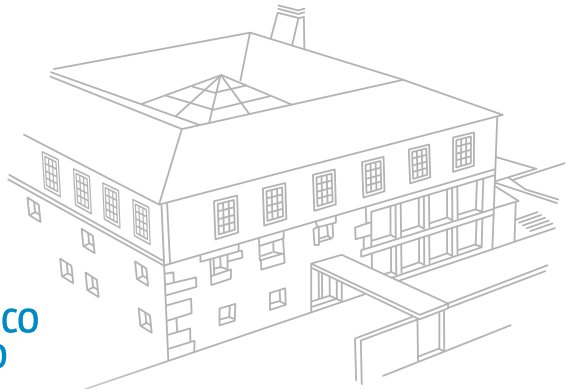

ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

RAMP para o Problema de Localização de Hubs com Afetação Múltipla e sem Restrições de Capacidade

DESIGNAÇÃO DO MESTRADO Mestrado em Engenharia Informática

AUTOR Fábio José Magalhães Maia

ORIENTADOR(ES) Dorabela Regina Chiote Ferreira Gamboa

ANO 2011

A todos aqueles que acreditam no meu valor, e quem não acredita com certeza vai passar a acreditar...

Agradecimentos

Quero agradecer a todas as pessoas, em particular à minha família, que ao longo destes anos de formação me têm apoiado e contribuído para o meu crescimento como pessoa. Agradecendo em geral à equipa de trabalho do gabinete do CIICESI (Centro de Investigação e Inovação de Ciências Empresariais e Sistemas de Informação), e em particular à minha orientadora Prof. Doutora Dorabela Gamboa, pelo apoio e toda a ajuda que me tem vindo a disponibilizar ao longo deste comprido caminho, bem como também ao investigador Telmo Matos pela ajuda ao longo desse tempo.
Quero também agradecer ao CIICESI pelas instalações e recursos disponibilizados para a realização deste trabalho.

Este trabalho é financiado por Fundos FEDER através do Programa Operacional Factores de Competitividade – COMPETE e por Fundos Nacionais, através da FCT – Fundação para a Ciência e a Tecnologia no âmbito do projeto PTDC/EGE-GES/104482/2008.

Resumo

Os Problemas de Localização de Instalações (*Facility Location Problems* – FLP) são problemas complexos que assumem um grande foco de estudo por parte da comunidade científica. Os FLP têm várias aplicações no mundo real e em diversas áreas, tais como, telecomunicações, redes de computadores, redes de transporte, rede elétrica, localização de hospitais, localização de aeroportos, entre muitos outros.

O Problema de Localização de Hubs com Afetação Múltipla e Sem Restrições de Capacidade (*Uncapacitated Multiple Allocation Hub Location Problem* – UMAHLP) faz parte do grupo de problemas de localização extensivamente estudados. Tratando-se de um problema de otimização combinatória NP-difícil, a utilização de métodos exatos na resolução de problemas práticos de grande dimensão pode ser seriamente comprometida pelos tempos computacionais necessários para a obtenção da solução ótima. Para ultrapassar esta dificuldade, um número significativo de algoritmos heurísticos têm sido propostos com o objetivo de encontrar soluções de boa qualidade em tempos tão reduzidos quanto possível.

O sucesso da metaheurística *Relaxation Adaptive Memory Programming* (RAMP) aplicada ao Problema de Localização de Instalações sem Restrições de Capacidade (*Uncapacitated Facility Location Problem* – UFLP) apresenta esta abordagem como bastante promissora na aplicação a outros problemas de localização. O UMAHLP é um exemplo clássico destes problemas.

Neste contexto, pretende-se com este estudo, explorar as vantagens da aplicação da abordagem RAMP ao UMAHLP. A abordagem RAMP baseia-se na exploração da relação primal-dual do problema, orientando a pesquisa com base em princípios de memória adaptativa. O método RAMP faz uso de vários níveis de sofisticação, definidos pelo grau de intensidade que são explorados os lados primal e dual do problema. Deve-se começar pela implementação da versão mais simples do método e só avançar para formas mais complexas, caso seja necessário, uma vez que o método RAMP é incremental.

Para o UFLP foram implementados dois algoritmos, um com base na metaheurística Pesquisa por Dispersão (*Scatter Search* – SS) e outro tendo por base a versão mais sofisticada do método RAMP, designada de PD-RAMP, que explora intensivamente ambos os

lados da relação primal-dual. O algoritmo PD-RAMP implementado engloba uma versão mais simples do algoritmo SS proposto, para explorar o espaço de soluções do lado primal, sendo o lado dual explorado pelo método *Dual-Ascent*. No UMAHLP foi aplicada uma versão mais simples do RAMP, intensificando a exploração do lado dual do Problema, através do método *Dual-Ascent*, enquanto que o lado primal é explorado, de uma forma mais simples, tendo por base o método de Pesquisa Tabu (*Tabu Search* – TS).

