para $k+1$ arestas. Na obtenção do novo valor $D[i, j]$ considera-se o valor $D[i, u] + L[u, j]$ para todo o vértice u e escolhe-se o menor valor. Isto é, consideram-se todos os caminhos de k vértices a partir de i , acrescentando-se, para cada um desses caminhos, a aresta que falta para alcançar j e seleccionar o caminho mais curto.

Exemplo

Considere-se o grafo ilustrado na Figura 3.5; as etapas da execução do algoritmo para o referido grafo são apresentadas na Tabela 3.8.

Tabela 3.8 - Resultados da aplicação da programação dinâmica ao grafo da Figura 3.5.

	Iteração	Matriz D
Etapa 1:	Inicialização	0 5 ∞ ∞
		50 0 15 5
		30 ∞ 0 15
		15 ∞ 5 0
Etapa 2:	1	0 5 20 10
		20 0 10 5
		30 35 0 15
		15 20 5 0
Etapa 3:	2	0 5 15 10
		20 0 10 5
		30 35 0 15
		15 20 5 0

3.3.3 Métodos heurísticos

Generalidades

A solução de problemas de elevado nível de complexidade computacional tem sido um desafio constante para os investigadores de diversas áreas, particularmente em Optimização, Investigação Operacional, Ciência da Computação, Matemática e Engenharias, onde aparecem frequentemente problemas altamente combinatórios.

Os métodos tradicionais de optimização exacta caracterizam-se pela rigidez de seus modelos matemáticos representados através de seus teoremas, dificultando a representação de situações reais cada vez mais complexas e dinâmicas (Reeves, 1993). O problema desta falta de flexibilidade foi um pouco reduzido a partir do momento em que se passou a associar técnicas de Optimização com ferramentas de Inteligência Artificial, mais especificamente, com as ferramentas de busca heurística. De facto, os algoritmos heurísticos, ou simplesmente heurísticas, caracterizam-se pela sua flexibilidade e têm como objectivo encontrar soluções de boa qualidade num tempo computacional tolerável.

Contudo, as heurísticas também apresentam limitações, sendo uma das principais o facto de não conseguirem superar os óptimos locais. Além disso, a falta de uma base teórica dos métodos heurísticos produz algoritmos muito especializados. Ou seja, apesar da sua flexibilidade em incorporar novas situações, o seu desempenho pode oscilar muito com modificações, mesmo que pequenas.

As heurísticas foram consideradas durante muito tempo modelos cognitivos por excelência; constituem-se como regras baseadas na experiência e no planeamento, substituindo as anteriores baseadas na procura algorítmica que chega às soluções correctas depois de ter combinado o problema com todas as soluções possíveis.

Os métodos heurísticos procuram um grau tão grande quanto possível de uma acção para cada situação. Assim, englobam estratégias, procedimentos, métodos de aproximação tentativa/erro, sempre na procura da melhor forma de chegar a um determinado fim. Os processos heurísticos exigem muitas vezes menos tempo que os processos algorítmicos, aproximando-se mais da forma como o ser humano raciocina e chega às resoluções dos problemas garantindo soluções eficientes.

Os métodos heurísticos dividem-se em duas grandes classes: métodos de construção e métodos de melhoria de rotas.

3.3.3.1 Métodos de construção de rotas

Destes métodos podem destacar-se os métodos de Clarke and Wright, inserção e obtenção de sementes. (Bodin et al., 1983):

3.3.3.1.1 Método de Clarke and Wright

Generalidades

O método de Clark and Wright é um dos mais tradicionais (Clark e Wright, 1964; Golden et al., 1984). A simplicidade da sua concepção e o critério da poupança utilizado nos percursos, fizeram com que os autores tivessem diversos seguidores.

O algoritmo de Clark and Wright é um procedimento iterativo, que permite uma rápida selecção de uma rota óptima ou quase óptima. Além disso, baseia-se na noção de poupança, que pode ser definida como o custo da combinação, ou união, de duas sub-rotas existentes. Por isso, para a execução, o algoritmo pressupõe o conhecimento da menor rota entre dois pontos.

Procedimento

O algoritmo consiste na construção de várias rotas simultaneamente. Em cada iteração efectuem-se trocas nas rotas por forma a obter rotas melhores. O processo inicia-se com um número de rotas igual ao número de nós. Supõe-se a existência de n veículos para visitar cada nó individualmente. Em seguida, verifica-se a hipótese de trocar os dois veículos que visitam os nós i e j apenas por um veículo. No passo seguinte, as rotas vão sendo construídas de acordo com a maior poupança $s(i,j)$ do seguinte modo: $s(i,j) = (2c(0,i) + 2c(0,j)) - (c(0,i) + 2c(0,j) + c(i,j)) = c(0,i) + c(0,j) - c(i,j)$.

A descrição mais pormenorizada do algoritmo é apresentada de seguida:

Etapa 1 Iniciar as n rotas ligando o nó 0 (depósito) a cada ponto: (0- i -0), para $i=1, 2, 3, \dots, n$;

Etapa 2 Calcular as poupanças entre os pontos i, j da forma descrita anteriormente: $s(i,j) = c(0,i) + c(0,j) - c(i,j)$;

Etapa 3 Ordenar as poupanças de forma decrescente;

Etapa 4 Começando no topo, deve-se percorrer a lista das poupanças por forma a construir sub-rotas:

- a) i, j devem ser nós extremos (3 e 2 são nós extremos na rota (0-2-1-4-3-0));
- b) i, j não devem ser atribuídos à mesma sub-rota, ou seja se i faz parte da sub-rota de j , as poupanças de i para j não devem ser consideradas;
- c) A carga atribuída ao veículo na sub-rota i mais a carga atribuída pela sub-rota j , não devem ultrapassar a capacidade do veículo;

Etapa 5 Se as condições do ponto 4 forem satisfeitas devem-se unir as sub-rotas;

Etapa 6 Parar se não existir $s(i,j)$ que satisfaça as condições da etapa 4; caso contrário, voltar à etapa 4.

3.3.3.1.2 Método da inserção

Generalidades

Os algoritmos heurísticos de inserção apareceram com Rosenkrantz, Sterns e Lewis, em 1977. O procedimento consiste na seguinte metodologia (Golden et al., 1980, Bodin et al., 1983 e Rosenkrantz, 1977): aquando da inserção de um novo vértice, começa-se por procurar determinar qual o vértice que será adicionado ao caminho e, em seguida, qual o local mais apropriado para fazer essa inserção.

Estes algoritmos dividem-se em vários tipos conforme a inserção seja económica, mais próxima, arbitrária e mais distante.

Procedimentos

Estes algoritmos são descritos através das etapas seguintes:

Inserção mais económica:

Etapa 1 Escolher um nó i e criar uma sub-rota;

Etapa 2 Encontrar o nó k tal que c_{ik} seja mínimo. Formar a sub-rota $i-k-i$;

Etapa 3 Encontrar o arco (i,j) na sub-rota que minimize $c_{ik} + c_{kj} - c_{ij}$. Inserir k entre i e j ;

Etapa 4 Voltar a 3 até construir um circuito Hamiltoniano.

Inserção mais próxima:

Etapa 1 Escolher um nó i e criar uma sub-rota;

Etapa 2 Encontrar o nó k tal que c_{ik} seja mínimo. Formar a sub-rota $i-k-i$;

Etapa 3 Em relação à sub-rota corrente, encontre um vértice k que ainda não esteja nele e seja o mais próximo;

Etapa 4 Encontrar o arco (i,j) na sub-rota que minimize $c_{ik} + c_{kj} - c_{ij}$. Inserir k entre i e j ;

Etapa 5 Voltar a 3 até construir um circuito Hamiltoniano.

Inserção arbitrária:

Etapa 1 Escolher um nó i e criar uma sub-rota;

Etapa 2 Encontrar o nó k tal que c_{ik} seja mínimo. Formar a sub-rota $i-k-i$;

Etapa 3 Em relação à sub-rota corrente, encontrar um vértice k qualquer que não esteja nele e adicione-o à sub-rota;

Etapa 4 Encontrar o arco (i,j) na sub-rota que minimize $c_{ik} + c_{kj} - c_{ij}$. Inserir k entre i e j ;

Etapa 5 Voltar a 3 até construir um circuito Hamiltoniano.

Inserção mais distante:

Etapa 1 Escolher um nó i e criar uma sub-rota;

- Etapa 2 Encontrar o nó k tal que c_{ik} seja mínimo. Formar a sub-rota $i-k-i$;
- Etapa 3 Em relação à sub-rota corrente, encontre um vértice k que ainda não esteja nele e seja o mais distante;
- Etapa 4 Encontrar o arco (i,j) na sub-rota que minimize $c_{ik} + c_{kj} - c_{ij}$. Inserir k entre i e j ;
- Etapa 5 Voltar a 3 até construir um circuito Hamiltoniano.

3.3.3.1.3 Obtenção de sementes

Generalidades

Este algoritmo selecciona as melhores posições para inserir sementes (pontos artificiais) aos quais são agrupados pontos com designação temporária que, futuramente, são afectados a cada veículo (Paraíba et al. 1990).

Procedimentos

Para iniciar a formação de sementes, é necessário determinar o número m de sementes, uma para cada veículo. Entre os diversos procedimentos existentes na literatura, descreve-se em seguida o apresentado por Paraíba em 1990:

- Etapa 1 Escolher os m pontos sementes que permitem que as m -rotas - R_i sejam iniciadas;
- Etapa 2 Fazer $k=1$, obter i_k tal que $c(0, i_k) = \text{Max}\{c(0, i_k)\}$, para todo o i , onde 0 é o depósito central, ou seja, encontrar o ponto mais distante do depósito central, que constituirá a primeira semente;
- Etapa 3 Fazer $k=k+1$ e obter as restantes sementes. Determinar i^* , mesmo que não pertença à rota, sendo o máximo para $k=2, 3, \dots, m$ de:
- $$c(i, i_{k-1}) + c(i, i_{k-2}) + \dots + c(i, i_2) + c(i, i_1).$$

Esta escolha evita que as m rotas sejam inicializadas por sementes próximas, ou seja, evita que as rotas sejam inicialmente concorrentes aos mesmos pontos.

3.3.3.2 Métodos de melhoria de rota

Nesta secção, apresentam-se algoritmos locais para melhoria de rotas. Estes algoritmos baseiam-se em operações (trocas ou movimentos) que podem ser usadas com o objectivo de converter uma rota noutra (Lin e Kernighan, 1973). Estabelecida uma rota possível, o algoritmo executa repetidas operações sobre a rota de forma a reduzir o comprimento da rota inicial, até que não seja possível efectuar mais trocas e obter uma rota melhor, tendo-se então obtido uma rota óptima. Alternativamente, pode encarar-se este algoritmo como um processo de busca do melhor vizinho, onde cada rota tenha um vizinho associado a rotas adjacentes, isto é, aquelas que podem ser alcançadas num único movimento, e se movem continuamente para um vizinho melhor até que nenhum vizinho melhor exista.

Laporte (1992) aponta ainda métodos de melhoria baseados em meta-heurísticas do tipo recozimento simulado, algoritmos genéticos e busca tabu, que também se baseiam em métodos de melhoria do tipo 2-opt ou 3-opt; contudo, procurando ir além da solução final obtida por alguns desses métodos (que pode corresponder a um ponto de óptimo local). No recozimento simulado, utiliza-se um controlo de “temperatura” para encontrar possibilidades de soluções melhores, partindo-se de soluções piores. Nos algoritmos genéticos, encontra-se a solução do problema utilizando um mecanismo de evolução a partir de uma solução aceitável para um nível superior de optimização. Já na busca tabu, seleccionam-se movimentos que são temporariamente proibidos (“tabu”), mesmo que melhorem a solução, por forma a encontrar soluções piores no princípio e que venham a resultar em soluções melhores no final.

3.3.3.2.1 k-opt

Os métodos de melhoria mais utilizados são do tipo k -opt, conforme proposto por Lin e Kernighan (1973), nos quais k arcos são removidos de um percurso e substituídos por outros k arcos, com a finalidade de diminuir a distância total percorrida.

Conforme Laporte (1992), quanto maior for o valor de k , melhor será a precisão do método; porém, maior será o esforço computacional. Na prática, são considerados os

métodos 2-opt e 3-opt, isto é, k assume os valores 2 ou 3, conforme apresentado nas Figuras 3.6 e 3.7, respectivamente.

De forma resumida, o método 2-opt, proposto pela primeira vez por Croes (1958), baseia-se na experimentação de trocas possíveis entre pares de arcos, refazendo as ligações quando houver uma melhoria no percurso. O processo termina quando não for possível realizar mais nenhuma troca que resulte em melhoria. Este procedimento possui ordem de complexidade $O(n^2)$. Como pode ocorrer inversão de sentido em parte do percurso, tal como mostrado na Figura 3.6, pressupõe-se simetria de distâncias.

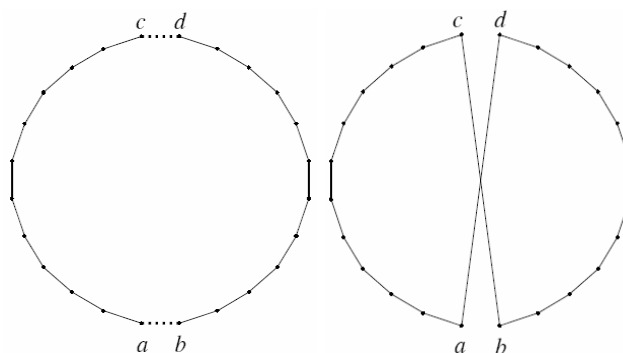


Figura 3.6 - Troca 2-Opt.

Esquerda: rota original; Direita: rota resultante da troca.

No caso do 3-opt (Bock, 1958 e Lin, 1965), são considerados três arcos em vez de dois a fim de se avaliarem as alterações nas ligações entre os nós, o que resulta em sete possíveis combinações, conforme pode ser visto na Figura 3.7.

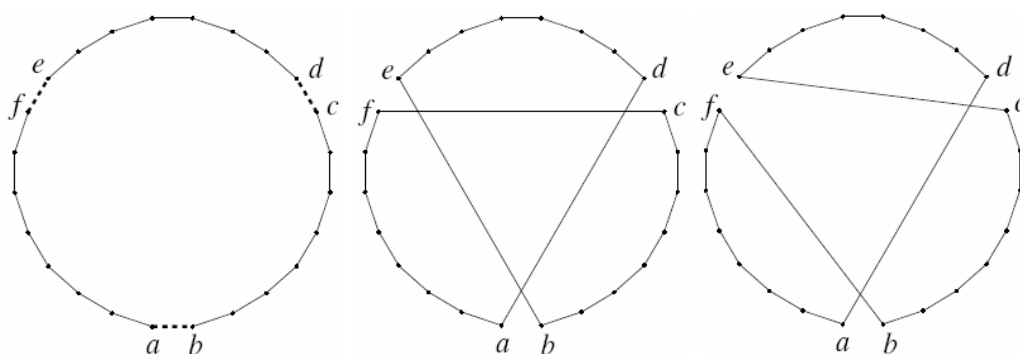


Figura 3.7 - Troca 3-Opt.

Esquerda: rota original; Direita: rotas resultantes das trocas.

Heslgaun (2000) propôs uma implementação eficiente da heurística proposta por Lin e Kernighan (1973), que considera um procedimento de melhoria do tipo k -opt, porém com o valor de k variável ao longo do processo e não fixo, tal como nas implementações clássicas descritas anteriormente. Um dos aspectos apontados pelo autor em relação ao método proposto é a dificuldade de implementação computacional. As modificações sugeridas pelo autor, no sentido de restringir e direccionar a busca de arcos a serem trocados, não tornaram a implementação mais simples, permitindo, no entanto, resolver problemas maiores com resultados significativos.

3.3.3.2 Recozimento simulado

O recozimento simulado (RS) é um meta-algoritmo probabilístico genérico para procura de óptimos globais. Foi desenvolvido por Kirkpatrick et al. em 1983. Surge como que uma generalização do método de Monte Carlo (descrito a seguir).

Esta meta heurística é uma metáfora de um processo térmico, dito de recozimento, utilizado em metalurgia para obtenção de estados de baixa energia em materiais. O processo compreende duas etapas: na primeira, procede-se ao aquecimento do material até uma temperatura judiciosamente escolhida; na segunda etapa, promove-se, de uma forma controlada, o seu arrefecimento, até à temperatura ambiente.

Este arrefecimento de forma controlada permite que se obtenham estruturas de equilíbrio e, conseqüentemente, de menor energia, daí resultando uma redução das tensões internas do material (Kirkpatrick et al., 1983).

De forma análoga, o algoritmo de recozimento simulado substitui a solução actual por uma solução próxima (i.e., na sua vizinhança no espaço de soluções), escolhida de acordo com uma função objectivo e com uma variável T (dita temperatura, por analogia). Quanto maior for T , maior a componente aleatória que será incluída na próxima solução escolhida. À medida que o algoritmo progride, o valor de T sofre um decréscimo, começando o algoritmo a convergir para uma solução óptima, necessariamente local.

Uma das principais vantagens deste algoritmo é a de permitir testar soluções mais distantes da solução actual e dar mais independência relativamente ao ponto inicial da pesquisa.

Esta técnica começa a busca a partir de uma solução inicial qualquer e o procedimento principal consiste num ciclo que gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s . Para cada geração de um novo vizinho s' de s , é testada a variação Δ do valor da função objectivo, isto é $\Delta = f(s') - f(s)$, podendo ocorrer as seguintes situações:

- $\Delta < 0$: Há uma diminuição de energia, a qual implica que a nova solução seja melhor que a anterior. O método aceita a solução e s' passa a ser a nova solução corrente.
- $\Delta = 0$: Caso de estabilidade, não havendo diminuição de energia. Na verdade, corresponde a uma situação pouco provável de acontecer na prática. A aceitação da solução é, portanto, indiferente.
- $\Delta > 0$: Houve um aumento do estado de energia. A aceitação deste tipo de solução é mais provável a altas temperaturas e bastante improvável a temperaturas mais baixas. Para reproduzir essas características, usa-se geralmente uma probabilidade de aceitação da nova solução (p), uma função conhecida por factor de Boltzmann, que é dada por $e^{-\Delta/T}$, onde T é o parâmetro do método que regula a probabilidade de soluções com pior custo.

O processo de aceitação é feito mediante a comparação da probabilidade “ p ”, e um número gerado aleatoriamente de uma distribuição uniforme no intervalo $[0, 1]$, da seguinte forma:

- Se este número for menor ou igual a “ p ”, aceita-se a solução;
- Se for maior que “ p ”, rejeita-se a solução.

A temperatura T assume inicialmente um valor elevado, T_0 . Após um número fixo de iterações (o qual representa o número de iterações para o sistema atingir o equilíbrio térmico a uma dada temperatura), a temperatura é gradualmente diminuída a uma taxa de arrefecimento α , tal que $T_n \leftarrow \alpha * T_{n-1}$, sendo $0 < \alpha < 1$. Como tal procedimento se

dá no início, há uma maior probabilidade de se escapar a mínimos locais e, à medida que T se aproxima de zero, o algoritmo comporta-se como um método de descida uma vez que diminui a probabilidade de se aceitarem movimentos que possam piorar ($T \rightarrow 0 \Rightarrow e^{-\Delta/T} \rightarrow 0$).

O procedimento termina quando a temperatura chega a um valor próximo de zero e não seja mais aceite qualquer solução que piore o valor da melhor solução, ou seja, quando o sistema atinge a estabilidade. A solução é obtida quando o sistema se encontra numa situação correspondente a um mínimo local.

3.3.3.2.3 Método de Monte Carlo

O método de Monte Carlo (MC) é um método estatístico utilizado em simulações estocásticas com diversas aplicações em áreas como a Física, Matemática e Biologia (Hromkovic, 2001). O método de Monte Carlo tem sido utilizado há bastante tempo como forma de obter aproximações numéricas de funções complexas. Este método envolve tipicamente a geração de observações de alguma distribuição de probabilidades e o uso da amostra obtida para aproximar a função de interesse. As aplicações mais comuns são em computação numérica para avaliar integrais. A ideia do método é escrever o integral que se deseja calcular como um valor esperado.

O método de Monte Carlo divide-se em três classes de algoritmos: erro-unilateral, erro-bilateral e erro-não-limitado.

3.3.3.2.4 Algoritmos genéticos

Um algoritmo genético (AG) é uma técnica de procura utilizada para determinar soluções aproximadas em problemas de optimização e procura (Goldberg, 1989). Algoritmos genéticos são uma classe particular de algoritmos evolutivos que usam técnicas inspiradas na Biologia como hereditariedade, mutação, selecção natural e recombinação.

Estes algoritmos são implementados na forma de simulação, em que é seleccionada uma população de representações abstractas da solução com o objectivo de encontrar melhores soluções. A evolução tem geralmente o seu início a partir de um conjunto de soluções aleatórias, sendo obtida através de gerações. Para cada geração, é avaliada a adaptação de cada solução na população; alguns indivíduos são seleccionados para a próxima geração, e recombinação ou sofrem mutação para formar uma nova população. A nova população é então utilizada como entrada para a próxima iteração do algoritmo.

Na prática, os algoritmos genéticos encontram a solução dos problemas utilizando um mecanismo de evolução a partir de uma solução aceitável para um nível superior de optimização. Em contraste com outros processos de optimização, os algoritmos genéticos centralizam-se numa optimização local. No entanto, a evolução é controlada para tentar percorrer todo o espaço de procura, o que pode tornar-se numa optimização global. Para cada iteração do algoritmo existe uma população de soluções. Essa população pode ser utilizada para evoluir para uma nova população na próxima iteração. A evolução obedece a uma hierarquia que utiliza os seguintes operadores específicos: reprodução, cruzamento e mutação.

O operador de mutação modifica aleatoriamente uma pequena parte da solução. No processo de procura ocorrem mutações muito pequenas; no entanto, esse passo é necessário quando se trata de uma procura de uma solução de optimização global. Usando apenas os operadores de reprodução e cruzamento, os algoritmos genéticos podem percorrer apenas uma parte do espaço de procura onde exista um óptimo local. O operador de mutação permite ao algoritmo saltar para uma nova solução noutro contexto. Desta forma, caso ocorram mutações suficientes, o algoritmo conseguirá atingir um valor óptimo global.

Os algoritmos genéticos exigem, obviamente, um elevado número de iterações; porém, a simplicidade do processo permite uma velocidade de execução razoável. Os algoritmos genéticos são particularmente usados na resolução de problemas sem restrições e funções objectivo difíceis de serem tratados. Os algoritmos genéticos funcionam com grandes populações de soluções, uma vez que tal significa um aumento nas possibilidades de encontrar uma boa solução. Infelizmente, o espaço necessário para uma solução simples é geralmente grande, pelo que esses algoritmos usam uma grande quantidade de memória para manter a população de todas as

soluções. O tempo gasto para gerar a população inicial é considerável, e não existe maneira de se saber previamente quando o algoritmo encontra uma solução ótima global. Como evolução natural, quanto mais se permitir o processamento do algoritmo genético, melhor será a solução obtida. A incerteza do fim da execução do algoritmo, combinada com o excessivo tamanho da população inicial de soluções, pode levar a um tempo muito longo de obtenção da solução.

Os algoritmos genéticos diferem dos algoritmos tradicionais de optimização basicamente em quatro aspectos:

- Baseiam-se numa codificação do conjunto das soluções possíveis, e não nos parâmetros da optimização em si;
- Os resultados são apresentados como uma população de soluções e não como uma solução única;
- Não necessitam de nenhum conhecimento derivado do problema; apenas de uma forma de avaliação do resultado;
- Usam transições probabilísticas e não regras determinísticas.

3.3.3.2.5 Busca Tabu

Os princípios básicos da Busca Tabu (BT) têm sido usados de diferentes maneiras e com nomes diferentes (Glover et al., 1977). Tiveram início no fim da década de 70 com Glover (1977), com procedimentos combinatórios aplicados a problemas não lineares.

As ideias básicas da forma actual foram desenvolvidas por Glover (1986), sendo também desenvolvidas, de forma independente, por Hansen (1986). Tentativas adicionais de formalização foram relatadas por Glover (1989), Werra (1989) e Hertz (1989).

A filosofia da BT decorre da observação de um conjunto de princípios da solução inteligente de problemas (Laguna, 1994). Um elemento fundamental na BT é o uso flexível da memória. Do ponto de vista da BT, esse uso engloba o processo de criar

estruturas que permitam explorar eficientemente a história de cada solução gerada, fazendo com que as informações obtidas anteriormente sirvam como parâmetros para a análise e obtenção de novas soluções.

O uso flexível da memória permite guiar o processo de busca no conjunto de soluções do problema, efectuando repetidos movimentos de uma solução para outra solução vizinha, e determinando alternativas de busca a serem qualificadas como proibidas ou atractivas.

A Busca Tabu é um procedimento de optimização local que admite a aceitação de soluções piores para escapar de óptimos locais. Na sua forma original, em cada iteração, procura-se um óptimo local seleccionando-se o melhor vizinho s' da vizinhança $N(s)$ da solução corrente s . Independentemente de $f(s')$ ser melhor ou pior que $f(s)$, s' será sempre a nova solução corrente. Entretanto, apenas este mecanismo não é suficiente para escapar de óptimos locais uma vez que pode haver regresso a uma solução previamente gerada.

Para evitar isso, o algoritmo usa o conceito de lista tabu. Esta lista associa a todos os movimentos feitos um certo atributo (tabu) para um determinado número de iterações, conhecido como tempo tabu. Tais movimentos são proibidos a menos que a solução satisfaça o critério pretendido; em geral, que seja estritamente melhor que a melhor solução encontrada até então. Os atributos são escolhidos para prevenir o regresso a soluções visitadas recentemente, e, para evitar que o algoritmo entre em ciclo, devem possuir um critério de paragem, geralmente um determinado número de iterações em que não haja obtenção de melhores resultados.

Na sua descrição mais simples, uma lista tabu contém as soluções que foram visitadas no passado recente. Outras estruturas da lista tabu proíbem as soluções que têm determinados atributos (por exemplo, no problema do caixeiro viajante (CV) soluções que incluem determinados arcos ou impedem determinados movimentos, pelo que um arco que seja adicionado a um trajecto de CV não pode ser removido nos movimentos seguintes).

Os atributos seleccionados nas soluções recentemente visitadas são marcados como tabu-activos. As soluções que contêm elementos tabu-activos são tabus. As listas tabu que contêm atributos são muito mais eficazes, embora levistem um novo

problema. Quando um único atributo for proibido como tabu, tipicamente várias soluções tornam-se tabu. Algumas destas soluções, que devem agora ser evitadas, poderão ser de excelente qualidade e podem não ter sido visitadas.

Para superar este problema, os critérios de aspiração são introduzidos permitindo o cancelamento do estado do tabu de uma solução para incluí-la na solução permitida. Um critério de aspiração geralmente usado é o de permitir soluções que são melhores do que a solução conhecida actualmente como a melhor.

3.3.3.3 Algoritmos compostos

Estes métodos referem-se a uma utilização sequencial dos algoritmos apresentados anteriormente. Utilizam procedimentos de construção para obter a rota e procedimentos de melhoria no sentido de otimizar a rota obtida. Os procedimentos são os seguintes:

- 1- Obter uma solução inicial usando um dos procedimentos de construção de rotas, apresentados anteriormente;
- 2- Aplicar o 2-opt para a rota encontrada;
- 3- Aplicar o 3-opt.

Este processo é relativamente rápido e fornece resultados adequados.

Em resumo, enumeram-se os principais algoritmos de resolução do problema do caixeiro-viajante descritos anteriormente:

- | | |
|---------------------|---------------------------|
| 1- Métodos exactos: | - Algoritmos de Dijkstra; |
| | - Algoritmos de Floyd; |
| | - Método Húngaro; |
| | - Programação inteira; |
| | - Programação dinâmica; |

- 2-Métodos Heurísticos:
- Construção de rotas:
 - Clarke and Wright;
 - Inserção;
 - Obtenção de sementes;
 - Melhorias de rotas:
 - K-opt;
 - Recozimento simulado;
 - Método Monte Carlo;
 - Algoritmos genéticos;
 - Busca tabu;
 - Compostos.

3.4 O problema dos m - caixeiros viajantes

Este problema é uma generalização do problema do caixeiro-viajante, possuindo mais aplicações práticas.

O problema clássico de rotas de veículos é considerado um problema de múltiplos caixeiros viajantes (MCV). Este problema consiste num problema de caminhos a percorrer por m caixeiros viajantes, obtendo m rotas, todos com partida e chegada a uma mesma cidade, de tal forma que todas as cidades sejam visitadas uma única vez, a um custo mínimo. O objectivo é obter as rotas minimizando a distância total percorrida.

Cada caixeiro deve sair de um ponto de partida, visitar diferentes nós da rede, uma única vez e regressar ao ponto inicial. Cada nó deverá ser visitado apenas por um caixeiro.

A Figura 3.8 ilustra o problema para o caso de 3 caixeiros-viajantes:

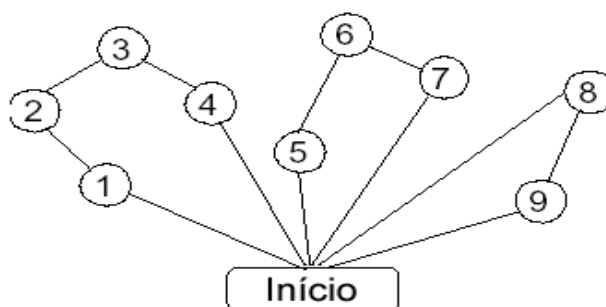


Figura 3.8 - Percursos diferentes para cada caixeiro.

O algoritmo desenvolvido por Bodin e Golden (1983) permite transformar um problema de m -caixeiros-viajantes no problema de um único caixeiro-viajante:

Etapa 1 Criar m nós, cópias do ponto de partida, designados por $D_1, D_2, \dots, D_m$, em que cada um se liga aos outros nós tal qual o ponto de partida, com arcos com o mesmo comprimento, formando uma rede expandida;

Etapa 2 As m -cópias não devem ligar-se entre si; caso se liguem, devem possuir arcos com comprimento excedendo a soma de todas as ligações, pois, desta forma, a ligação entre duas cópias não será escolhida para a rota;

Etapa 3 Após aplicar um método de optimização, transformar as cópias num único nó. Desta forma, a rota para um único caixeiro-viajante será transformada em m -rotas, uma para cada caixeiro-viajante.

Os esquemas das Figuras 3.9 e 3.10 ilustram o método descrito anteriormente. Como exemplo, considere-se um problema com 4 nós, sendo o nó 1 o ponto de partida, e dois caixeiros-viajantes. Para cada caixeiro é criada uma cópia do nó 1, D_1 e D_2 , que respeitam as condições de ligação do nó 1. Admita-se que a rota óptima na rede expandida é D_1 -3- D_2 -4-2- D_1 . Se contrair a rede, obtêm-se as duas rotas 1-3-1 e 1-4-2-1, respectivamente para cada caixeiro-viajante.

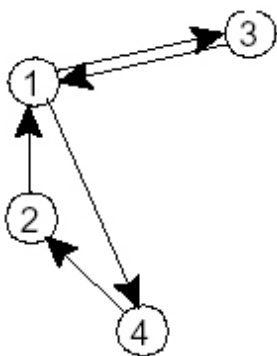


Figura 3.9 - Rede contraída.

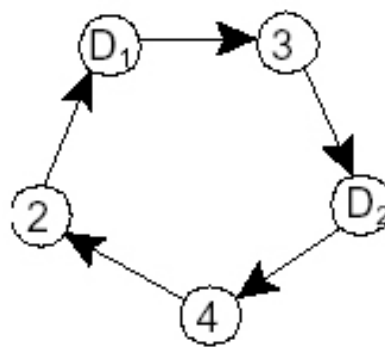


Figura 3.10 - Rede expandida.

3.5 Formulação clássica de atribuição de rotas

O problema de atribuição de rotas a veículos (caixeiros-viajantes) define-se como um sub-conjunto de rotas de passagem por determinados pontos (de recolha), partindo de um ponto inicial, com o objectivo de minimizar a distância total percorrida (Bodin et al., 1983). Os veículos possuem restrições em termos de capacidade e de tempo. Todos os veículos partem e chegam ao mesmo ponto.

A função objectivo é definida por:

$$\text{Min} \sum_{i=1}^N \sum_{j=1}^N \sum_{v=1}^V c_{ij} x_{ijv} \quad (3.10)$$

Sujeito a:

$$\sum_{i=1}^N \sum_{v=1}^V x_{ijv} = 1 \quad (j=2, \dots, N) \quad (3.11)$$

$$\sum_{j=1}^N \sum_{v=1}^V x_{ijv} = 1 \quad (i=2, \dots, N) \quad (3.12)$$

$$\sum_{i=1}^N x_{ipv} - \sum_{j=1}^N x_{p jv} = 0 \quad (v=1, \dots, V; p=1, \dots, N) \quad (3.13)$$

$$\sum_{i=1}^N d_i \left(\sum_{j=1}^N x_{ijv} \right) \leq K_v \quad (v=1, \dots, V) \quad (3.14)$$

$$\sum_{i=1}^N t_{iv} \sum_{j=1}^N x_{ijv} + \sum_{i=1}^N \sum_{j=1}^N t_{ijv} x_{ijv} \leq T_v \quad (v=1, \dots, V) \quad (3.15)$$

$$\sum_{j=2}^N x_{1jv} \leq 1 \quad (v=1, \dots, V) \quad (3.16)$$

$$\sum_{i=2}^N x_{i1v} \leq 1 \quad (v=1, \dots, V) \quad (3.17)$$

$$X \in S \quad (3.18)$$

$$x_{ijv} = \begin{cases} 0 \\ 1 \end{cases} \quad (v=1, \dots, V; i,j=1, \dots, N) \quad (3.19)$$

onde:

N - Número de nós;

V - Número de Veículos;

K_v - capacidade do veículo v ;

T_v - Tempo máximo para a rota de cada veículo;

d_i - Recolha no nó i ;

t_{iv} - Tempo necessário do veículo v para recolha no nó i ;

t_{ijv} - Tempo de viagem do veículo v a partir do nó i até ao nó j ;

C_{ij} - Custo da viagem do nó i para o nó j ;

$x_{ijv}=1$ - Arco i - j percorrido pelo veículo v ;

$x_{ijv}=0$ - caso contrário.

As restrições são estabelecidas da seguinte forma:

- Função custo a ser minimizada (equação 3.10);
- Cada nó é percorrido por um único veículo (equações 3.11 e 3.12);
- Assegura a continuidade da rota (equação 3.13);
- Assegura que a capacidade de cada veículo não é ultrapassada (equação 3.14);
- Assegura que o tempo de percurso de cada veículo não é ultrapassado (equação 3.15);
- Garante o não uso desnecessário de veículos (equações 3.16 e 3.17);
- Restrições de quebra de sub-rotas (equação 3.18).

Note-se que as restrições apresentadas pelas equações 3.11 e 3.13 implicam a restrição da equação 3.12 e que as restrições das equações 3.13 e 3.16 implicam a restrição apresentada na equação 3.17, fazendo com que as equações 3.12 e 3.16 possam ser retiradas. Note-se ainda que é assumido que a capacidade de qualquer veículo é superior à necessidade inerente a qualquer ponto da rota.

A resolução deste problema efectua-se usando uma heurística qualquer de construção de rotas (descritas anteriormente) a partir da matriz de custos e, em seguida, usando um algoritmo de melhoria, conforme citado nos pontos anteriores.

3.6 Abordagem ao problema de recolha de RSU

3.6.1 Introdução

Tal com já referido anteriormente, o sistema de recolha de RSU em zonas urbanas refere-se a um conjunto de unidades a serem visitadas, geograficamente distribuídas, podendo ainda considerar que a recolha porta a porta pode ser vista como a existência de um ponto de concentração de RSU, localizado na respectiva rua e visto como um contentor. Tendo em conta a definição clássica de planeamento de rotas de veículos, designado por *vehicle routing problem* (VRP), todos os contentores deverão ser esvaziados.

O problema do esvaziamento de um contentor poderá ser condicionado por dois factores: as necessidades periódicas do contentor e a quantidade de lixo a partir da qual o contentor deverá ser esvaziado. Desta forma, há que ter em conta que os contentores cheios têm de ser esvaziados, os outros sê-lo-ão caso circunstâncias diversas assim a tal obriguem.

A frota a usar é limitada, o volume de carga dos veículos poderá ou não ser constante, cada veículo poderá servir qualquer ponto de recolha, os veículos partem de um ponto único (ponto de partida), sendo os RSU depositados num depósito final (destino).

O planeamento considera essencialmente os custos variáveis de cada veículo e proporcionais às distâncias percorridas, consumo, desgaste, bem como o número de

peessoas afectas a cada veículo. A definição das matrizes de custo e tempos entre diferentes pontos de recolha é obtida a partir da determinação dos melhores caminhos a partir duma base de informação geográfica. Ao tempo de transição entre pontos deverá ser adicionado o tempo de carga, obtendo-se assim o tempo necessário que separa o primeiro contentor do segundo.

Um dos objectivos do planeamento de rotas consiste na minimização da distância percorrida pelos veículos, atendendo a factores condicionantes e a restrições relacionadas com as necessidades de recolha.

3.6.2 Definição

O VRP pode ser visto como uma rede espacial caracterizada por pontos que representam os contentores e por arcos que representam os movimentos no espaço entre os diversos contentores.

Seja $N=\{1, 2, \dots, n-1\}$ e $K=\{1, 2, \dots, m\}$ o conjunto dos contentores e os veículos disponíveis limitados a uma capacidade Q_k e a um tempo T_k .

O depósito é representado pelo ponto 0 e o destino pelo ponto n , que serão comuns para todos os veículos. Seja V_k o conjunto dos pontos e A_k o conjunto dos arcos entre dois pontos, para um veículo k . Um arco $(i, j) \in A_k$ nunca terá origem no ponto n ou por destino o ponto 0. No entanto, $V_k = N \cup \{0, n\}$ e $A_k = \{N \cup \{0\} \times N \cup \{n\}\}$.

Os arcos $(i, j) \in A_k$ definem um trajecto entre os contentores i e j , os arcos $(0, i)$ e (j, n) definem, respectivamente, o primeiro e último percurso duma rota que passa em i e j . Finalmente, um arco $(0, n) \in A_k$ representa a não utilização do veículo k .

A cada contentor está associada uma carga p_i a ser carregada, e, entre cada percurso, está associado um tempo para cada veículo, t_{ij}^k . Estes dois parâmetros podem ser vistos como os coeficientes de consumos dos recursos do problema, ou seja a capacidade do veículo e a duração das rotas que determinam a validade da respectiva rota.

A cada arco $(i, j) \in A_k$ está associado um custo c_{ijk} , dependente do veículo e do percurso. Um dado veículo poderá por várias razões estar impossibilitado de percorrer determinados percursos, definindo-se assim $c_{ijk} = +\infty$. Deve-se ainda considerar o custo fixo dos veículos, adicionando a todos o c_{0jk} , visto que cada veículo sai uma única vez do ponto de partida.

É ainda necessário considerar prioridades de recolha pois, conforme referido anteriormente, é possível impor diferentes necessidades de esvaziamento de determinados contentores em pontos diferentes; por isso, surgem p_{ij} como representando as prioridades do percurso i para j .

Finalmente, há que ponderar os custos da rota de cada veículo, os custos de saída dos veículos do ponto de partida e as prioridades de recolha nos diversos percursos.

3.6.3 Formulação do problema usando programação linear

Esta formulação pressupõe variáveis de decisão com três índices; $x_{ijk}=1$ significa que o veículo k vai para o local j depois de ter vindo de i ($x_{ijk} = 0$ no caso contrário).

A forma matemática dada por Christophides (1979) ao problema é muito completa e apresenta um conjunto de restrições a nível de percursos, cuja formulação pode ser adaptada da seguinte forma:

$$\text{Min } Z = \alpha \sum_{i=0}^N \sum_{j=0}^N \sum_{k=1}^M c_{ijk} x_{ijk} + \beta \sum_{k=1}^M (c_k \sum_{j=1}^N x_{0jk}) + \gamma \sum_{j=1}^N \sum_{i=0}^N (p_{ij} \sum_{k=1}^M x_{ijk}) \quad (3.20)$$

sujeito a:

$$\sum_{i=0}^N \sum_{k=1}^M x_{ijk} \leq 1, \quad \forall j \quad (3.21)$$

$$\sum_{i=0}^N x_{ipk} - \sum_{j=0}^N x_{pj k} = 0, \quad \forall k, \forall p \quad (3.22)$$

$$\sum_{i=0}^N \sum_{j=0}^N t_{ij} x_{ijk} + \sum_{i=1}^N (u_i \sum_{j=0}^N x_{ijk}) < T_k^F - T_k^S, \forall k \quad (3.23)$$

$$\sum_{j=1}^N x_{0jk} \leq 1, \forall k \quad (3.24)$$

$$\sum_{i=0}^N p_i (\sum_{j=1}^N x_{ijk}) \leq Q_k, \forall k \quad (3.25)$$

$$Y_i - Y_j + N \sum_{k=1}^M x_{ijk} \leq N - 1, \forall i \neq j, Y_i \text{ e } Y_j - \text{arbitrários} \quad (3.26)$$

$$x_{ijk} \in [0,1], \forall i, j, k \quad (3.27)$$

As três componentes da função objectivo (Equação 3.19), ponderadas pelas constantes α , β e γ , pretendem respectivamente:

- Minimizar o custo global das viagens, sendo C_{ijk} o custo do percurso de i para j pelo veículo k ;
- Minimizar o custo do número total de veículos usados, sendo C_k o custo relativo a fazer sair um veículo do “armazém”;
- Minimizar as prioridades p_{ij} atribuídas aos vários tipos de percursos - os percursos obrigatórios terão uma prioridade mais elevada que os percursos de ligação opcionais em vazio, necessários nas ligações entre percursos obrigatórios.

As restrições significam que:

- Cada local só terá um veículo a passar por lá uma vez (equação 3.21);
- Garante que um veículo que tenha como destino um ponto terá que igualmente ter nesse ponto uma origem posterior (equação 3.22);

- Significa que o tempo de percurso entre dois pontos (t_{ij}), assim como o tempo necessário para efectuar a carga no ponto i (u_i), têm de estar dentro do período de trabalho ($T_k^f - T_k^s$) (equação 3.23);
- Implica que o mesmo veículo não seja afectado a duas sequências de percursos diferentes através da garantia que cada veículo só sai uma vez do “armazém” (equação 3.24);
- Obriga a que a carga de cada veículo não ultrapasse a carga máxima admissível de cada veículo (Q_k) (equação 3.25);
- Elimina os sub-ciclos e força cada viagem a passar pela origem e ponto de descarga, pelo menos uma vez (equação 3.26).

3.6.4 Programação linear binária

Descreve-se em seguida um dos métodos utilizados para a resolução de problemas de programação linear, em que às variáveis apenas se atribuem os valores 0 ou 1 (Balas, E., 1965).

Definição

Seja o problema de programação linear binária (PPLB) o seguinte:

$$\text{Max} \quad \sum_{j=1}^n c_j x_j \quad (3.28)$$

$$\text{sujeito a:} \quad \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i=1, \dots, m \quad (3.29)$$

$$x_j \in \{0,1\}, \text{ com } c_1 \geq c_2 \geq \dots \geq c_n \geq 0 \quad (3.30)$$

Sendo a_{ij} os coeficientes da matriz de restrições do problema.

Transformações de equivalência

No caso do problema não se apresentar desta forma, dever-se-á proceder às seguintes transformações:

- a) Efectuar a substituição $x_j = 1 - x'_j$, para cada $c_i < 0$;
- b) Trocar as restrições do tipo $\sum_{j=1}^n a_{ij} x_j = b_i$ por duas restrições:
 - $\sum_{j=1}^n a_{ij} x_j \leq b_i$;
 - $\sum_{j=1}^n a_{ij} x_j \geq b_i$;
- c) Trocar as restrições do tipo $\sum_{j=1}^n a_{ij} x_j \geq b_i$ por $-\sum_{j=1}^n a_{ij} x_j \leq -b_i$;
- d) Reordenar as variáveis de modo a que $c_1 \geq c_2 \geq \dots \geq c_n$.

Algoritmo

O algoritmo consiste numa busca numa árvore binária na qual cada nó corresponde a uma solução parcial do PPLB cujos valores das primeiras k variáveis são conhecidos.

Após transformar o problema na forma padrão, pode definir-se o algoritmo da seguinte forma: seja z_{opt} e z_{est} , respectivamente, a solução óptima e a solução estimada da função objectivo:

Passo 0 $z_{opt} = -\infty$ e $k=1$;

Passo 1 $x_k=1$

Verificar se:

$$z_{est} = \sum_{j=1}^k c_j x_j + \sum_{j=k+1}^n c_j > z_{opt} \text{ e } \sum_{j=1}^k a_{ij} x_j \leq b_i - \sum_{j=k+1}^n \min(0, a_{ij}), i=1, \dots, m$$

em caso afirmativo, saltar para o passo 3;

Passo 2 $x_k=0$

Verificar se:

$$z_{est} = \sum_{j=1}^k c_j x_j + \sum_{j=k+1}^n c_j > z_{opt} \text{ e } \sum_{j=1}^k a_{ij} x_j \leq b_i - \sum_{j=k+1}^n \min(0, a_{ij}), i=1, \dots, m$$

caso contrário, saltar para o passo 4;

Passo 3 $k=k+1$, se $k \leq n$, saltar para o passo 1, se não uma solução completa, melhor que a solução ótima temporária, foi encontrada. Guardar a solução como sendo uma nova solução ótima temporária. Actualizar o valor de z_{opt} ;

Passo 4 Encontrar $K = \{j \mid x_j = 1 \text{ e } 0 < j < k\}$. Se $K = \{\}$, então fim; a solução ótima do problema é a solução temporária. Caso contrário, determine $k = \max\{j \mid j \in K\}$ e saltar para o passo 2.

Exemplo

Segue-se o resultado da aplicação deste algoritmo a um problema de programação linear binária:

$$\max \quad 10x_1 + 9x_2 + 7x_3 + 5x_4 + 2x_5 + 1x_6$$

$$s.a : \quad 1x_1 - 2x_2 + 2x_3 + 1x_4 - 2x_5 + 1x_6 \leq 1$$

$$2x_1 + 1x_2 - 2x_3 - 1x_4 + 1x_5 + 2x_6 \leq 2$$

$$3x_1 + 2x_2 - 1x_3 + 3x_4 - 4x_5 + 1x_6 \leq 2$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \in \{0, 1\}$$

A Tabela 3.9 apresenta as várias iterações, assim como os resultados obtidos.

Tabela 3.9 - Exemplo de aplicação da programação linear binária.

k	x_j	A	B	C	D	E	F	G	H
0									$-\infty$
1	1	10	24	34	(1;2;3)	(5;5;7)	Ok	Ok	
2	1,1	19	15	34	(-1;3;5)	(3;5;7)	Ok	Ok	
3	1,1,1	26	8	34	(1;1;4)	(3;3;6)	Ok	Ok	
4	1,1,1,1	31	3	34	(2;0;7)	(3;2;6)	Ok	--	
4	1,1,1,0	26	3	29	(1;1;4)	(3;2;6)	Ok	Ok	
5	1,1,1,0,1	28	1	29	(-1;2;0)	(1;1;2)	Ok	Ok	
6	1,1,1,0,1,1	29	0	29	(0;4;1)	(1;1;2)	Ok	--	
6	1,1,1,0,1,0	28	0	28	(-1;2;0)	(1;1;2)	Ok	Ok	28
5	1,1,1,0,0	26	1	27			--		
3	1,1,0	19	8	27			--		
2	1,0	10	15	25			--		
1	0	0	24	24			--		
	Fim								

$$A = \sum_{j=1}^k c_j x_j ; B = \sum_{j=k+1}^n c_j , C = z_{est}, D = \sum_{j=1}^k a_{ij} x_j , E = b_i - \sum_{j=k+1}^n \min(0, a_{ij}) .$$

$$F = Ok_1, G = Ok_2 \text{ e } H = z_{opt}.$$

Solução óptima: $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, z^* = 28$.

Implementação

Atendendo ao método descrito anteriormente, é possível transformar o problema da recolha de RSU num problema equivalente ao anterior efectuando as seguintes transformações:

1º - $\text{Max} \sum_{j=1}^n c_j x_j$ é equivalente a $\text{Min} \sum_{j=1}^n -c_j x_j$;

2º - Como os coeficientes das variáveis que entram na função objectivo deverão ser

todos positivos, usa-se uma tabela de conversão que transforma os custos positivos em custos negativos de uma forma inversa, ou seja, custos elevados passarão a ser representados por valores negativos próximos de zero, e custos baixos, por valores negativos, maiores em valor absoluto.

3º - Efectuar o desdobramento das restrições conforme descrito anteriormente.

Sendo assim, o modelo apresentado assumirá a forma seguinte:

$$\text{Max } Z = \alpha \sum_{i=0}^N \sum_{j=0}^N \sum_{k=1}^M c_{ijk}^* x_{ijk} + \beta \sum_{K=1}^M (c_k^* \sum_{j=1}^N x_{0jk}) + \gamma \sum_{j=1}^N \sum_{i=0}^N (p_{ij}^* \sum_{k=1}^M x_{ijk}) \quad (3.31)$$

* - Valores convertidos da forma referida anteriormente

Sujeito a:

$$\sum_{i=0}^N \sum_{k=1}^M x_{ijk} \leq 1, \quad \forall j \quad (3.32)$$

$$\sum_{i=0}^N x_{ipk} - \sum_{j=0}^N x_{pj k} \leq 0, \quad \forall k, \forall p \quad (3.33)$$

$$-\sum_{i=0}^N x_{ipk} + \sum_{j=0}^N x_{pj k} \leq 0, \quad \forall k, \forall p \quad (3.34)$$

$$\sum_{i=0}^N \sum_{j=0}^N t_{ij} x_{ijk} + \sum_{i=1}^N (u_i \sum_{j=0}^N x_{ijk}) < T_k^F - T_K^S, \quad \forall k \quad (3.35)$$

$$\sum_{j=1}^N x_{0jk} \leq 1, \quad \forall k \quad (3.36)$$

$$\sum_{i=0}^N p_i (\sum_{j=1}^N x_{ijk}) \leq Q_k, \quad \forall k \quad (3.37)$$

$$Y_i - Y_j + N \sum_{k=1}^M x_{ijk} \leq N - 1, \quad \forall i \neq j, \quad Y_i \text{ e } Y_j - \text{arbitrários} \quad (3.38)$$

$$x_{ijk} \in [0,1], \quad \forall i, j, k \quad (3.39)$$

3.7 Referências do capítulo

Ahuja, R., Magnanti, T., Orlin, J., Network Flows: Theory, Algorithms and Applications, Prentice-Hall, 1993.

Alves, C. M., Planeamento de rotas de veículos em sistemas de recolha/distribuição, dissertação de Mestrado em Engenharia Industrial, Especialidade de Logística e Distribuição, Universidade do Minho, 126, Dezembro, 2000.

Alves, J. C., Provas de Aptidão Científica e Capacidade Pedagógica. FEUP, 1989.

Andrade, J., Ares, J., Garcia, R., Pazos, J., A methodological framework for viewpoint-oriented conceptual modeling, IEEE Trans. on Software Engineering, 30, 5, 2004.

Balas, E., An Additive Algorithm For Solving Linear Programs With Sero-one Variables, ORSA, 13, 517-546, 1965.

Balas, E., The Prize Collecting Traveling Salesman Problem, Networks, 19, 1989.

Balinski, M., Quandt, R., On an Integer Program for Delivery Problem, Operations Research, 12, 1964.

Ball, M., Routing and Scheduling of Vehicles and Crews. The state of the Art. Operations Research, 10, 69-211, 1983.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P., Branch-and-price: Column Generation for Solving Huge Integer Programs, Operations Research, 46, 1998.

Bazgana, C., Hassinb, R., Monnota, J., Approximation algorithms for some vehicle routing problems, Discrete Applied Mathematics, 146, 27 - 42, 2005.

Bektas, T., The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega, 34, 209 - 219, 2006.

Bellman R., Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1957.

Bellmore, M., Nemhauser, G., The travelling salesman problem: A survey, Operations Research, 16, 538-558, 1968.

Billy, E. G., Leland, R. M., 1971.

Bock, F., An algorithm for solving “traveling-salesman” and related network optimization problems, unpublished manuscript associated with talk presented at the 14th ORSA, National Meeting, 1958.

Bodin L., Golden, B., Assad, A., Ball, M., Routing and Scheduling of Vehicles and Crews, the State of the Art, Computers & Operations Research 10, 69-211, 1983.

Burkard, R. E., Traveling salesman and assignment problem: A survey. In P. L. Hammer, E. L. Johnson, & B. H. Korte (Eds.), Discrete optimization, 1, 193-215, 1979.

Carravilla, M., Modelos e Algoritmos para o Planeamento Hierárquico da Produção, Aplicações a um Caso de Estudo, Tese de Doutoramento, FEUP, 1996.

Chabrier, A., Vehicle Routing Problem with elementary shortest path based column generation, Computers & Operations Research, 33, 2972-2990, 2006.

Christophides, N., Mingozzi, A., Toth, P., Sandi, C., The Traveling Salesman Problem, in Combinatorial Optimization, Eds. John Wiley and Sons, New York, 1979.

Clarke, G., Wright, J., Scheduling of vehicles from a central depot to a number of delivery points, Operations Research, 12, 4, 568-581, 1964.

Cormen, T. H., Leiserson, C., E., Rivest, R. L., Introduction to Algorithms, first edition, MIT Press and McGraw-Hill, 1990.

Cormen, T. H., Leiserson, C., E., Rivest, R. L., Introduction to Algorithms, second edition, McGraw-Hill, 1999.

Cormen, T. H., Leiserson, C., E., Rivest, R. L., Stein, C., Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001.

Dantzig, G. B., Wolfe, P., Decomposition Principle for Linear Programs, Operations Research, 8, 1960.

Desrochers, M., Desrosiers, J., Solomon, M., A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows, Operations Research 40, 1992.

Desrochers, M., Soumis, F., A Column Generation Approach to the Urban Transit Crew Scheduling Problem, Transportation Science, 23, 1989.

Desrosiers J., Dumas, Y., Solomon, M., Soumis, F., Time Constrained Routing and Scheduling, Handbooks in OR & MS, 6, 35-139, 1995.

Desrosiers, J., Soumis, F., Desrochers, M., Routing with Time Windows by Column Generation, Networks, 14, 1984.

Dijkstra, E. W., A note on two problems in connexion with graphs. In: Numerische Mathematik, 1, 269-271, 1959.

Fekete, S., Schepers, J., New Classes of Lower Bounds for Bin-Packing Problems, Lecture Notes in Computer Science, 1412, 1998.

Fisher M., Vehicle Routing, Handbooks in OR & MS, 8, 1995.

Ford, L., Fulkerson, D., A Suggested Computation for Multicommodity Flows, Management Science, 5, 1958.

Garey, M. R., Johnson, D. S., Computers and Intractability: a Guide to the Theory Of NP Completeness. 1. ed. New York: W. H. Freeman, 1979.

Glover, F., Future paths for integer programming and links to artificial intelligence. Computers and Operations Research, Vol. 5, 533-549, 1986.

Glover, F., Heuristics for integer programming using surrogate constraints. Decision Science, 8, 156-166, 1977.

Glover, F., Laguna M., Tabu Search, Kluwer, Norwell, MA, 1997.

Glover, F., Tabu search - Part I, ORSA Journal on Computing, 1, 190-206 e Part II, ORSA Journal on Computing, 2, 4-32, 1989.

Goldberg, M., David, E., Genetic Algorithms in Search, Optimization, and Machine Learning, EUA: Addison-Wesley, 1989.

Goldberg, M., Luna, H., Optimização Combinatória e Programação Linear, Editora CAMPUS, 2000.

Golden, B. L., Assad, A., Levy L., Gheysens F., The Fleet Size and Mix Vehicle Routing Problem, Computers & Operations Research, 11, 1, 49-66, 1984.

Golden, B. L., Bondin, L., Doyle, T., Stewart Jr. W., Approximate Travelling Salesman Algorithms, Operations Research, Linthicum, 28, 2, 694-711, 1980.

Golden, B. L., Magnanti, T., Nguyew, Implementing Vehicle Routing Algorithms Networks, 7, 113-148, 1977.

Golden, B. L., Stewart, W.R., Empirical analysis of heuristics in The Traveling Salesman Problem, John Wiley & Sons, Inc., 1985.

Hansen, P., The steepest ascent mildest descent heuristic for combinatorial programming. Congress on Numerical Methods in Combinatorial Optimization, Capri, Itália, 1986.

Harold, K., The Hungarian Method for the assignment problem, Naval Research Logistic Quarterly, Kuhn's original publication, 2, 83-97, 1955.

Helsgaun, K., An effective implementation of the Lin-Kernighan Traveling Salesman Heuristic, European Journal of Operational Research, 126, 106-130, 2000.

Ho, S. C., Haugland, D., A tabu search heuristic for the vehicle routing problem with time windows and split deliveries, Computers and Operations Research, 31, 1947-1964, 2004.

Hoffman, K., Padberg, M., Solving Airline Crew Scheduling Problems by Branch-and-Cut, *Management Science*, 39, 1993.

Hromkovic, J., Algorithms for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics, Springer-Verlag London Berlin Heidelberg New York, 2001.

[Http://pt.wikipedia.org](http://pt.wikipedia.org), P versus NP, Dezembro de 2007.

Hu, T. C., Integer Programming and Network Flows, published by Addison-Wesley, 1969.

Jones, K., Lustig, I., Farvolden, J., Powell, W., Multicommodity Network Flows: The Impact of Formulation on Decomposition, *Mathematical Programming*, 62, 1993.

Karp, R. M., On the computational complexity of combinatorial problems, *Networks*, 5, 45-68, 1975.

Kima, B., Kimb, S., Sahoob, S., Waste collection vehicle routing problem with time windows, *Computers & Operations Research*, 33, 3624-3642, 2006.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., Optimization by Simulated Annealing, *Science*, 220, 4598, 671-680, 1983.

Kohl, N., Exact Methods for Time Constrained Routing and Related Scheduling Problems, Ph. D. thesis, DTU Technical University of Denmark, 1995.

Laguna, M., A guide to implementing tabu search. *Investigación Operativa*, 4, 5-25, 1994.

Laporte, G., Asef-Vaziril, A., Sriskandarjah, C., Some Applications of the generalized Travelling Salesman Problem, *Journal of Operations Research Society*, 47, 1461-1467, 1996.

Laporte, G., The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms, *European Journal of Operational Research*, 59, 345-358, 1992.

Lau, H., Sim, M., Teo, K., Vehicle routing problem with time windows and a limited number of vehicles, *European Journal of Operational Research*, 148, 559-569, 2003.

Lee, C., Epelman, M., White III, C., Bozer, Y., A shortest path approach to the multiple-vehicle routing problem with split pick-ups, *Transportation Research Part B*, 40, 265-284, 2006.

Lin, S., Computer solutions of the traveling salesman problem, *Bell Syst. Tech. J.*, 44, 2245-2269, 1965.

Lin, S., Kernighan, B. W., An Effective Heuristic Algorithm for the Traveling-Salesman Problem, *Operations Research*, 21, 498-516, 1973.

MacLaughlin, F. S., Pickhardt, P. C., *Quantitative Techniques for Management Decisions*. Boston, Houghton Mifflin Company, 1979.

Martello, S., Toth, P., *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, 1990.

Mazzeo, S., Loiseau, I., An Ant Colony Algorithm for the Capacitated Vehicle Routing, *Electronic Notes in Discrete Mathematics*, 18, 181-186, 2004.

Montané, F., Galvão, R., A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service, *Computers & Operations Research*, 33, 595-619, 2006.

Munkres, J. Algorithms for the assignment and transportation problems, *Journal of the Society for Industrial and Applied Mathematics* 5, 32-38, 1957.

Nemhauser, G., Wolsey, L., *Integer and Combinatorial Optimization* John Wiley & Sons, Inc., 1988.

Papadimitriou, Christos H., Steiglitz, Kenneth, *Combinatorial Optimization Algorithms and Complexity* Prentice Hall, Inc., 1982.

Paraíba, L. C., Fernandes, J. F. R., Ando, A. S., Um Algoritmo Heurístico de Contrução Paralela para o problema do M-Caixeiro viajante. Campinas. UNICAMP, 1990.

Petch, R., Salhi, S., A multi-phase constructive heuristic for the vehicle routing problem with multiple trips, *Discrete Applied Mathematics*, 133, 69 - 92, 2004.

Ramalhete, M., Guerreiro, J. e A. Magalhães, *Programação Linear*. McGraw-Hill, 1995.

Reeves, C. R., *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell, Blackwell Scientific Publications, 1993.

Reinelt, G., *Traveling Salesman: Computational Solutions for TSP Applications*, Springer: Lecture Notes in Computer Science, 1994.

Rosenkrantz, Stens. R., Lewis, P., An Analisis of Several Heuristics for the Travelling Salesman Problem, *SIAM Journal Computing*, 6, 563-581, 1977.

Russell, R., Chiang, W., Scatter search for the vehicle routing problem with time windows, *European Journal of Operational Research*, 169, 606-622, 2006.

Savelsbergh, M., A Branch-and-Price Algorithm for the Generalized Assignment Problem, *Operations Research*, 45, 1997.

Savelsbergh, M., Local Search in Routing Problems with Time Windows. *Annals of Operations Research* 4, 285-305, 1985.

Savelsbergh, M., Sol, M., The General Pickup and Delivery Problem, *Transportation Science*, 29, 1995.

Soumis, F., *Decomposition and Column Generation*, Les Cahiers du GERAD, 1997.

Tarantilis, C., Loannou, G., Prastacos, G., Advanced vehicle routing algorithms for complex operations management problems, *Journal of Food Engineering*, 70, 455-471, 2005.

Werra, D., Hertz, A., Tabu search techniques: a tutorial and an application to neural networks, OR Spektrum, 11, 131-141, 1989.

Wosley, L., Integer Programming, New York: John Wiley and Sons, 1998.

Capítulo 4 - Aplicação da Programação Linear Binária ao Problema da Recolha de RSU

Sumário do Capítulo

Este capítulo tem por objectivo validar o método de programação binária à resolução de problemas de recolha de RSU. São criadas várias situações com o objectivo de validar o método. É ainda referida a utilização da noção de super-contentor, passível de ser usada com a programação linear binária.

Finalmente, apresenta-se uma utilização recursiva da programação linear binária, utilizando a técnica de super-contentores e super-veículos, que permite transformar o nosso problema inicial em n problemas de dimensão mais reduzida.

4.1 Casos de estudo simples

4.1.1 Introdução

Com o objectivo de testar a programação linear binária conceptualizaram-se vários problemas de recolha de RSU. Por uma questão de simplicidade, formularam-se as seguintes hipóteses:

- 1) Todos os veículos possuem a mesma capacidade;
- 2) Os contentores possuem todos um volume unitário;
- 3) Considera-se na função objectivo (Equação 3.20), $\alpha = 1, \beta = 0, \gamma = 0$, isto é, pretende-se minimizar o custo global das viagens não tendo em consideração o custo dos veículos nem o estabelecimento de prioridades em determinados percursos;
- 4) Cada contentor está ligado aos contentores mais próximos por um número de percursos definido pelo utilizador e os custos são automaticamente calculados tendo em conta o número do contentor e o imediatamente antes, ou seja $c_{ijk} = |j - i|$, se existe ligação entre i e j ; caso contrário, $c_{ijk} = \infty$. Como exemplo, o custo de ir do contentor 3 para o 4 é 1, o custo de ir do contentor 3 para o 5 é 2, desde que haja ligação;
- 5) As rotas apenas estão limitadas pela capacidade do veículo, não se entrando com informação relativa ao horário de trabalho.

A Figura 4.1 apresenta a janela de introdução de dados na aplicação desenvolvida.

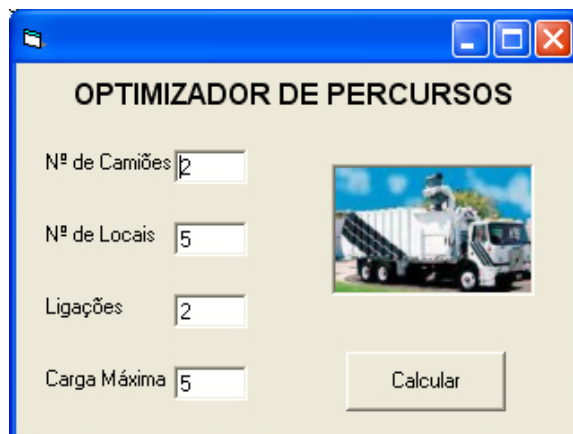


Figura 4.1 - Janela inicial do programa implementado para validação do modelo de optimização.

Dada a simplificação introduzida, o utilizador apenas necessita de especificar os seguintes dados:

- **Nº de Veículos:** Quantidade de veículos disponíveis na frota;
- **Nº de Locais:** Número de pontos de recolha, incluindo o ponto inicial (partida) e final (chegada);
- **Ligações:** Número de ligações possíveis em cada ponto. Por exemplo, se este valor for 2, significa que cada ponto só tem ligação possível aos dois pontos imediatamente a seguir, conforme se ilustra na Figura 4.2;
- **Carga Máxima:** Carga máxima de cada veículo, em unidades. Refira-se que cada contentor tem carga de uma unidade, conforme foi especificado inicialmente.

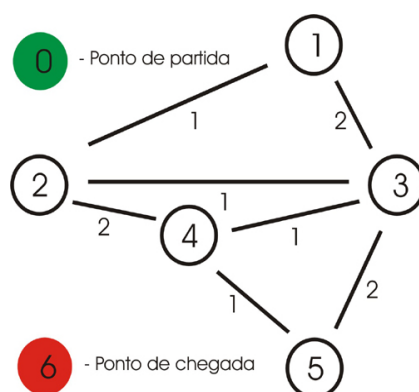


Figura 4.2 - Ligações possíveis e custos entre os seis pontos.

Seja N o número de locais, incluindo a origem e o destino, e M o número de veículos que compõem a frota.

É de salientar que, de acordo com a formulação do problema, o algoritmo irá gerir um sistema composto por $N*N*M$ incógnitas (Equação 3.31) e $N*N+2M*N+2M$ equações, (Equação 3.32 (N) + Equação 3.33 ($M*N$) + Equação 3.34 ($M*N$) + Equação 3.36 (M) + Equação 3.37 (M) + Equação 3.38 ($N*N-N$)).

4.1.2 Resultados

Os resultados apresentados a seguir referem-se à execução da aplicação informática num computador Pentium 4, a 2.66 Ghz, e com 512 Mbytes de RAM.

Tabela 4.1 - Resultados da aplicação do programa a diferentes casos simples.

Caso	Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função objectivo
1	2	7	2	5	98	81	6
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 6 Veículo 1: não usado Os resultados estão de acordo com o previsto uma vez que cada veículo tem capacidade para comportar a carga de todos os contentores, não sendo utilizado o segundo veículo.						
2	2	7	2	4	98	81	12
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 6 Veículo 1: 0 > 5 > 6 Um veículo não é suficiente para fazer a recolha dado que se dispõe de 5 unidades para recolher e cada veículo apenas consegue comportar 4 unidades. Observa-se a afectação de 4 locais a um veículo, sendo apenas afectado um local ao segundo veículo.						

Tabela 4.1 - Resultados da aplicação do programa a diferentes casos simples (continuação).

Caso	Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função objectivo
3	2	9	2	7	162	121	8
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 6 > 7 > 8 Veículo 1: não usado O que significa que apenas é usado um veículo dado que este tem capacidade para comportar a carga de todos os pontos de recolha, não sendo utilizado o segundo veículo.						
4	2	9	2	5	162	121	16
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 8 Veículo 1: 0 > 6 > 7 > 8 Neste exemplo não é suficiente um veículo para fazer a recolha dado que se têm 7 unidades para recolher e cada veículo apenas consegue comportar 5 unidades. Observe-se que são afectados 5 locais a um veículo, sendo apenas afectado 2 contentores ao segundo veículo.						
5	2	11	2	9	242	169	10
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 6 > 7 > 8 > 9 > 10 Veículo 1: não usado Mais uma vez, apenas é usado um veículo dado que este tem capacidade para comportar a carga de todos os pontos de recolha, não sendo utilizado o segundo veículo.						
6	2	11	2	5	242	169	20
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 10 Veículo 1: 0 > 6 > 7 > 8 > 9 > 10 Neste exemplo, não é suficiente um veículo para fazer a recolha dado que se têm 9 unidades para recolher e cada veículo apenas consegue comportar 5 unidades. Observa-se que são afectados 5 locais a um veículo, sendo apenas afectado 4 contentores de recolha ao segundo veículo.						

Tabela 4.1 - Resultados da aplicação do programa a diferentes casos simples (continuação).

Caso	Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função objectivo
7	3	7	2	5	147	97	6
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 6 Veículo 1: não usado Veículo 2: não usado Mais uma vez, apenas é usado um veículo dado que este tem capacidade para comportar a carga de todos os pontos de recolha, não sendo utilizados o segundo e terceiro veículos.						
8	3	7	2	3	147	97	12
	Veículo 0: 0 > 1 > 2 > 3 > 6 Veículo 1: 0 > 4 > 5 > 6 Veículo 2: não usado Desta vez, não é suficiente um veículo para fazer a recolha dado que se têm 5 unidades para recolher e cada veículo apenas consegue comportar 3 unidades. Observa-se que são afectados 3 locais a um veículo, sendo apenas afectados 2 contentores ao segundo veículo. O terceiro não é usado.						
9	3	7	2	2	147	97	18
	Veículo 0: 0 > 1 > 2 > 6 Veículo 1: 0 > 3 > 4 > 6 Veículo 2: 0 > 5 > 6 Neste exemplo, não é suficiente dois veículos para fazer a recolha dado que se têm 5 unidades para recolher e cada veículo apenas consegue comportar 2 unidades. Observa-se que são afectados 2 locais a dois veículos, sendo afectado 1 contentor ao terceiro veículo.						

Tabela 4.1 - Resultados da aplicação do programa a diferentes casos simples (continuação).

Caso	Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função objectivo
10	3	9	2	7	243	141	8
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 6 > 7 > 8 Veículo 1: não usado Veículo 2: não usado Como previsto, apenas é usado um veículo dado que este tem capacidade para comportar a carga de todos os pontos de recolha, sendo poupada a utilização do segundo e terceiro veículos.						
11	3	9	2	5	243	141	16
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 8 Veículo 1: 0 > 6 > 7 > 8 Veículo 2: não usado Neste exemplo, não é suficiente um veículo para fazer a recolha dado que temos 7 unidades para recolher e cada veículo apenas consegue comportar 5 unidades. Observa-se que são afectados 5 locais a um veículo, sendo apenas afectados 2 pontos de recolha ao segundo veículo, poupando a utilização do terceiro veículo.						
12	3	9	2	3	243	141	24
	Veículo 0: 0 > 1 > 2 > 3 > 8 Veículo 1: 0 > 4 > 5 > 8 Veículo 2: 0 > 6 > 7 > 8 Como era de esperar, não são suficientes dois veículos para fazer a recolha dado que temos 7 unidades para recolher e cada veículo apenas consegue comportar 3 unidades. Observa-se que são afectados 2 contentores a dois veículos, sendo afectados 3 pontos de recolha ao primeiro veículo.						

Tabela 4.1 - Resultados da aplicação do programa a diferentes casos simples (continuação).

Caso	Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função objectivo
13	3	11	2	9	363	193	10
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 6 > 7 > 8 > 9 > 10 Veículo 1: não usado Veículo 2: não usado Mais uma vez, é usado um veículo dado que este tem capacidade para comportar a carga de todos os contentores, sendo poupada a utilização do segundo e terceiro veículos.						
14	3	11	2	5	363	193	20
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 10 Veículo 1: 0 > 6 > 7 > 8 > 9 > 10 Veículo 2: não usado Não é suficiente um veículo para fazer a recolha. Existem 9 unidades para recolher e cada veículo apenas consegue comportar 5 unidades. Observa-se que são afectados 5 locais a um veículo, sendo afectados 4 pontos de recolha ao segundo veículo. O terceiro veículo não é usado.						
15	3	11	2	3	363	193	30
	Veículo 0: 0 > 1 > 2 > 3 > 10 Veículo 1: 0 > 4 > 5 > 6 > 10 Veículo 2: 0 > 7 > 8 > 9 > 10 Como seria de esperar, não são suficientes dois veículos para fazer a recolha dado que temos 9 unidades para recolher e cada veículo apenas consegue comportar 3 unidades. Observa-se que são afectados 3 locais a cada veículo.						

4.1.3 Comentários

Os exemplos apresentados neste capítulo apresentam resultados credíveis de acordo com as restrições impostas. Note-se que, além da optimização dos percursos, a aplicação optimiza o número de veículos a serem utilizados e, naturalmente, o número de pessoas envolvidas na recolha. Como se depreende, o crescimento do número de variáveis é exponencial com o crescimento do número de contentores e/ou veículos. Desta forma, o tempo de execução da aplicação também crescerá de uma forma exponencial.

No que se refere ao número de ligações, que é fixo e igual em cada ponto, este parâmetro poderia ser alterado e variar em cada ponto. Para tal bastava ter em consideração um maior número de variáveis e a necessidade de introdução de uma matriz de custos.

Outra restrição nesta versão é a carga máxima de cada veículo, que é fixa e igual para todos, além de que se supõe que a quantidade de lixo em cada contentor é a mesma e de valor unitário. Estas questões podem também ser ultrapassadas introduzindo uma matriz de valores para as cargas.

4.2 Casos de estudo mais complexos

4.2.1 Introdução

Um dos principais inconvenientes da utilização da programação linear binária é o crescimento exponencial do número de variáveis e/ou equações, que poderá levar a um tempo de execução indesejável. A ultrapassagem deste problema tem por base as quantidades de RSU produzidas em diferentes zonas. Ou seja, sendo possível avaliar as concentrações de RSU da cidade e subdividir a área em estudo em sub-áreas, definidas por forma a que apenas seja necessário um veículo para efectuar a recolha em cada uma das sub-áreas, será possível afectar apenas um veículo e optimizar a sua recolha. Considere-se agora cada sub-área, os contentores que lhe estão afectos, as ligações possíveis entre elas, e os respectivos custos. Aplicando o método definido

anteriormente, obtêm-se qual ou quais a(s) sub-área(s) que fica(m) afecta(s) a cada veículo, bem como os veículos que não são utilizados.

Determinadas essas sub-áreas, basta otimizar o percurso de cada veículo na(s) sub-área(s) que lhe está/estão destinada(s), obtendo desta forma o melhor percurso para cada recolha.

Assim, usando super-contentores, está-se a reduzir o número de pontos que fazem parte do problema original, reduzindo assim o número de equações e de incógnitas.

4.2.2 Criação de sub-áreas

Considere-se uma área de recolha bem como a sua distribuição de RSU em todos os pontos de recolha. Divida-se essa zona em sub-áreas. A divisão poderá ser efectuada de várias formas:

- ✓ por freguesias;
- ✓ por áreas geográficas;
- ✓ por volume de resíduos;
- ✓ etc.

Considerem-se estas sub-áreas como sendo super-contentores (pontos de recolha) e as ligações possíveis entre esses pontos. Refaça-se o problema descrito anteriormente com a nova formulação.

Uma dedução importante que se pode retirar tem a ver com a redução drástica do número de equações e de incógnitas que fazem parte do problema inicial. Admita-se que, do problema inicial, faziam parte 10 000 contentores (pontos de recolha), incluindo origem e destino, e uma frota de 10 veículos. Significa então que o problema inicial possui 1 000 000 000 incógnitas e 100 200 020 equações. Feita a divisão em sub-áreas, obtendo por exemplo 50 super-contentores, a nova formulação passa a considerar 25 000 incógnitas e 3 520 equações.

4.2.3 Aplicação do algoritmo de programação binária

Considere-se uma zona de recolha e suponha-se que se dispõe de informação suficiente sobre a distribuição de RSU a serem recolhidos de forma a proceder-se a uma divisão em sub-áreas (super-contentores). Aplicando o algoritmo referido, obtêm-se como solução o(s) super-contentor(es) afectos a cada veículo de uma forma optimizada.

Aplicação 1

Considere-se que é conhecido o volume de RSU de modo a que se consegue fazer uma distribuição volumétrica, obtendo-se super-contentores com volume unitário para a recolha. Considere-se o exemplo onde são definidos 8 super-contentores (Figura 4.3).



Figura 4.3 - Distribuição do volume de RSU por super-contentores.

Mantendo válidas as suposições apresentadas na Secção 4.1.1, e aplicando o algoritmo a 8 super-contentores, obtêm-se os resultados seguintes:

Tabela 4.2 - Resultados da aplicação do programa aos super-contentores.

Caso	Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função
1	2	10	2	5	200	144	18
	Veículo 0: 0 > 1 > 2 > 3 > 4 > 5 > 9 Veículo 1: 0 > 6 > 7 > 8 > 9 Neste exemplo, não é suficiente um veículo para fazer a recolha dado que existem 8 unidades de recolha e cada veículo apenas consegue comportar 5 unidades. Observa-se que são afectados 5 super-contentores a um veículo, sendo afectados 3 a um segundo veículo.						
2	4	10	2	3	400	188	27
	Veículo 0: 0 > 1 > 2 > 3 > 9 Veículo 1: 0 > 4 > 5 > 6 > 9 Veículo 2: 0 > 7 > 8 > 9 Veículo 3: não usado Neste exemplo, não são suficientes dois veículos para fazer a recolha dado que se têm 8 unidades a recolher e cada veículo apenas consegue comportar 3 unidades. Observa-se que são afectados 3 locais a dois veículos, sendo afectados 2 pontos de recolha ao terceiro veículo, enquanto o quarto veículo não é usado.						

Como se pode ver, existe uma distribuição dos super-contentores pelos diversos veículos de acordo com a sua capacidade e optimizando a sua utilização.

Após a obtenção dos super-contentores afectados a cada veículo, o percurso óptimo associado a cada um destes é obtido usando um dos métodos descritos no Capítulo 3.

Aplicação 2

Considere-se o concelho de Gondomar e divida-se o concelho nas 12 freguesias, considerando o baixo concelho (citadino, logo com uma concentração de RSU maior) e o alto concelho (aldeia, logo uma concentração de RSU menor), tal como apresentado na Figura 4.4.

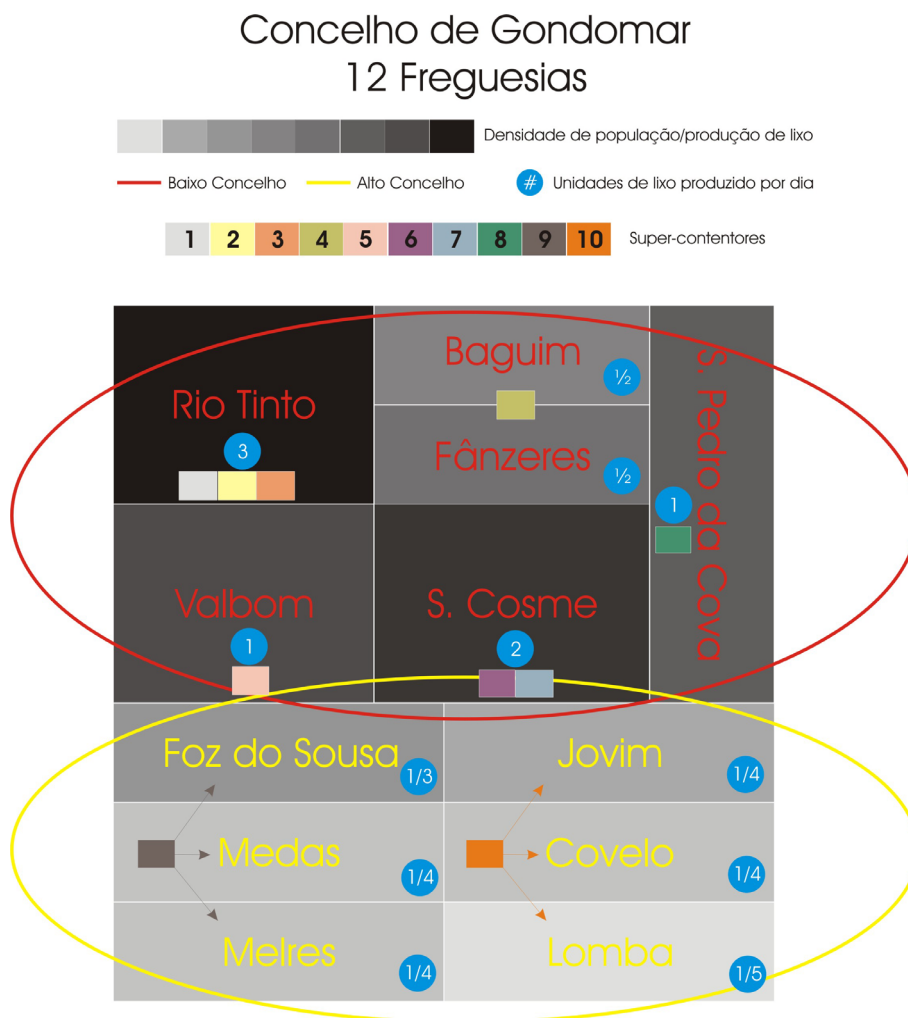


Figura 4.4 - Distribuição do volume de RSU por super-contentores no concelho de Gondomar.

Considere-se a produção hipotética de lixo em cada freguesia de acordo com a sua localização e densidade populacional em unidades de super-contentores. Admita-se ainda que o agrupamento de freguesias que não produzem uma unidade de super-contentor foi feita de acordo com a sua disposição geográfica.

Mantenham-se as suposições da Secção 4.1.1.

Aplicando o algoritmo implementado a este caso, 10 super-contentores, obtêm-se os resultados seguintes:

Tabela 4.3 - Resultados da aplicação informática aos super-contentores nas freguesias do concelho usando uma carga máxima de duas unidades.

Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função
5	12	2	2	720	274	55
Veículo 0: 0 > 1 > 2 > 11 Veículo 1: 0 > 3 > 4 > 11 Veículo 2: 0 > 5 > 6 > 11 Veículo 3: 0 > 7 > 8 > 11 Veículo 4: 0 > 9 > 10 > 11 Todos os veículos são afectados aos super-contentores de acordo com a restrição de carga de cada veículo.						

Dado que a carga máxima é de 2 super-contentores, pode-se concluir, analisando os resultados, que se obtém a seguinte distribuição de veículos:

- 0 - Zona 1 e 2 da freguesia de Rio Tinto;
- 1 - Zona 3 da freguesia de Rio Tinto, Baguim e Fânzeres;
- 2 - Valbom e Zona 1 da freguesia de S. Cosme;
- 3 - Zona 2 de S. Cosme, S. Pedro da Cova;
- 4 - Foz do Sousa, Medas, Melres, Jovim, Covelo e Lomba.

Tabela 4.4 - Resultados da aplicação informática aos super-contentores nas freguesias do concelho usando uma carga máxima de quatro unidades.

Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função
5	12	2	4	720	274	33
Veículo 0: 0 > 1 > 2 > 11 Veículo 1: 0 > 3 > 4 > 5 > 6 > 11 Veículo 2: 0 > 7 > 8 > 9 > 10 > 11 Veículo 3: não usado Veículo 4: não usado Apenas 3 dos 5 veículos são afectados aos super-contentores de acordo com a restrição de carga de cada veículo.						

Dado que a carga máxima é de 4 super-contentores, pode-se concluir, analisando os resultados, que se obtém a seguinte distribuição de veículos:

- 0 - Zona 1 e 2 da freguesia de Rio Tinto;
- 1 - Zona 3 de Rio Tinto, Baguim, Fânzeres, Valbom, Zona 1 de S. Cosme;
- 2 - Zona 2 de S. Cosme, S. Pedro da Cova, Foz do Sousa, Medas, Melres, Jovim, Covelo e Lomba;
- 3 - Não é usado;
- 4 - Não é usado.

Note-se que estes exemplos foram desenvolvidos admitindo que, a partir de cada super-contentor (i), existe ligação a dois super-contentores ($i+1$ e $i+2$).

Observe-se agora o que acontece se cada super-contentor (i) tiver apenas ligação ao próximo contentor ($i+1$).

Tabela 4.5 - Resultados da aplicação informática aos super-contentores nas freguesias do concelho usando uma carga máxima de duas unidades e apenas uma ligação possível entre os super-contentores.

Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função
5	12	1	2	720	274	55
Veículo 0: 0 > 1 > 2 > 11 Veículo 1: 0 > 3 > 4 > 11 Veículo 2: 0 > 5 > 6 > 11 Veículo 3: 0 > 7 > 8 > 11 Veículo 4: 0 > 9 > 10 > 11 Todos os veículos são afectados aos super-contentores de acordo com a restrição de carga de cada veículo e do percurso.						

Dado que a carga máxima é de 2 super-contentores, pode-se concluir, analisando os resultados, que se obtém a seguinte distribuição de veículos:

- 0 - Zona 1 e 2 da freguesia de Rio Tinto;
- 1 - Zona 3 da freguesia de Rio Tinto, Baguim e Fânzeres;
- 2 - Valbom e Zona 1 da freguesia de S. Cosme;
- 4 - Zona 2 da Freguesia de S. Cosme e S. Pedro da Cova;
- 5 - Foz do Sousa, Medas, Melres, Jovim, Covelo e Lomba.

Tabela 4.6 - Resultados da aplicação do programa aos super-contentores nas freguesias do concelho usando uma carga máxima de quatro unidades e apenas uma ligação possível entre os super-contentores.

Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função
5	12	1	4	720	274	33
Veículo 0: 0 > 1 > 2 > 3 > 4 > 11 Veículo 1: 0 > 5 > 6 > 7 > 8 > 11 Veículo 2: 0 > 9 > 10 > 11 Veículo 3: não usado Veículo 4: não usado Apenas 3 dos 5 veículos são afectados aos super-contentores de acordo com a restrição de carga de cada veículo.						

Dado que a carga máxima é de 4 super-contentores, pode concluir-se, analisando os resultados, que se obtém a seguinte distribuição de veículos:

- 0 - Freguesia de Rio Tinto, Baguim e Fânzeres;
- 1 - Valbom, freguesia de S. Cosme e S. Pedro da Cova;
- 2 - Foz do Sousa, Medas, Melres, Jovim, Covelo e Lomba;
- 3 - Não é usado;
- 4 - Não é usado.

4.2.4 Aplicação dum método para a resolução do problema do caixeiro viajante

A optimização do percurso de recolha para cada veículo, após a afectação das respectivas zonas, pode ser efectuada usando um dos métodos para a obtenção do caminho mínimo do caixeiro viajante. A aplicação dum dos métodos do caixeiro viajante ao percurso de cada veículo pode ser vista como uma matriz referência a

cada ponto de recolha que se distingue pela sua posição (x, y) , num plano de referência. Por exemplo, considere-se que a distribuição dos pontos de recolha dos super-contentores afectados ao veículo número 1 do caso anterior é a representada no plano da Figura 4.5.



Figura 4.5 - Distribuição dos pontos de recolha para o veículo 1.

Considere-se ainda que algumas das ligações possíveis entre os vários pontos de recolha são as representadas na Figura 4.6.

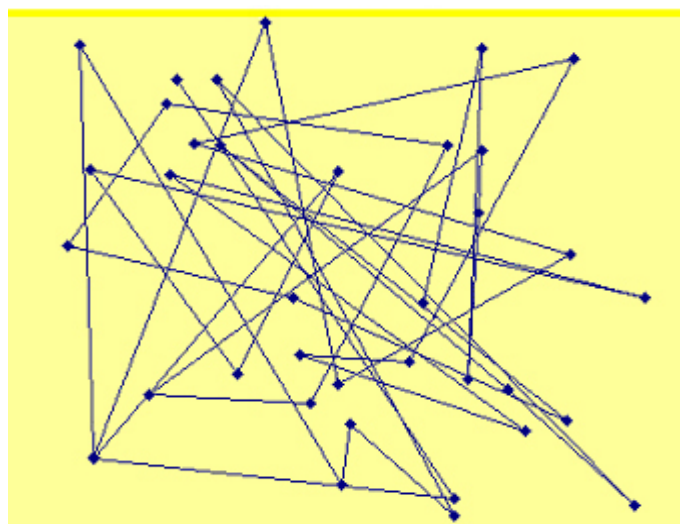


Figura 4.6 - Representação gráfica de alguns percursos entre os vários pontos de recolha.

Um dos métodos implementados para a resolução do problema do caixeiro viajante foi o recozimento simulado que, aplicado directamente a este exemplo, produz o resultado apresentado na Figura 4.7.



Figura 4.7 - Percurso optimizado do veículo 1 na recolha dos super-contentores que lhe foram afectados.

Desta forma é possível obter o percurso óptimo do veículo 1 ao longo dos super-contentores que lhe tinham sido afectados anteriormente.

Na prática, a repetição deste processo a todos os veículos em utilização, permite obter os percursos óptimos para a nossa frota.

4.2.5 Conclusão

Desta forma, usando uma combinação da programação linear binária e um dos métodos para a resolução do problema do caixeiro viajante (CV), pode resolver-se o problema em causa, ou seja é possível ultrapassar o problema do tempo de execução dum aplicação directa dum método de optimização. A programação linear binária é usada para obter a distribuição das cargas pelos vários veículos, optimizando-se o percurso de cada veículo usando um dos métodos aplicável à resolução do problema do CV.

Note-se que o que esta combinação permite fazer não é mais que transformar, usando programação linear binária, um problema de múltiplo caixeiro viajante (m veículos) em m problemas de caixeiro viajante directos (tantos quantos o número de veículos da frota a serem usados).

4.3 Método alternativo usando apenas programação linear binária

4.3.1 Introdução

Caso o problema seja demasiado grande, que apenas a distribuição por super-contentores não seja suficiente para a sua resolução, pode alargar-se a metodologia dos super-contentores à existência de super-veículos, veículos estes que não são mais que a representação de vários veículos.

Desta forma, usando super-veículos, estamos a reduzir o número de veículos que compõem a frota, reduzindo assim o número de equações e de incógnitas que fazem parte do nosso sistema inicial.

Considere-se que do problema inicial fazem parte 10 000 contentores (pontos de recolha), incluindo origem e destino, e uma frota de 50 veículos. Significa então que o nosso problema inicial possui 5 000 000 000 incógnitas e 101 000 100 equações. Feita a divisão em sub-áreas, obtendo por exemplo 50 super-contentores, e agrupando os nossos veículos em 5 super veículos, a nova formulação passa a possuir 12 500 incógnitas e 3 010 equações.

4.3.2 Descrição do método

Desta forma considere-se o método descrito abaixo:

- 1º Considerar super-contentores, de acordo com a distribuição geográfica da quantidade de lixo a recolher;

- 2º Considerar super-veículos (representam o somatório de n veículos), de acordo com os super-contentores definidos no ponto anterior;
- 3º Aplicar a programação linear binária, obtendo desta forma a distribuição dos super-contentores pelos super-veículos;
- 4º Para cada super-veículo, aplicar a programação linear binária aos veículos que este representa, e aos respectivos contentores que representam o(s) super contentor(es) que lhe está/estão afectado(s).

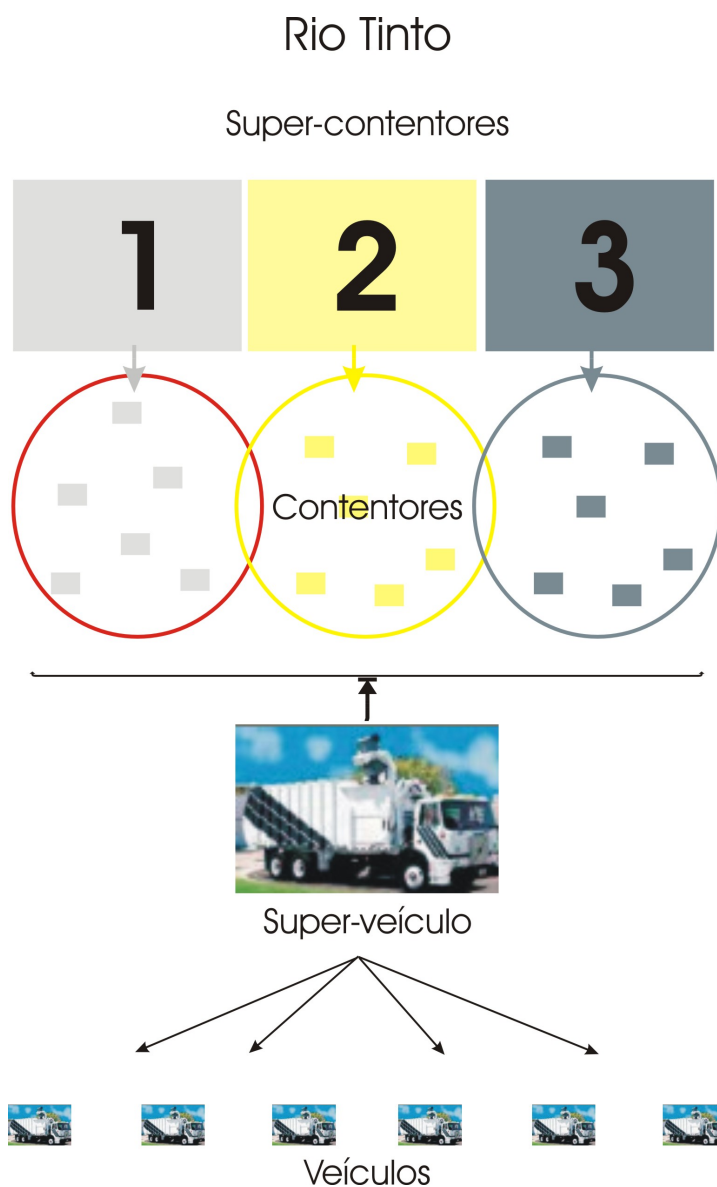


Figura 4.8 - Exemplo de formação de super-contentores e super-veículos.

4.3.3 Aplicação do método

Considere-se o exemplo do Concelho de Gondomar, composto pelas suas 12 freguesias (Figura 4.2). Neste método, cada veículo apenas transporta $\frac{1}{4}$ de unidade do super-contentor, ou seja para carregar um super-contentor, que corresponde a uma unidade, serão necessários 4 veículos. Considerem-se, por exemplo, super-veículos correspondentes a duas unidades de super-contentores (cada super-veículo representa 8 veículos).

Nestas condições, obtêm-se 10 super-contentores unitários para recolher. Se cada veículo comporta apenas $\frac{1}{4}$ da unidade de recolha, será necessária uma frota de 40 veículos. Agrupando os 40 veículos em super-veículos que comportem 2 unidades de carga, impõe-se o recurso a uma frota de 5 super-veículos.

Ou seja, com 10 super-contentores, 5 super-veículos, e supondo a existência de duas ligações a cada um (i liga com $i+1$ e $i+2$), e uma carga máxima de cada super-veículo de duas unidades, a aplicação do algoritmo de programação linear binária conduz, como primeiro resultado, aos dados apresentados na tabela abaixo.

Tabela 4.7 - Resultados da aplicação do programa aos super-contentores e super-veículos nas freguesias do concelho de Gondomar.

Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função
5	12	2	2	720	274	55
Super - veículo 0: 0 > 1 > 2 > 11 Super - veículo 1: 0 > 3 > 4 > 11 Super - veículo 2: 0 > 5 > 6 > 11 Super - veículo 3: 0 > 7 > 8 > 11 Super - veículo 4: 0 > 9 > 10 > 11 Todos os super-veículos são afectados aos super-contentores de acordo com a restrição de carga de cada super-veículo e do percurso.						

Dado que a carga máxima é de 2 super-contentores, pode-se inferir, analisando os resultados, a seguinte distribuição de super-veículos:

- 0 - Zonas 1 e 2 da freguesia de Rio Tinto;
- 1 - Zona 3 da freguesia de Rio Tinto, Baguim e Fânzeres;
- 2 - Valbom e Zona 1 da freguesia de S. Cosme;
- 3 - Zona 2 de S. Cosme, Foz do Sousa, Medas e Melres;
- 4 - S. Pedro da Cova, Jovim, Covelo e Lomba.

Desta forma, transforma-se o problema inicial em 5 sub-problemas de menores dimensões, aos quais se pode aplicar novamente o método de programação linear binária. Aplicando recursivamente o método de programação linear binária, transforma-se o problema inicial de grande dimensão em n problemas de dimensão mais reduzida, o que conduz a uma solução muito mais rápida, embora exigindo um trabalho de distribuição inicial por super-contentores e super-veículos.

Considere-se agora o super-veículo 0, que tem afectado os super-contentores das zonas 1 e 2 de Rio Tinto, tendo em linha de conta os vários contentores que compõem os dois super-contentores, e, por exemplo, quatro veículos que lhe estão afectados. Por uma questão de simplificação, vamos admitir que cada zona possui 12 contentores, numerados de 1 a 12, sendo o zero (0) o ponto de partida, e D (13) o ponto de chegada dos veículos em causa. Vai-se ainda supor que os contentores são de igual capacidade e que cada veículo comporta 3 contentores.

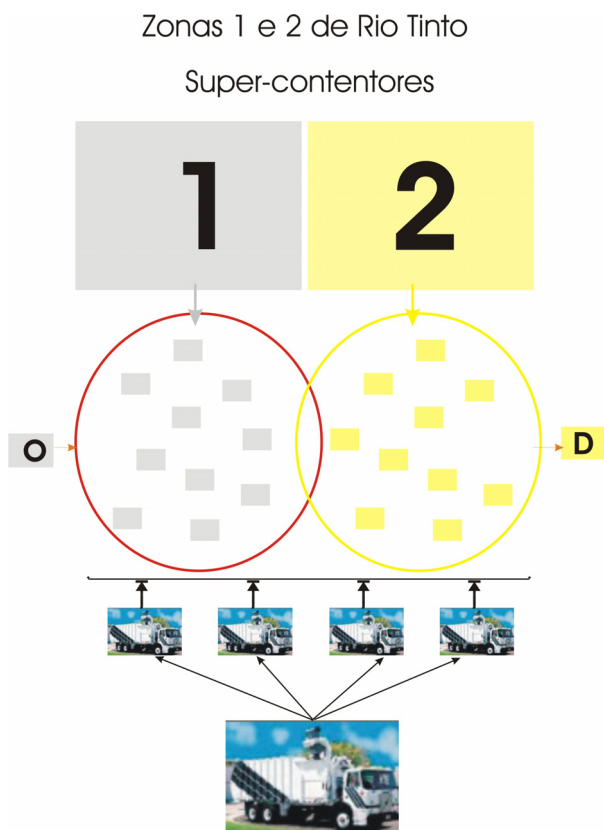


Figura 4.9 - Distribuição dos pontos de recolha dos super-contentores 1 e 2 afectos ao super-veículo 0.

Aplicando o algoritmo, obtêm-se os resultados da Tabela 4.8.

Tabela 4.8 - Resultados da aplicação do programa aos contentores correspondentes aos super-contentores 1 e 2.

Nº de veículos	Nº de locais	Nº de ligações	Carga máxima	Nº de variáveis	Nº de equações	Valor função
4	14	2	3	784	316	52
Veículo 0: 0 > 1 > 2 > 3 > 13						
Veículo 1: 0 > 4 > 5 > 6 > 13						
Veículo 2: 0 > 7 > 8 > 9 > 13						
Veículo 3: 0 > 10 > 11 > 12 > 13						
Todos os veículos são afectados aos contentores de acordo com a restrição de carga de cada veículo e do percurso.						

Dado que a carga máxima é de 3 contentores, pode concluir-se, analisando os resultados, que se obtém a seguinte distribuição de veículos:

- 0 - Contentores 1, 2 e 3 da zona 1;
- 1 - Contentores 4, 5 e 6 da zona 1;
- 2 - Contentores 7, 8 e 9 da zona 2;
- 3 - Contentores 10, 11 e 12 da zona 2.

4.3.4 Conclusão

Apresentou-se assim uma forma de utilização recursiva do programa que recorre ao método de programação linear binária com o objectivo de reduzir o tempo de execução da aplicação informática e obter dados concretos sobre a optimização dos circuitos de recolha. Ou seja, partindo dum problema global envolvendo super-contentores e super-veículos, de acordo com a densidade populacional e quantidades de RSU, obtém-se aplicando o algoritmo, sub-problemas de menor dimensão de mais rápida resolução, reduzindo a área geográfica de optimização bem como o número de veículos envolvidos. Transforma-se um problema de dimensão n em m problemas de dimensão reduzida.

De modo análogo ao referido na Secção 4.2, este método apresentado transforma um problema de múltiplo caixeiro viajante (m veículos e n locais) em m_1 problemas de múltiplo caixeiro viajante (de acordo com o número de super-contentores e super-veículos).

É de realçar que a aplicação recursiva do método pode levar à resolução de múltiplos problemas de caixeiro viajante, onde é possível aplicar um método simples para a resolução do problema directo, tal como foi descrito na Secção 4.2.

Capítulo 5 - Aplicação Informática MVBAGIS e Caso de Estudo Real

Sumário do Capítulo

Este capítulo apresenta as principais características da aplicação informática desenvolvida no âmbito desta tese que incorpora a programação linear binária, o método da inserção mais próxima e as funcionalidades de informação geográfica. Abordam-se as potencialidades desta ferramenta através da sua aplicação ao programa Ecofone (Serviço Gratuito de Recolha Selectiva de Resíduos) da LIPOR. É ainda apresentada a importância do conceito de super-contentor, referido no Capítulo 4, com aplicação neste caso de estudo real.

5.1 Aplicação informática MVBAGIS

Esta aplicação, desenvolvida em *Visual Basic*, usa a biblioteca de rotinas *MapObjects* da ESRI (Environmental Systems Research Institute, Inc.). Com vista a criar um interface que permita a introdução de dados necessários para executar um algoritmo de optimização de percursos e a apresentação dos resultados.

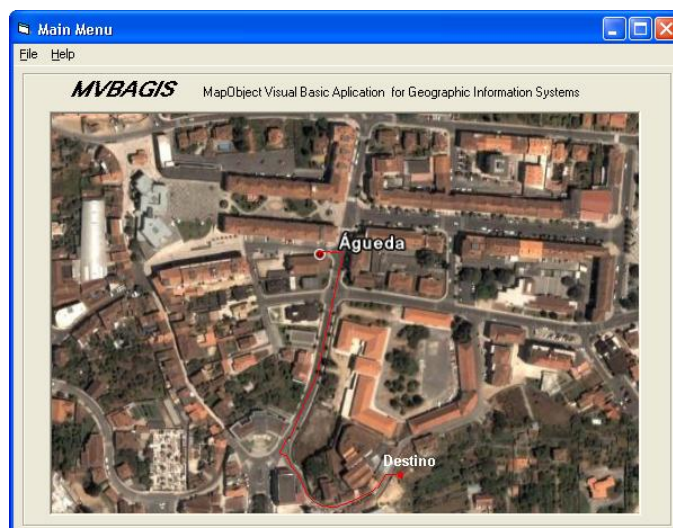


Figura 5.1 - Janela inicial da aplicação MVBAGIS.

A Figura 5.1 apresenta a janela inicial da aplicação informática. Foi desenvolvido um módulo que permite criar e alterar dados de um ficheiro *Shape* com informações geográficas e visuais de um mapa. Embora a aplicação permita a importação de ficheiros *Shape*, criados com outras aplicações, a informação existente nesse ficheiro não é, em muitos casos, compatível com a informação necessária à definição do problema da optimização de percursos.

Apresentam-se, de seguida, os passos necessários à criação ou reformulação de um ficheiro *Shape*:

Passo 1: Abrir um ficheiro de imagem com o mapa;

Passo 2: No caso de um ficheiro *Shape* já criado pela aplicação, saltar para o passo 4;

Passo 3: Introduzir dois pontos no mapa e a distância entre eles, obtendo-se desta forma a escala do mapa. Com dois pontos definidos pelo utilizador, pode calcular-se

a distância entre eles relacionando-a com a distância real atribuída pelo utilizador. Desta forma obtém-se a escala do mapa através de uma simples interpolação;

Passo 4: Traçar um segmento de uma rua (definido por pontos) partindo de um extremo da mesma até encontrar uma intersecção. Todos os segmentos ou ruas são definidos por linhas, as quais serão traçadas por pontos no mapa;

Passo 5: Quando se encontrar uma intersecção, é necessário introduzir as informações relativas ao segmento que se traçou: nome e largura da rua, sentido, proibições de trânsito;

Passo 6: Continuar a traçar a rua começando no último ponto definido ou seleccionando um ponto de intersecção existente, que se encontrará numa lista. Os pontos de intersecção são automaticamente criados para facilitar a construção do mapa a partir de uma intersecção anterior, ignorando-se a obrigatoriedade de seguir a construção de modo sequencial;

Passo 7: Saltar para o passo 4 até que o mapa fique concluído;

Passo 8: Guardar o ficheiro *Shape* num directório destino e com o nome escolhido;

A Figura 5.2 ilustra a janela de edição e de criação de um ficheiro *Shape* bem como a informação a introduzir para cada segmento de rua. A Figura 5.3 apresenta o fluxograma relativo à introdução dessa informação.

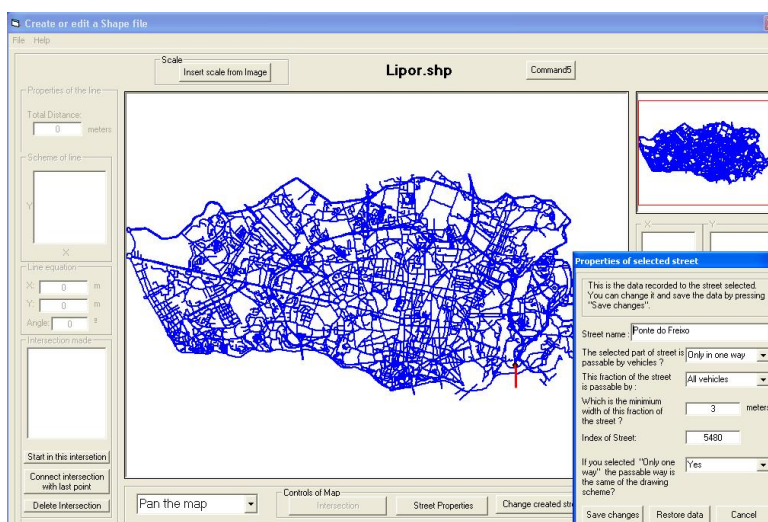


Figura 5.2 - Janela de edição e de criação de ficheiros *Shape*.

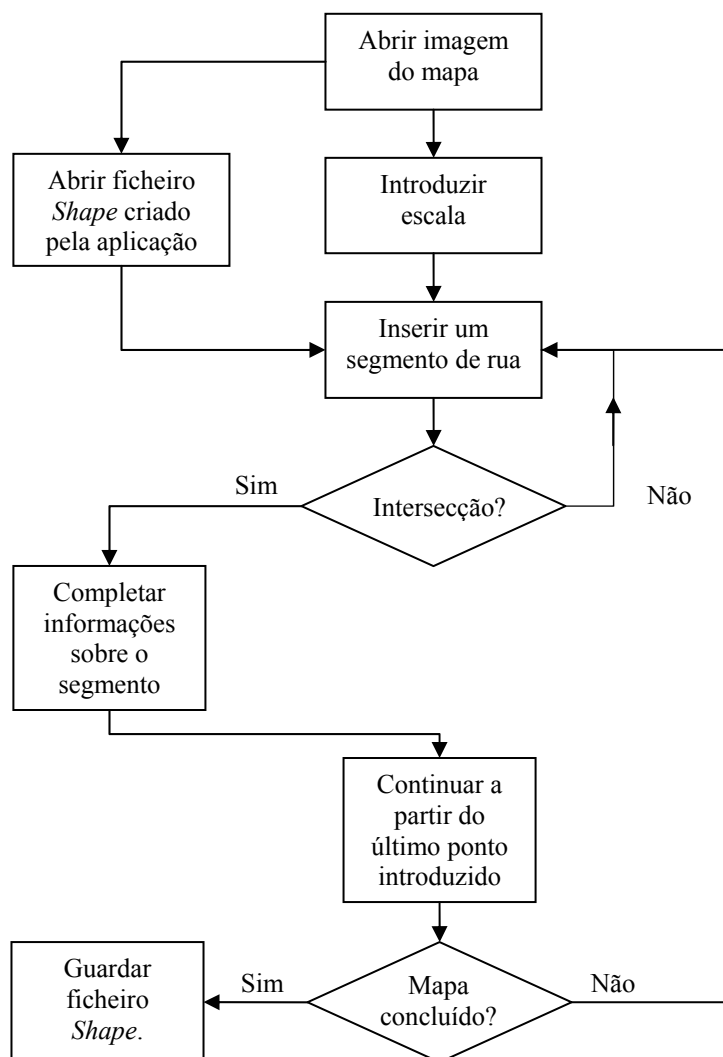


Figura 5.3 - Fluxograma relativo à edição e criação dum ficheiro *Shape*.

O ficheiro *Shape* do mapa será utilizado na definição do problema de optimização de percursos de recolha de resíduos sólidos urbanos, usando o algoritmo seguinte:

1. Abrir um ficheiro *Shape* do mapa;
2. Caso disponha de dados já guardados, passar ao passo 11;
3. Introduzir no mapa as localizações de pontos de partida, de chegada, e de recolha. Para cada ponto a ser introduzido, a aplicação verifica se encontrou

- de forma precisa o segmento de rua em que se encontra e a localização do ponto nesse segmento de rua. Enquanto o utilizador não preencher estas condições este passo será repetido (sendo o utilizador informado da condição que não respeitou) para o ponto a ser introduzido até que estas se verifiquem;
4. Face ao ponto introduzido no passo anterior introduzir a quantidade de RSU correspondente. De seguida enquanto todos os pontos não se encontrarem em memória, saltar para o passo 3;
 5. Introduzir informações relacionadas com consumo, número de homens, identificação dos veículos, dimensões e capacidade de cada um dos veículos disponíveis;
 6. Escolher se pretende efectuar a introdução de caminhos de forma manual (saltar para passo 7) ou automática (saltar para passo 11).
 7. Definir um caminho possível entre dois pontos, seleccionados pela aplicação, tendo em conta que este caminho seja contínuo no aspecto puramente físico (definam uma linha contínua entre os dois pontos). Para tal é necessário seleccionar um segmento de rua que contacte com o ponto/segmento anterior). Se esta condição não se verificar, o utilizador é informado e este passo é repetido.
 8. Esse caminho deverá estar de acordo com as leis de trânsito que foram atribuídas ao mapa quando este foi criado. Caso não se verifiquem estas condições, o utilizador é informado da situação e deverá seleccionar outro segmento que as verifique (repetindo o passo 7). Caso contrário repetir este passo;
 9. Após definir um caminho entre os dois pontos seleccionados, caso exista mais caminhos possíveis entre os mesmos pontos, o utilizador pode defini-los. Neste caso voltar ao passo 7. Caso o caminho introduzido já se encontre armazenado em memória, o caminho actual é apagado, o utilizador é informado e salta para o passo 7 onde deverá introduzir outro caminho válido se o desejar;

10. Se não existir mais nenhum caminho ou não o queira definir, a aplicação verifica após armazenar a informação em memória, se todas as combinações entre dois pontos já foram concluídas, o que significa que todos os caminhos possíveis já foram introduzidos (no caso particular de para o mesmo conjunto de pontos o utilizador introduzir mais que um caminho, a aplicação escolhe para a optimização, o que tiver menor distância). Se todos os dados já foram introduzidos, saltar para o passo 11, caso contrário, a aplicação define outros dois pontos baseando-se nas combinações necessárias para obter todos os caminhos possíveis, e salta para o passo 7;
11. A recolha desta informação serve de base à optimização dos percursos;
12. A escolha de uma pesquisa automática de caminhos permite definir o melhor caminho (menor distância atendendo aos requisitos impostos de continuidade e regras de trânsito) entre cada conjunto de pontos usando o algortimo de Floyd (descrito no Capítulo 3). Desta forma todos os dados necessários para a optimização são armazenados em memória de forma automática. Saltar para o passo 11.

Em termos de resultados finais, pode-se visualizar a informação relativa ao percurso de cada veículo, incluindo o movimento ao longo das ruas, pode-se ainda visualizar cada ponto de recolha e imprimir o relatório por veículo, com informação de trajecto e quantidades a recolher em cada ponto. Existe ainda a informação do custo por veículo bem como o custo total da recolha efectuada por todos os veículos. Estes custos dizem respeito ao custo do percurso, ao custo do veículo no percurso e ao custo referente à equipa adstrita a cada veículo.

Em qualquer momento, o utilizador poderá inserir, alterar ou remover pontos de recolha, bem como alterar as suas propriedades.

As Figuras 5.4, 5.5, 5.6 e 5.7 apresentam o fluxograma de inserção de caminhos, dados geográficos e logísticos, necessários para proceder à optimização.

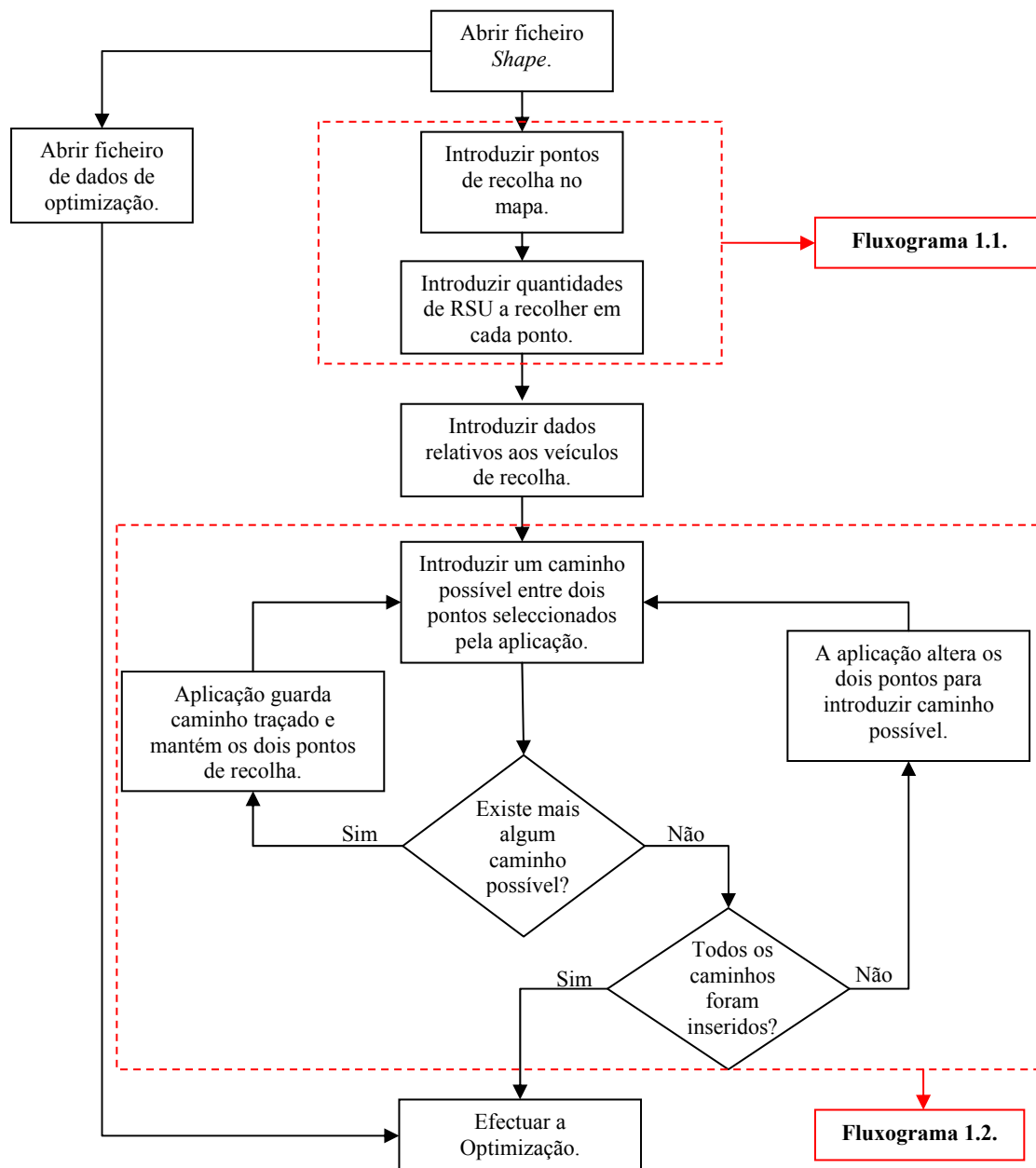


Figura 5.4 - Fluxograma de inserção de caminhos, dados geográficos e logísticos.

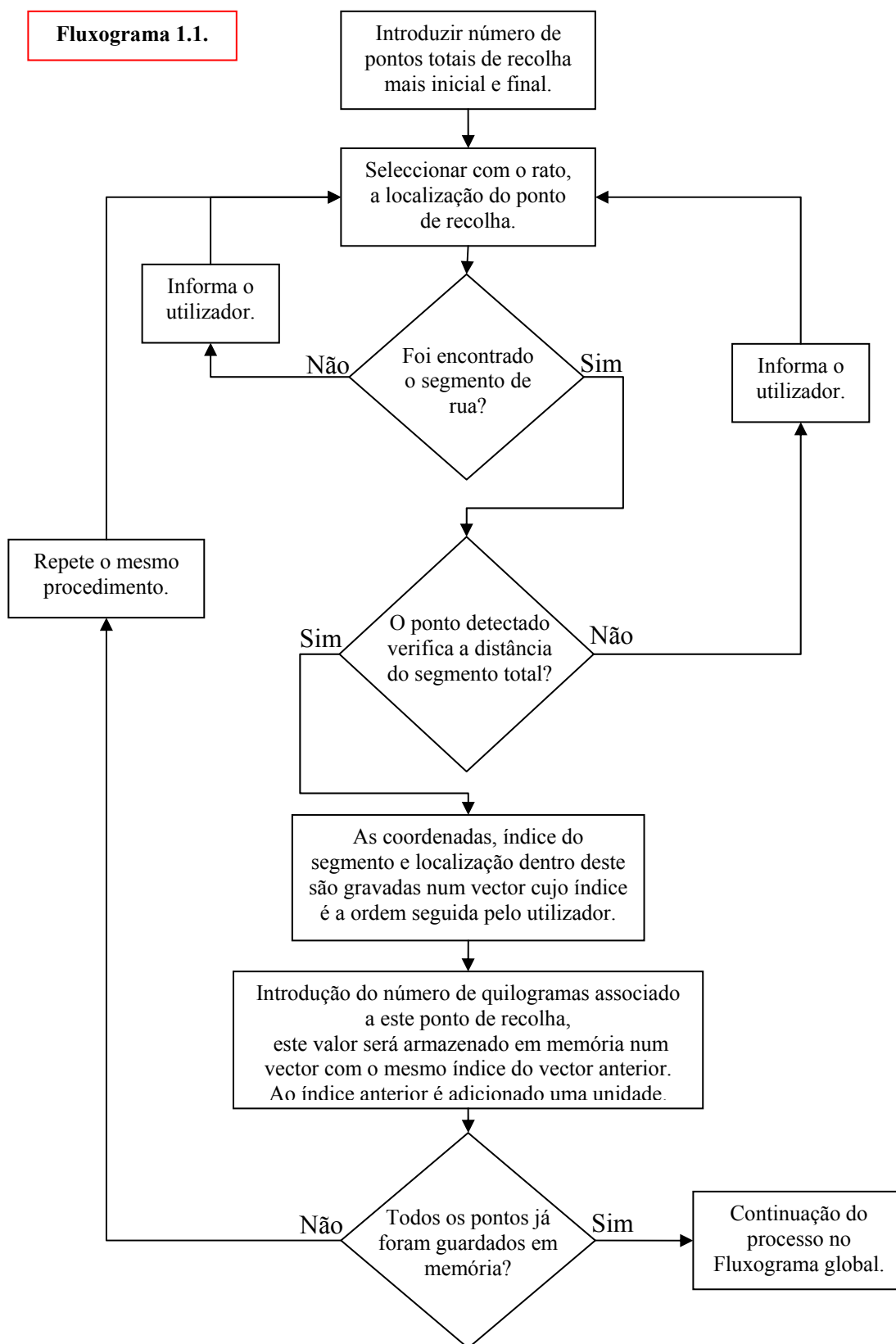


Figura 5.5 - Fluxograma 1.1 (parte do fluxograma da Figura 5.4).

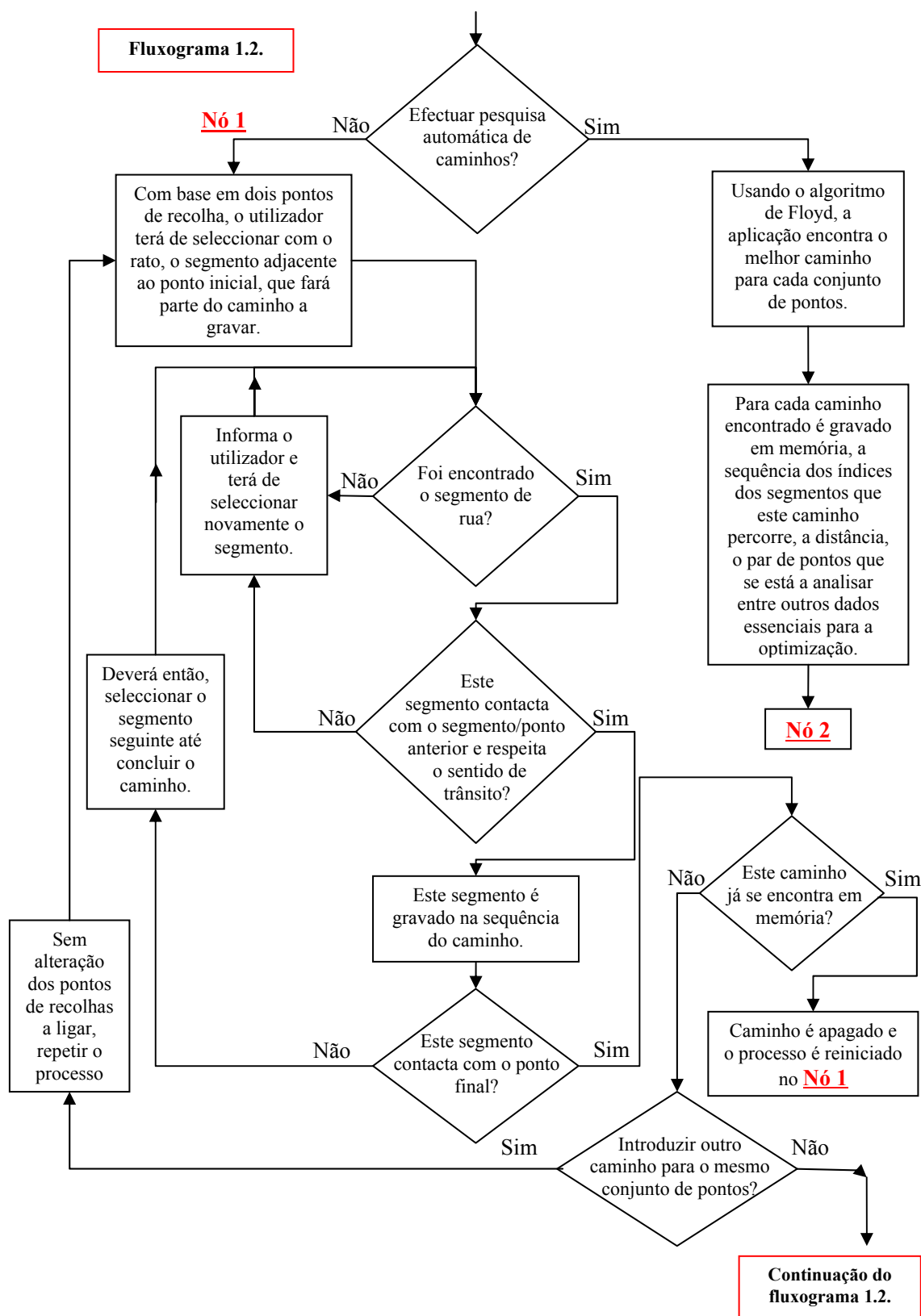


Figura 5.6 - Fluxograma 1.2 (parte do fluxograma da Figura 5.4).

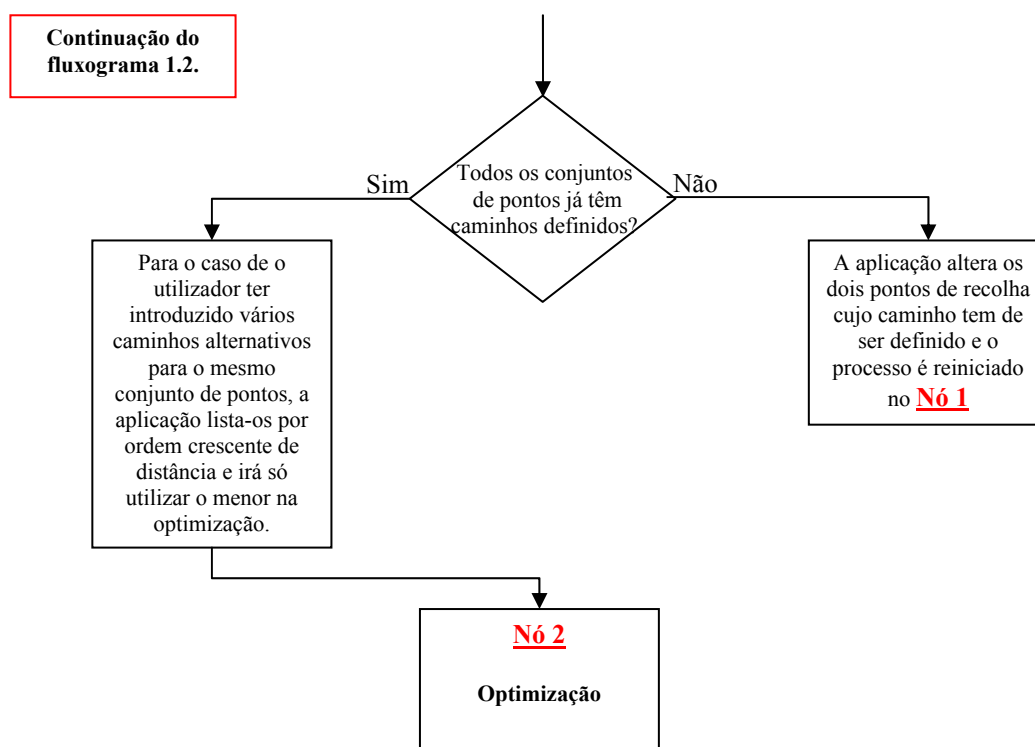


Figura 5.7 - Continuação do fluxograma 1.2 (parte do fluxograma da Figura 5.4).

As Figuras 5.8 e 5.9 apresentam as janelas de introdução da informação relacionada com as frotas e com os pontos de recolha, respectivamente.

The screenshot shows a window titled "Truck's data" with a close button in the top right corner. Inside the window, there is a section labeled "Truck's information" containing several input fields: "Truck Name:" with the value "XP-00-00", "Width (meters):" with the value "2", "Capacity (Kg):" with the value "1234", "Gas : (1L / 100 Km)" with the value "13", and "Number of men:" with the value "3". To the right of these fields is a list box containing two items: "XP-00-00" (which is selected) and "XP-00-01". At the bottom of the window, there are four buttons: "New", "Save", "Edit", and "Remove". Below these buttons are two more buttons: "Goto Main Menu" and "Cancel".

Figura 5.8 - Janela de introdução das características da frota.

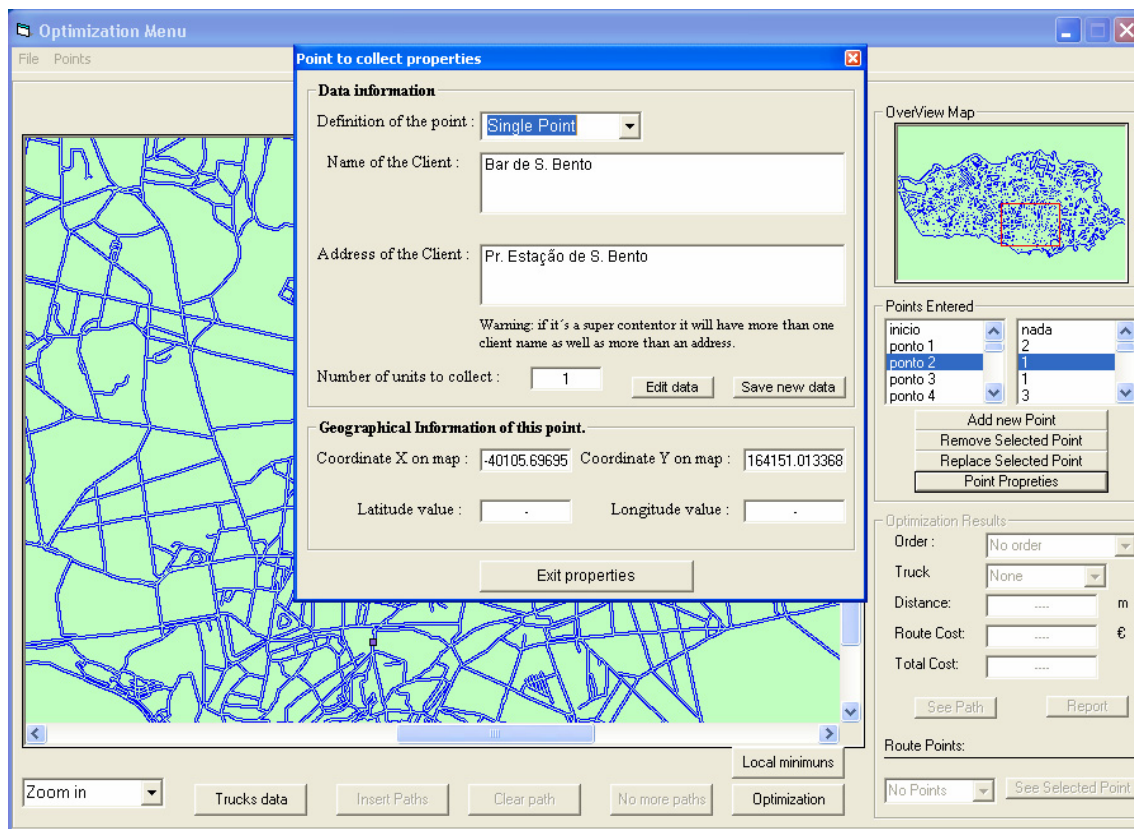


Figura 5.9 - Janela de introdução das características dos pontos de recolha.

No caso da existência de mais do que um ponto de recolha num único segmento de rua, os pontos de recolha aglomeram-se num ponto, designado por super-contentor, que corresponderá a um único ponto de recolha com a quantidade de RSU correspondente ao somatório de todas as quantidades de resíduos sólidos existentes nos pontos de recolha desse segmento.

A Figura 5.10 exemplifica um ponto super-contentor. O ponto 4 é um super-contentor que aglomera 3 clientes e as suas quantidades de recolha.

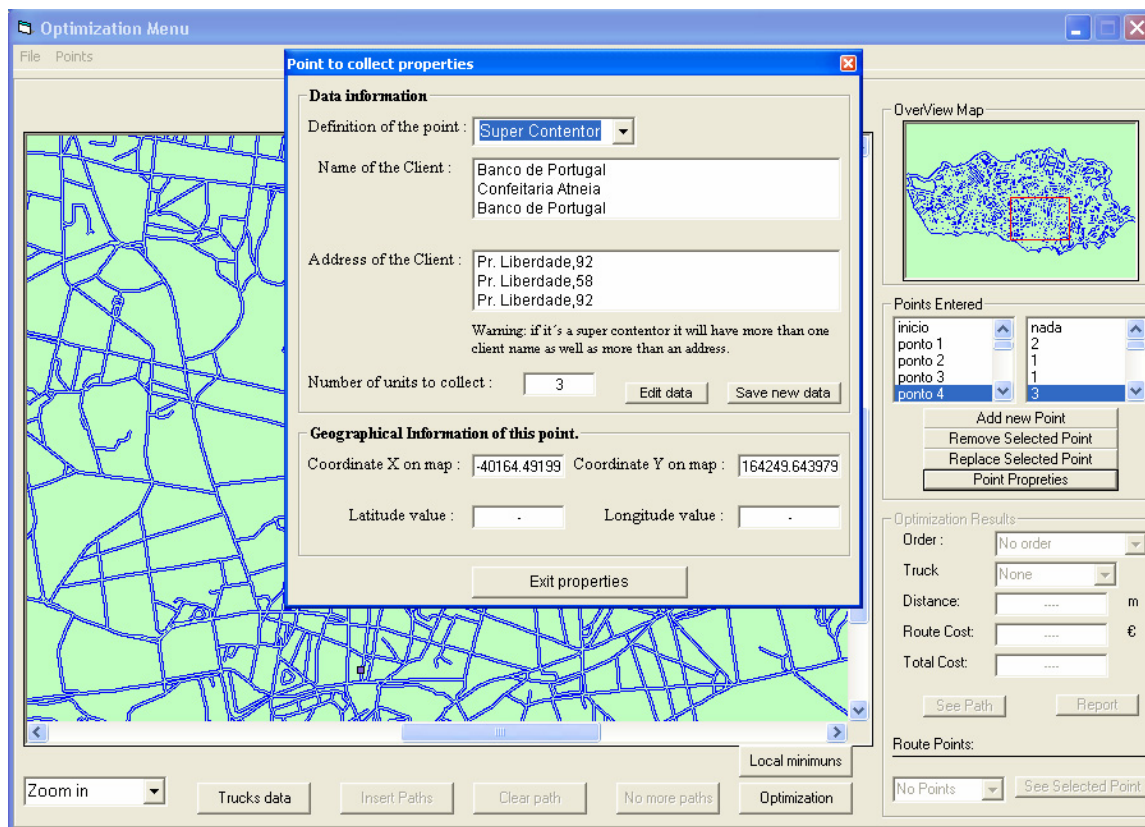


Figura 5.10 - Exemplo de um ponto super-contentor.

5.2 Aplicação ao caso do programa ECOFONE da LIPOR

A aplicação descrita anteriormente foi testada no programa ECOFONE da LIPOR, que se refere à recolha de resíduos sólidos em pontos específicos na área do grande Porto.

A área do grande Porto possui uma elevada densidade de segmentos de ruas. Assim, houve necessidade de desenvolver um novo módulo que permite encontrar automaticamente caminhos entre dois pontos de recolha quaisquer.

A determinação do caminho mais curto entre dois pontos segue as etapas seguintes:

- Definição de uma área de procura que engloba todos os pontos. Determina-se um raio de uma circunferência, em que o seu centro corresponde ao centroíde de todos os pontos, e cujo raio corresponde a 1,1 vezes a distância do ponto mais

afastado a esse centroíde. A Figura 5.11 ilustra a área de aplicação do algoritmo de Floyd para obtenção dos caminhos entre os quatro pontos definidos;

- A determinação do caminho mais curto entre cada dois pontos faz-se recorrendo ao algoritmo de Floyd (descrito no Capítulo 3) a partir de uma matriz de custos formada a partir das distâncias entre todos os pontos de ligação dos segmentos inseridos na área de procura. No caso de não existir ligação entre dois pontos, atribui-se um custo infinito (10^{100});
- Caso não se encontrem caminhos entre todos os pontos, o raio de pesquisa é aumentado de 1,1 vezes, voltando-se de seguida à etapa anterior até conseguir determinar as ligações entre todos os pontos.

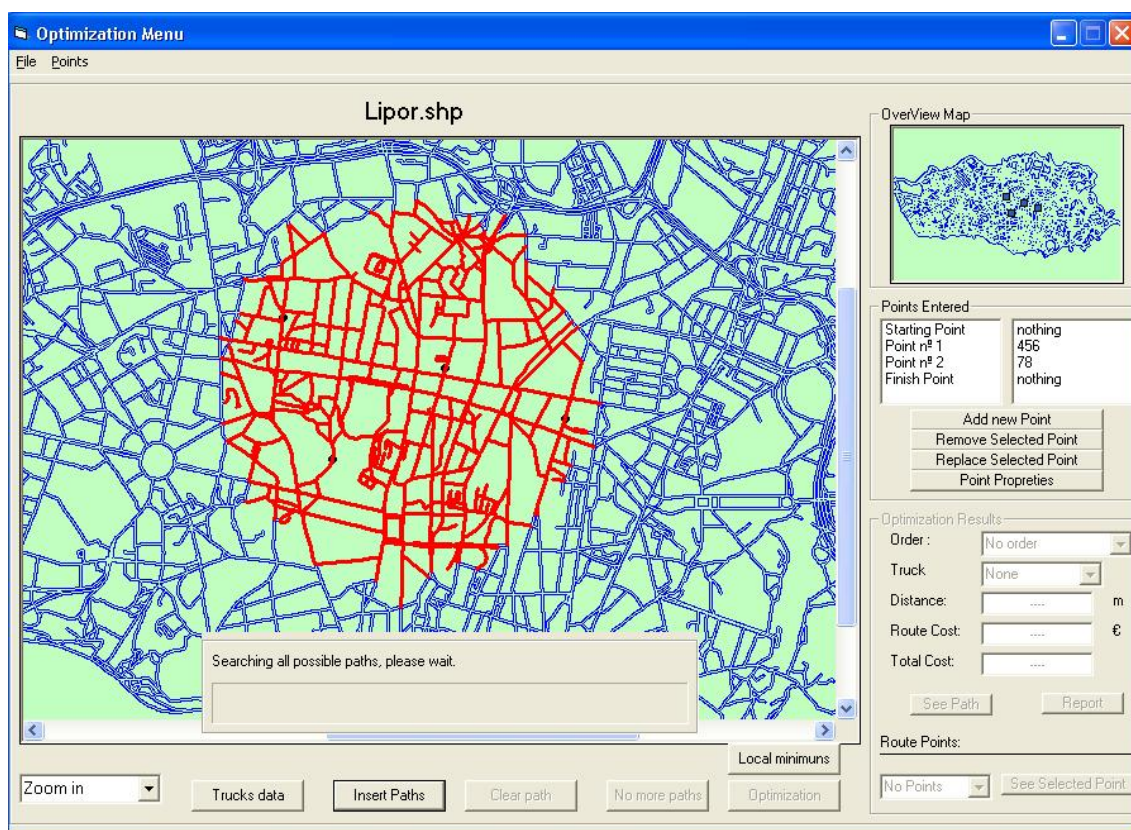


Figura 5.11 - Área de aplicação do algoritmo de Floyd para obtenção dos caminhos entre os quatro pontos definidos.

Seguidamente, apresentam-se os resultados dos circuitos otimizados de recolha de resíduos sólidos urbanos para o programa ECOFONE, com dez pontos de recolha e a utilização de um ou dois veículos.

Recolha com 1 veículo.

De seguida, apresentam-se os resultados, aplicando os algoritmos de optimização para a recolha de dez pontos, considerando apenas um veículo devido ao facto da sua capacidade ser suficiente para a recolha de todos os pontos.

A Figura 5.12, refere-se à janela de visualização dos resultados, onde se pode ver o mapa do percurso, o movimento do veículo ao longo deste e os custos inerentes.

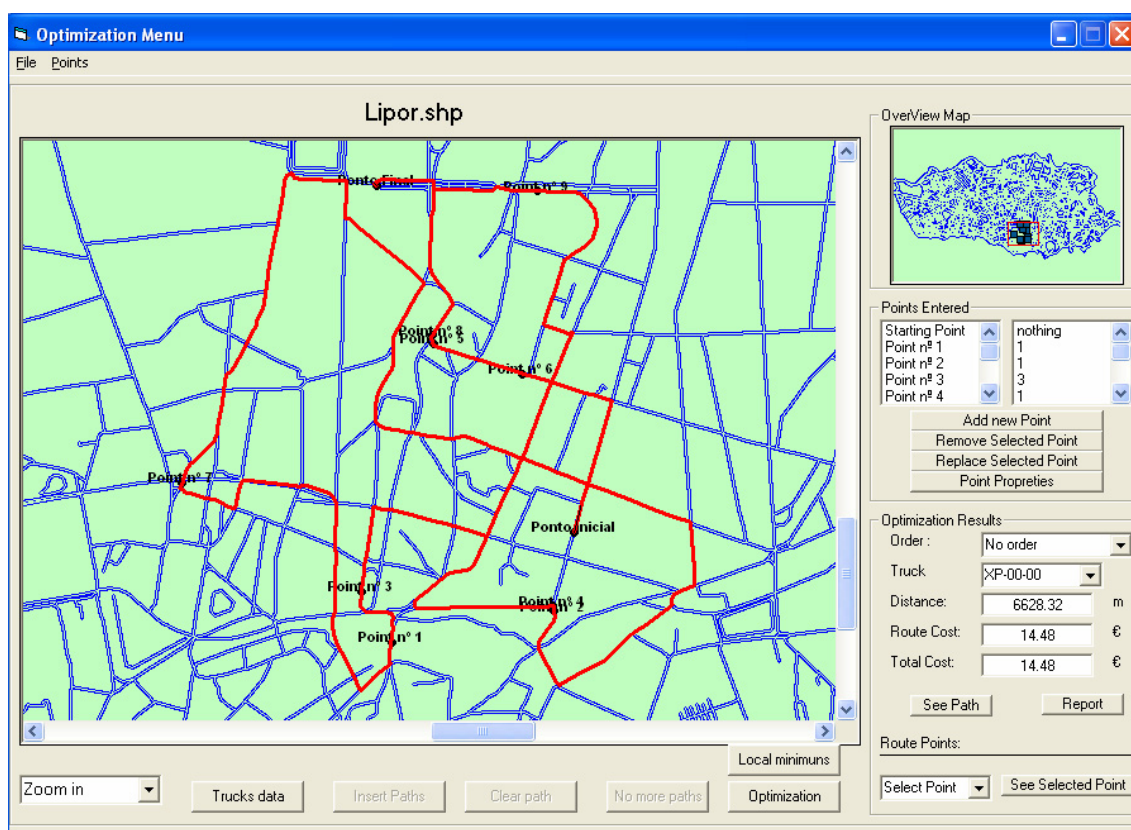


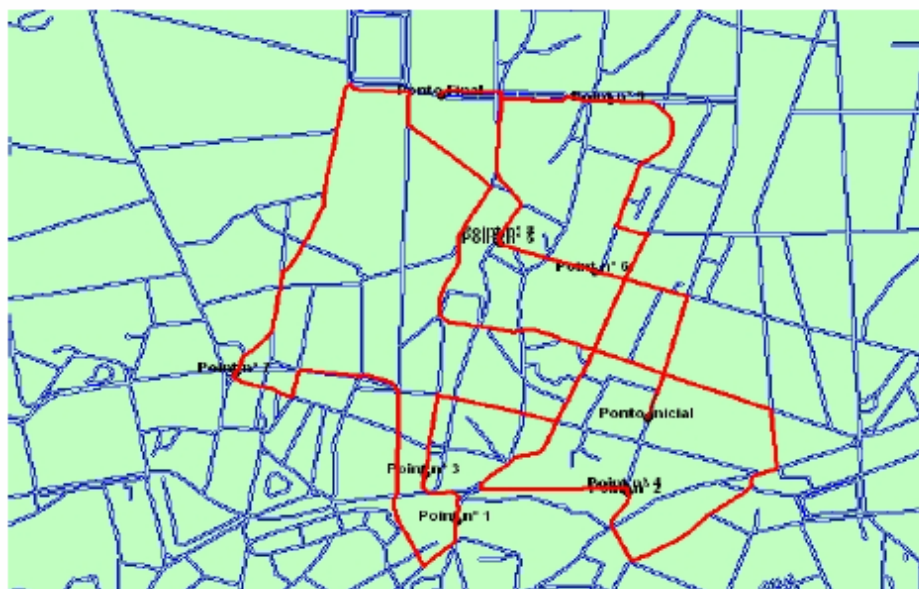
Figura 5.12 - Percurso do veículo XP-00-00, distância percorrida e respectivos custos.

A Figura 5.13, apresenta o relatório do trajecto optimizado e a listagem das unidades de RSU a recolher. Além do mapa do percurso, é ainda possível ver todos os pontos de recolha, passagem, partida e chegada, assim como as quantidades a recolher em cada ponto.

Trajecto optimizado e lista de recolhas

Veículo: XP-00-00

Imagem do trajecto a ser completado



Nota: O símbolo "#" encontrado ao lado de algumas secções do trajecto optimizado significa, nessa secção, o veículo deverá efectuar inversão de marcha.

Legenda

- Ponto de Partida
- Pontos de Recolha
- Ponto Final

Descrição do Percurso

Nome da Rua	Distância (Percorrida em m)	Designação	Unidades de recolha (caso existam)
Rua de Santa Catarina	56	Partida	Sem recolha de RSU
...			
Rua de Gonçalo Cristóvão	33.9	Ponto n.º 9	1 unidade(s) de recolha
...			
Praça da Batalha	84.6	Ponto n.º 2	1 unidade(s) de recolha
Rua de 31 de Janeiro	89	Ponto n.º 4	1 unidade(s) de recolha
Rua de Sá da Bandeira	15	Percurso	Sem Recolha de RSU
...			
Distância Total: (Km)		6.6	Unidade(s) Recolhida(s) Total: 12

Figura 5.13 - Relatório do percurso do veículo XP-00-00 e distância percorrida.

Recolha com 2 veículos.

Neste exemplo, a capacidade do primeiro veículo foi reduzida de forma a provocar uma recolha usando dois veículos.

As Figuras 5.14, 5.15, 5.16 e 5.17 apresentam os resultados obtidos aplicando os algoritmos para a recolha de dez pontos, e recorrendo a dois veículos devido ao facto da capacidade de um só não ser suficiente para a recolha de todos os pontos.

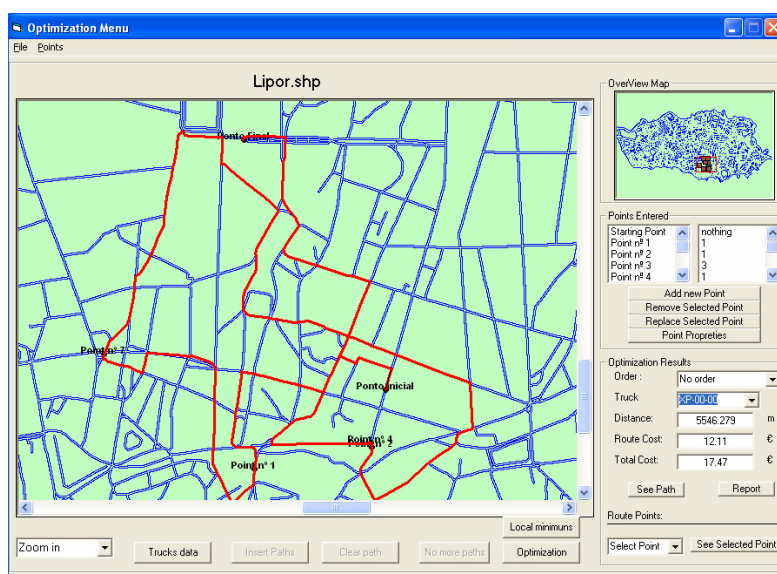


Figura 5.14 - Percurso do veículo XP-00-00, distância percorrida e respectivos custos.

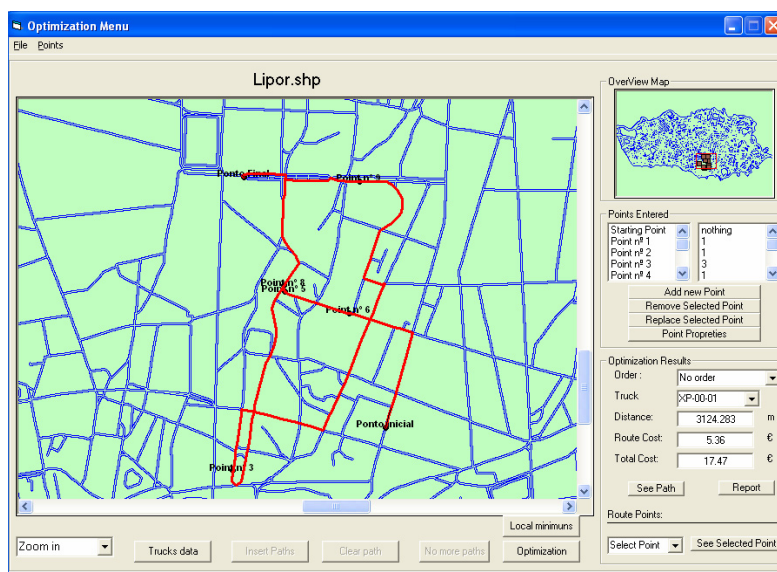
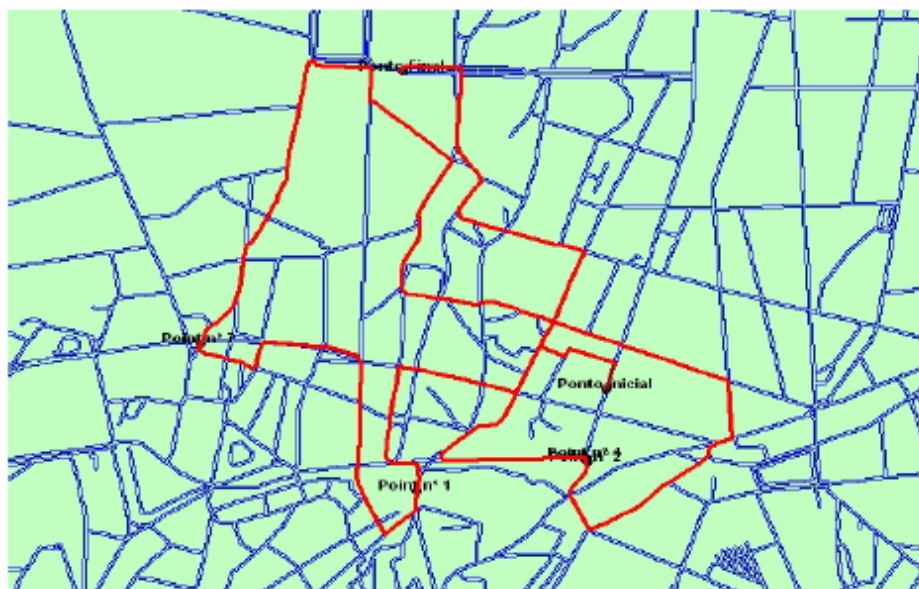


Figura 5.15 - Percurso do veículo XP-00-01, distância percorrida e respectivos custos.

Trajecto optimizado e lista de recolhas

Veículo: XP-00-00

Imagem do trajecto a ser completado



Nota: O símbolo "#" encontrado ao lado de algumas secções do trajecto optimizado significa, nessa secção, o veículo deverá efectuar inversão de marcha.

Legenda

- Ponto de Partida
- Pontos de Recolha
- Ponto Final

Descrição do Percurso

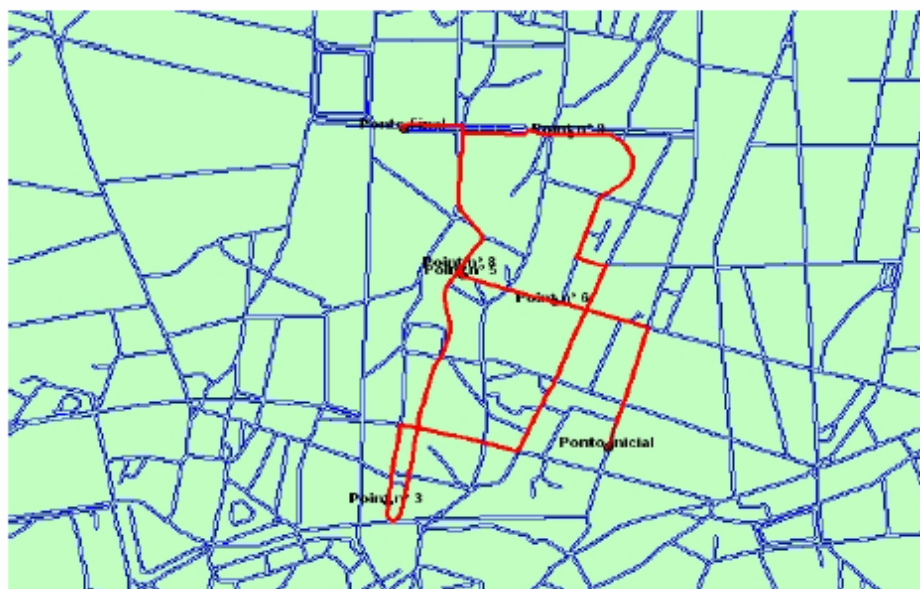
Nome da Rua	Distância (Percorrida em m)	Designação	Unidades de recolha (caso existam)
Rua de Santa Catarina	56	Partida	Sem recolha de RSU
...			
Praça de Almeida Garrett	39.2	Ponto nº 1	1 unidade(s) de recolha
Rua de Mouzinho da Silveira	79.7	Percurso	Sem Recolha de RSU
...			
Rua de 31 de Janeiro	89	Ponto nº 4	1 unidade(s) de recolha
...			
Rua de Gonçalo Cristóvão	6.6	Destino Final	Sem recolha de RSU
Distância Total: (Km)		5.5	Unidade(s) Recolhida(s) Total: 4

Figura 5.16 - Relatório do percurso do veículo XP-00-00 e distância percorrida.

Trajecto optimizado e lista de recolhas

Veículo: XP-00-01

Imagem do trajecto a ser completado



Nota: O símbolo "#" encontrado ao lado de algumas secções do trajecto optimizado significa, nessa secção, o veículo deverá efectuar inversão de marcha.

Legenda

- Ponto de Partida
- Pontos de Recolha
- Ponto Final

Descrição do Percurso

Nome da Rua	Distância (Percorrida em m)	Designação	Unidades de recolha (caso existam)
...			
Rua de Fernandes Tomás	56.9	Ponto nº 5	2 unidade(s) de recolha
Rua da Trindade	47.9	Ponto nº 8	1 unidade(s) de recolha
...			
Avenida dos Aliados	73.6	Percurso	Sem Recolha de RSU
Rua do Dr. António Luis Gomes	87.4	Percurso	Sem Recolha de RSU
...			
Distância Total: (Km)		3.1	Unidade(s) Recolhida(s) Total: 8

Figura 5.17 - Relatório do percurso do veículo XP-00-01 e distância percorrida.

Por último, testou-se a aplicação desenvolvida com todos os pontos de recolha de uma equipa da LIPOR do programa Ecofone. Alguns desses pontos são definidos como super-contentores uma vez que se encontravam no mesmo segmento de rua. A Tabela 5.1 apresenta a listagem dos pontos de recolha dessa equipa durante um dia.

Tabela 5.1 - Pontos de recolha do programa Ecofone de uma equipa durante um dia.

Nome	Morada	Sacos Recolher	Super Contentor
Confeitaria Império	Rua Santa Catarina, 149	2	s
Confeitaria Mengos	Rua Santa Catarina, 161	-	s
Bar de S. Bento	Praça Estação de S. Bento	1	n
Confeitaria Marbella	Rua Santa Catarina, 13	1	n
Banco de Portugal	Praça Liberdade, 92	3	s
Confeitaria Atneia	Praça Liberdade, 58	-	s
Confeitaria Barbarela	Rua 31 de Janeiro, 215	1	n
Confeitaria Santa Alexandrina	Rua Fernandes Tomás, 775	2	s
Confeitaria Capri	Rua Fernandes Tomás, 965	1	n
Pastelaria Cristo Rei	Rua Fernandes Tomás, 793	-	s
Café Progresso	Rua Actor João Guedes, 5	1	n
Banco de Portugal	Praça Liberdade, 92	-	s
Chafariz da Trindade	Rua Fernandes Tomás, 963	1	n
TSF	Rua Gonçalo Cristovão, 195	1	n
Conservatória do Registo do Porto	Rua Gonçalo Cristovão, 347	1	n
Governo Civil do Porto	Rua Gonçalo Cristovão, 373	1	n
Junta Freguesia de S. Ildefonso	Rua Gonçalo Cristovão, 187	1	n
Instituto Estradas de Portugal	Rua Camões, 219	1	n
Confeitaria Porto Trindade	Rua Camões, 89	1	n
Geestrada	Rua João das Regras, 285	2	s
Jardim de Infância nº1	Rua João das Regras, 305	-	s
Restaurante Madeirense	Rua Entreparedes, 36	1	n
Restaurante Irmãos Unidos	Rua Campinho, 13	1	n
FNAC	Rua Passos Manuel, 38	1	n
Jornal 1º de Janeiro	Rua Santa Catarina, 489	1	n
Adm. Regional Saúde do Norte	Rua Santa Catarina, 1288	1	n
Pizaria Meidin	Rua Santa Catarina, 949	1	n
Alessandro Nannini	Rua Firmeza, 476	2	s
Sublunar	Rua Firmeza, 364	-	s
Barbara Rangel	Rua Firmeza, 480 4ºEsq	1	n

Observação: Quando a quantidade a recolher não está definida, significa que o ponto de recolha está aglomerado no super-contentor anterior.

Os trinta pontos de recolha foram agrupados em vinte e quatro: cinco super-contentores e mais dezanove contentores. Os super-contentores correspondem à junção de contentores que se encontram no mesmo segmento de rua.

Recolha com 1 veículo.

De seguida, apresentam-se os resultados, aplicando os algoritmos de optimização para a recolha de vinte e quatro pontos, recorrendo a um único veículo devido à sua capacidade ser suficiente para a recolha de todos os pontos.

A Figura 5.18, refere-se à janela de visualização dos resultados, onde se pode ver o mapa do percurso, o movimento do veículo ao longo deste e os custos inerentes.

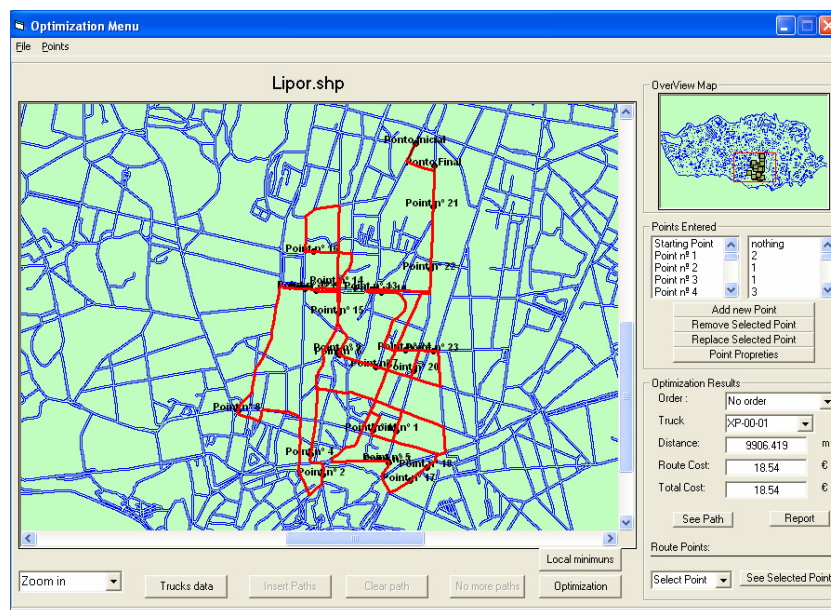


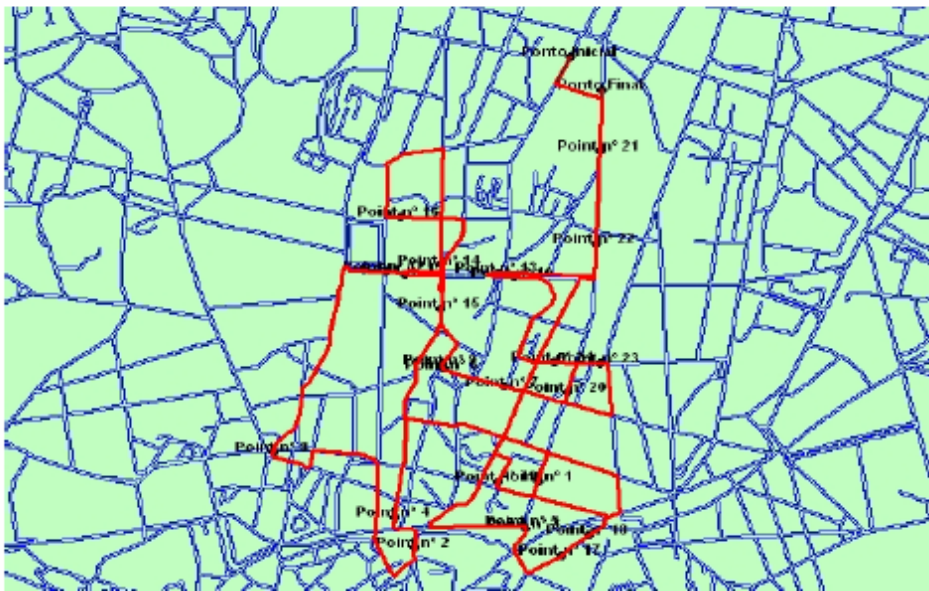
Figura 5.18 - Percurso do veículo XP-00-01, distância percorrida e respectivos custos.

A Figura 5.19, apresenta o relatório do trajecto optimizado e a listagem das unidades de RSU a recolher. Além do mapa do percurso, é ainda possível ver todos os pontos de recolha, passagem, partida e chegada, assim como as quantidades a recolher em cada ponto.

Trajecto optimizado e lista de recolhas

Veículo: XP-00-01

Imagem do trajecto a ser completado



Nota: O símbolo "#" encontrado ao lado de algumas secções do trajecto optimizado significa, nessa secção, o veículo deverá efectuar inversão de marcha.

Legenda

- Ponto de Partida
- Pontos de Recolha
- Ponto Final

Descrição do Percurso

Nome da Rua	Distância (Percorrida em m)	Designação	Unidades de recolha (caso existam)
Rua do Bonjardim	76.8	Partida	Sem recolha de RSU
...			
Praça da Batalha	84.6	Ponto nº 3	1 unidade(s) de recolha
Rua de 31 de Janeiro	89	Ponto nº 5	1 unidade(s) de recolha
Rua de Sá da Bandeira	15	Percurso	Sem Recolha de RSU
...			
Rua de Santa Catarina	26	Destino Final	Sem recolha de RSU
Distância Total: (Km)		9.9	Unidade(s) Recolhida(s) Total: 30

Figura 5.19 - Relatório do percurso do veículo XP-00-01 e distância percorrida.

Recolha com 2 veículos.

Neste exemplo, a capacidade do primeiro veículo foi novamente reduzida de forma a provocar uma recolha usando dois veículos.

As Figuras 5.20, 5.21, 5.22 e 5.23 apresentam os resultados obtidos aplicando os algoritmos para a recolha de vinte e quatro pontos, e recorrendo a dois veículos devido ao facto da capacidade de um só não ser suficiente para a recolha de todos os pontos.

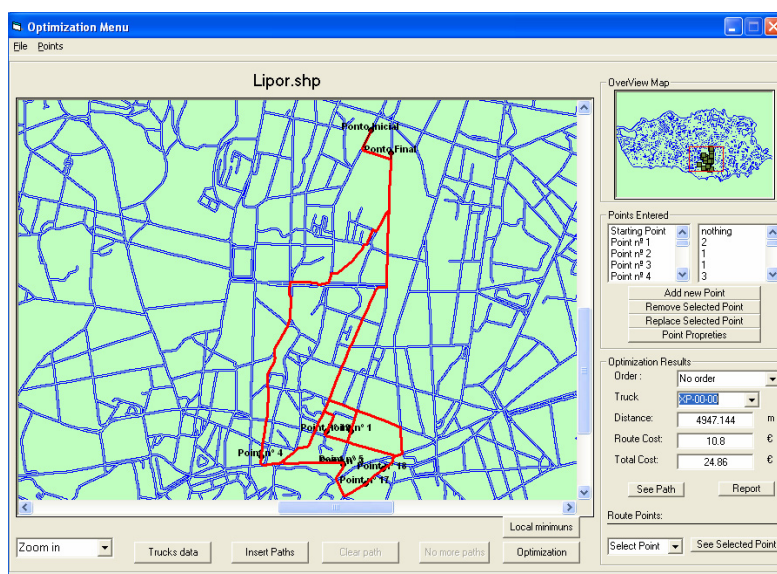


Figura 5.20 - Percurso do veículo XP-00-00, distância percorrida e respectivos custos.

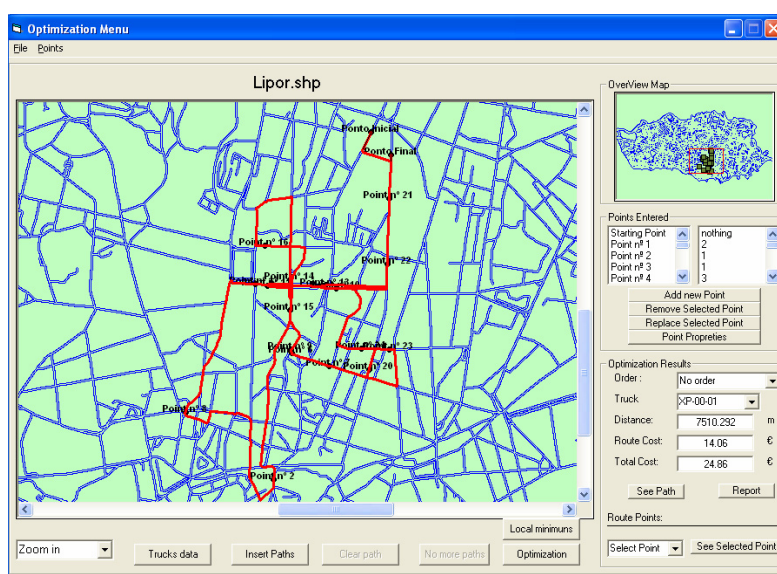
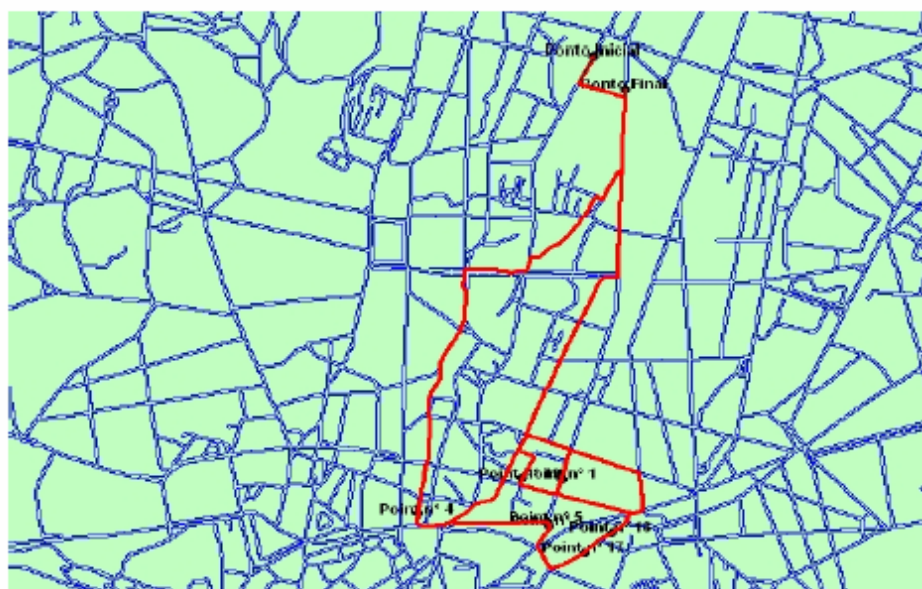


Figura 5.21 - Percurso do veículo XP-00-01, distância percorrida e respectivos custos.

Trajecto optimizado e lista de recolhas

Veículo: XP-00-00

Imagem do trajecto a ser completado



Nota: O símbolo "#" encontrado ao lado de algumas secções do trajecto optimizado significa, nessa secção, o veículo deverá efectuar inversão de marcha.

Legenda

- Ponto de Partida
- Pontos de Recolha
- Ponto Final

Descrição do Percurso

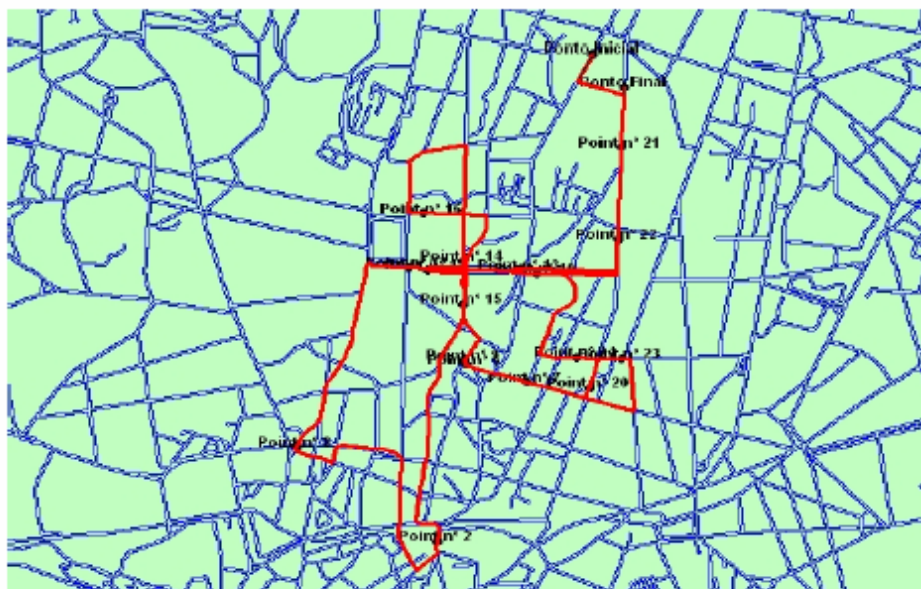
Nome da Rua	Distância (Percorrida em m)	Designação	Unidades de recolha (caso existam)
Rua do Bonjardim	76.8	Partida	Sem recolha de RSU
...			
Praça da Batalha	84.6	Ponto n° 3	1 unidade(s) de recolha
Rua de 31 de Janeiro	89	Ponto n° 5	1 unidade(s) de recolha
...			
Rua do Ateneu Comercial do Porto	106.6	Ponto n° 19	1 unidade(s) de recolha
Rua do Dr. António Emílio de Magalhães	49.9	Percurso	Sem Recolha de RSU
Rua de Sá da Bandeira	159.5	Percurso	Sem Recolha de RSU
...			
Distância Total: (Km)		4.9	Unidade(s) Recolhida(s) Total: 10

Figura 5.22 - Relatório do percurso do veículo XP-00-00 e distância percorrida.

Trajecto optimizado e lista de recolhas

Veículo: XP-00-01

Imagem do trajecto a ser completado



Nota: O símbolo "#" encontrado ao lado de algumas secções do trajecto optimizado significa, nessa secção, o veículo deverá efectuar inversão de marcha.

Legenda

- Ponto de Partida
- Pontos de Recolha
- Ponto Final

Descrição do Percurso

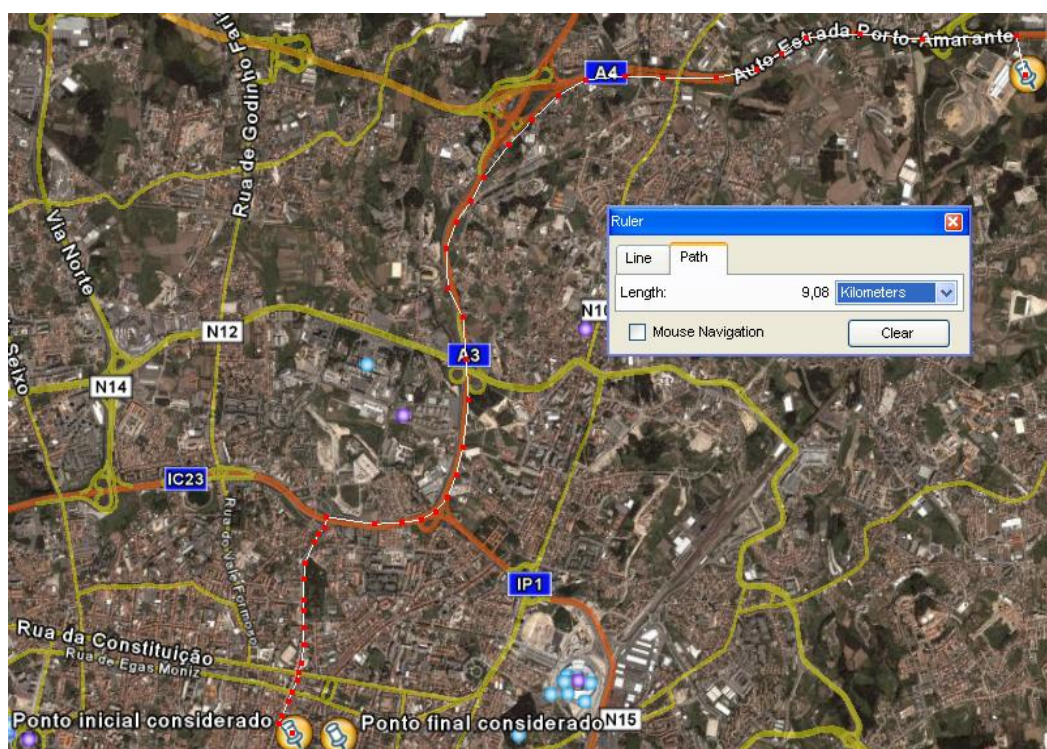
Nome da Rua	Distância (Percorrida em m)	Designação	Unidades de recolha (caso existam)
...			
Rua de Santa Catarina	199.6	Ponto n° 22	1 unidade(s) de recolha
Rua de Gonçalo Cristóvão	58.9	Percurso	Sem Recolha de RSU
...			
Rua de João das Regras	66.5	Ponto n° 16	2 unidade(s) de recolha
...			
Rua de Santa Catarina	26	Destino Final	Sem recolha de RSU
	Distância Total: (Km)	7.5	Unidade(s) Recolhida(s) Total: 20

Figura 5.23 - Relatório do percurso do veículo XP-00-01 e distância percorrida.

Comparando a realidade do programa Ecofone, para a equipa e veículo testado, que corresponde ao caso optimizado de trinta pontos de recolha (agrupados em vinte e quatro pontos: cinco super-contentores e dezanove contentores). Os dados fornecidos pela Lipor apontam para um percurso diário de 40 km.

Atendendo a que, conforme ilustra a Figura 5.24, o percurso envolvido desde a Lipor ao ponto arbitrado como de partida é de cerca de 9 km, e a distância do ponto final arbitrado à Lipor é aproximadamente a mesma (cerca de 9.2 km), então o mesmo percorre uma distância de cerca de 18.2 km para se deslocar da Lipor ao ponto de partida e do ponto final à Lipor. Ou seja, para efectuar a recolha, o veículo percorre uma distância de cerca de 22 km.

O percurso optimizado apresenta uma distância de 9.9 km o que corresponde a uma redução significativa do percurso do veículo. Convém, no entanto, referir que alguns pontos de recolha (super-contentores) foram colocados no meio de cada segmento de rua e que os pontos de entrada e de saída para a recolha poderão não corresponder aos pontos de entrada e saída do percurso realizado.



Nota: Imagem retirada do Google Earth

Figura 5.24 - Trajecto do veículo entre Lipor-ponto de partida e ponto final-Lipor.

Capítulo 6 - Conclusões e Trabalhos Futuros

Sumário

Neste capítulo apresentam-se as conclusões retiradas dos estudos efectuados. São igualmente apresentadas várias linhas de orientação que podem ser seguidas em desenvolvimentos posteriores das matérias abordadas neste trabalho.

6.1 Síntese Final

Nos últimos anos, a densidade populacional em zonas urbanas tem crescido de uma forma acentuada fazendo com que o problema da recolha de resíduos sólidos urbanos e o desenvolvimento de sistemas que permitam reduzir os custos envolvidos sejam objectivos pertinentes e actuais que necessitam de ser analisados.

Na primeira fase deste trabalho descreveram-se formas de resolução deste problema baseadas no conhecimento adquirido e introduziram-se as potencialidades dos sistemas de informação geográfica na especificação do problema da recolha de resíduos sólidos.

Enumeraram-se vários métodos de optimização que permitem resolver o problema da recolha de resíduos sólidos, apresentando-se, para cada caso, as características específicas de cada método.

Na penúltima fase do trabalho, criaram-se vários problemas teóricos de recolha de resíduos sólidos que foram resolvidos usando o método da programação linear binária. Deste estudo pode concluir-se que este método permite resolver o problema da recolha de resíduos de um modo eficaz.

Por fim, testou-se a aplicação desenvolvida com um caso real. A situação considerada foi um sistema de recolha de resíduos sólidos urbanos da Lipor, designado por Ecofone. Os resultados demonstram que é possível reduzir drasticamente os custos da recolha se os percursos efectuados corresponderem aos percursos mínimos.

6.2 Trabalhos futuros

Alguns dos tópicos abordados nesta tese podem ser objecto de estudo em trabalhos posteriores. Vários módulos podem ser desenvolvidos e as metodologias usadas podem ser empregues noutras situações.

6.2.1 Algoritmos de optimização

Neste trabalho resolveu-se o problema da recolha de resíduos sólidos urbanos usando a programação linear binária e o método de inserção mais próxima. No futuro, torna-se importante averiguar o desempenho dos outros métodos apresentados no Capítulo 3 na resolução deste problema.

Poder-se-á ainda proceder à introdução de outros valores para os parâmetros α , β e γ , na função objectivo (Equação 3.20) diferente dos considerados. Desta forma poder-se-á minimizar o custo global das viagens tendo em consideração o custo de saída dos veículos e o estabelecimento de prioridades em determinados percursos.

6.2.2 Integração com sistemas de informação geográfica

A formulação do problema de recolha de resíduos sólidos urbanos permitirá, com auxílio das potencialidades dos sistemas de informação geográfica, o desenvolvimento de um conjunto de funcionalidades como as que se descrevem de seguida:

- Introdução automática dos pontos de recolha no mapa com informação das ruas e sentidos de trânsito através do sistema de posicionamento global;
- Introdução das quantidades a recolher usando sensores, a colocar nos colectores de resíduos, que transmitam informação regular sobre a taxa de enchimento dos colectores;
- Ligação directa entre o programa e a folha de cálculo Excel de forma a poder trocar informação de uma forma directa para proceder, por exemplo a tratamento estatístico;
- Criação de relatórios extras para que a informação das recolhas seja mais completa e automática; por exemplo, de forma a que no final da recolha, a informação seja armazenada de uma forma automática;

- Localização em tempo real do posicionamento e do estado do veículo, recorrendo a sistemas de posicionamento global (GPS);
- Possibilidade de, alterar, em tempo real uma rota, pelas mais variadas razões, e de transmiti-la directamente ao veículo envolvido, por forma a que este possa executá-la de imediato.

Anexo A - Noções Gerais sobre Grafos

Este anexo tem por objectivo descrever algumas noções básicas sobre a teoria dos grafos. São ainda apresentadas algumas definições e notações usadas ao longo da tese e que servem de base para o estudo e desenvolvimento de alguns métodos descritos e desenvolvidos neste trabalho.

A.1 Noções básicas

Uma rede ou também designado por grafo é um conjunto de vértices (pontos, ou nós) $x_1, x_2, \dots, x_n$ e um conjunto de linhas (arcos ou arestas) $a_1, a_2, \dots, a_m$, que ligam todos ou alguns destes vértices (Cristofides, 1975).

O grafo G é definido por $G(X, A)$, sendo X é o conjunto de vértices e A é o conjunto de linhas.

No que diz respeito às linhas, o grafo poderá ser direccionado (Figura A.1), isto é, as linhas possuem uma direcção; não direccionado (Figura A.2), quando suas linhas não possuem direcção ou misto (Figura A.3) com linhas direccionadas e não direccionadas.

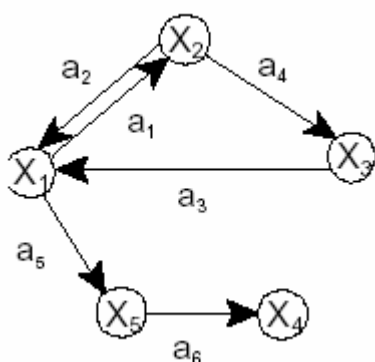


Figura A.1 - Grafo direccionado.

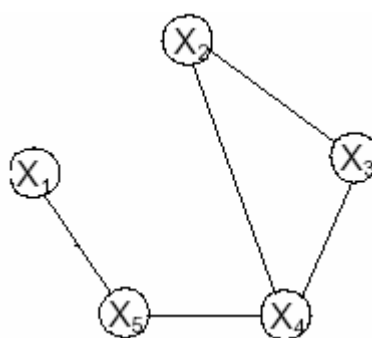


Figura A.2 - Grafo não direccionado.

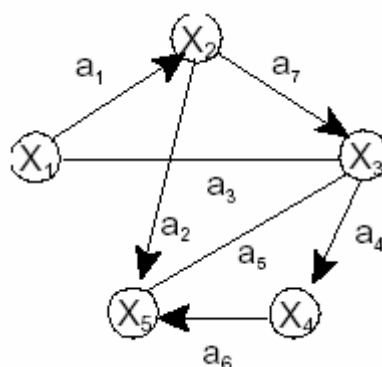


Figura A.3 - Grafo misto.

Define-se **grafo esparso**, como sendo um grafo onde o número de linhas é muito inferior ao número de vértices.

Um caminho designa-se por **caminho (direccionado)** num grafo direccionado se existir uma qualquer sequência de arcos onde o vértice final de um é o vértice inicial do seguinte.

Designa-se por **caminho simples** um caminho no qual um arco não aparece mais que uma vez.

Um **caminho elementar** é um caminho no qual um vértice não aparece mais que uma vez.

Uma **cadeia (não direccionada)** num grafo é qualquer sequência de arestas na qual cada aresta liga às suas arestas adjacentes através dos seus dois vértices.

Um grafo é dito **conexo** se existir entre quaisquer pares de nós do grafo G pelo menos uma cadeia. Caso contrário o grafo é dito não conexo.

Se a cada arco (x_i, x_j) de um grafo for associado um custo c_{ij} , o **custo de um caminho** é igual à soma dos custos dos arcos que formam este caminho.

Um caminho $a_1, a_2, \dots, a_q$, no qual o vértice inicial a_1 coincide com o vértice final a_q designa-se por Circuito.

Circuito elementar: É um circuito no qual um vértice só aparece uma vez (excepto o inicial e o final que são os mesmos).

Circuito hamiltoniano: É um circuito elementar que contém todos os n vértices do grafo G .

Ciclo: É uma parte não direccionada de um circuito. Assim, um ciclo é uma sequência de arcos onde o vértice inicial e final são os mesmos. A Figura A.4 apresenta um exemplo de um ciclo.

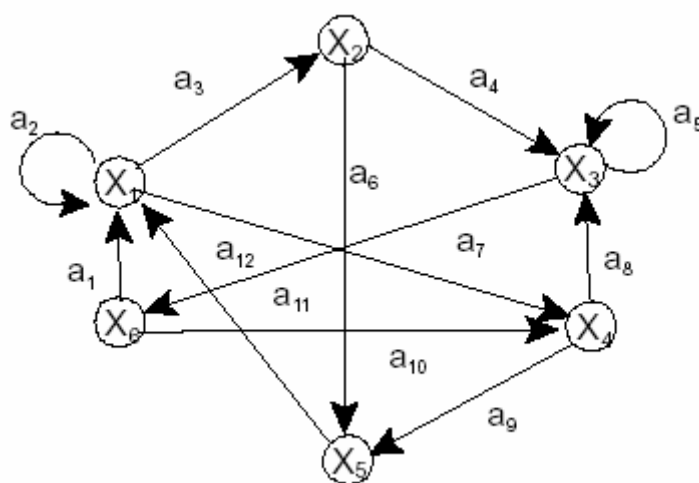


Figura A.4 - ilustração de um ciclo (a_1, a_3, a_4, a_7 formam um ciclo).

A.2 Representação matricial de um grafo

Matriz de adjacência: Para um grafo $G(X, A)$, a sua matriz de adjacência, representa-se por $A=[a_{ij}]$ e define-se como:

$$a_{ij} = \begin{cases} 1 & \text{se o arco } (x_i, x_j) \text{ existe em } G \\ 0 & \text{se o arco } (x_i, x_j) \text{ não existe em } G \end{cases}$$

Matriz de incidência: Para um grafo $G(X, A)$, a sua matriz de incidência, denotada por $B=[b_{ij}]$, é uma matriz $n \times m$, onde m é o número de arcos, definida como:

$$b_{ij} = \begin{cases} 1 & \text{se } x_i \text{ é o vértice inicial de } a_j \\ -1 & \text{se } x_i \text{ é o vértice final de } a_j \\ 0 & \text{se } x_i \text{ não é o vértice terminal de } a_j, \text{ ou se } a_j \text{ é um laço} \end{cases}$$

Se $G(X, A)$ for um grafo não direccionado, a matriz de incidência é definida do mesmo modo trocando apenas os valores “-1” por “1”.

Grau de um vértice: O número de arcos que têm x_i como vértice inicial é o grau de saída do vértice x_i , e é denotado por $d_s(x_i)$ e do mesmo modo o número de arcos que tem x_i como seu vértice final é chamado o grau de entrada do vértice x_i e é denotado por $d_e(x_i)$. Assim, na Figura A.4, o grau de saída exterior do vértice x_6 , por exemplo, é 2 e o grau de entrada do vértice x_6 é 1.

A soma dos graus de entrada e de saída de todos os vértices deve ser igual ao número de arcos, m , de G , isto é:

$$\sum_{i=1}^n d_s(x_i) = \sum_{i=1}^n d_e(x_i) = m$$

A.3 Referências do capítulo

Cristofides, N., Graph Theory - An Algorithmic Approach; Academic Press, 1975.