A aplicação do método RAMP aos problemas UFLP e UMAHLP, revelou-se muito robusta e eficiente, demonstrando bons resultados para as instâncias de teste padrão existentes para cada um dos problemas. Em ambos os problemas tratados os algoritmos propostos conseguem encontrar a maior parte das melhores soluções conhecidas, obtendo excelentes resultados. Para o UMAHLP são encontradas duas soluções melhores do que as conhecidas.
O método RAMP demonstrou, mais uma vez, ser uma metaheurística, que apesar de ser recente, já apresenta um elevado nível de sucesso na resolução de problemas complexos.

Palavras-chave: Localização de Instalações, Localização de Hubs, UFLP, UMAHLP, RAMP, Relaxação Matemática.

Abstract

Facility Location Problems (FLP) are complex problems that have been intensively studied by the scientific community. FLP have many applications in different areas in the real world, such as, telecommunications, computer networks, transportation and electrical networks, hospital and airports location, among others.

The Uncapacitated Multiple Allocation Hub Location Problem (UMAHLP) belongs to the facility location problems widely study in the literature. Being a NP-Hard combinatorial optimization problem, the use of exact algorithms can be seriously compromised due to the high computational times required to obtain the optimal solution. To overcome this difficulty, a significant number of heuristic algorithms of various types have been proposed with the aim of finding high quality solutions in reasonable computational times.

The success obtained with the RAMP (Relaxation Adaptive Memory Programming) application to the Uncapacitated Facility Location Problem (UFLP) suggested RAMP as a promising metaheuristic approach for other location problems. The UMAHLP is a classical example of this kind of problems.

In this context, our goal was to exploit the advantages of a RAMP application to the UMAHLP. RAMP is a new metaheuristic framework that exploits the problem's primal-dual relationship, guiding the search with adaptive memory. Since the RAMP framework is incremental, we start by implementing the simplest version of the RAMP approach and, if necessary, we move to more sophisticated versions of the method.

Two algorithms were implemented for the UFLP, one based on the Scatter Search (SS) method and another one designed within the RAMP framework, namely a more a sophisticated RAMP approach designated by PD-RAMP, that intensively explores both the primal and dual sides of the problem. This algorithm includes a simpler version of the SS to exploit the primal side and a Dual-Ascent procedure to exploit the dual side. For the UMAHLP, we propose the simplest version of the RAMP method, that only intensively explores the dual side. This algorithm uses a Dual-Ascent method in the dual side and the primal side is explored by a Tabu Search (TS) based heuristic.

The RAMP applications for the UFLP and the UMAHLP, proved to be very effective and efficient, obtaining excellent results for the standard data sets tested. In both problems report here, the proposed algorithms achieved the optimal solution for almost all instances in reasonable computational times. Additionally, we revealed two new best known solutions for the UMAHLP.
Despite being recent, the RAMP framework has proved to be very effective for the solution of large scale complex problems.

Keywords: Facility Location, Hub Location, UFLP, UMAHLP, RAMP, Relaxation Mathematics.

Índice

Agradecimentos ii
Resumo iii
Abstract v
Lista de Tabelas ix
Lista de Figuras x

1 Introdução **1**

2 Conceitos e Definições Gerais **3**
2.1 Problemas de Localização de Instalações 3
2.2 Conceitos sobre métodos de resolução 5
2.3 Pesquisa Tabu 7
2.4 Pesquisa por Dispersão 9

3 Relaxation Adaptative Memory Programming – RAMP **11**

4 Problema de Localização de Instalações sem Restrições de Capacidade – UFLP **14**
4.1 Introdução 14
4.2 Algoritmos Metaheurísticos para o UFLP 16
4.3 Descrição do algoritmo de Pesquisa por Dispersão (SS) 18
4.4 Descrição do algoritmo PD-RAMP 22
4.4.1 Método Dual 22
4.4.2 Método Primal 23
4.5 Estruturas de Dados 23
4.6 Resultados Computacionais 24

5 Problema de Localização de Hubs com Afectação Múltipla e Sem Restrições de Capacidade – UMAHLP **27**
5.1 Introdução 27
5.2 Algoritmos Heurísticos para o UMAHLP 31

5.3 Descrição do Algoritmo 32
5.3.1 Método Dual 33
5.3.2 Método Primal 37
5.4 Estruturas de Dados 40
5.5 Resultados Computacionais 41

6 Conclusões e Trabalho Futuro **50**

Lista de Tabelas

4.1 UFLP: Resultados dos melhores algoritmos conhecidos 25

5.1 UMAHLP: Instâncias de teste de problemas AP assimétricos — $\chi = 3$, $\alpha = 0.75$ e $\delta = 2$ 44
5.2 UMAHLP: Instâncias de teste de problemas AP assimétricos grandes — $\chi = 3$, $\alpha = 0.75$ e $\delta = 2$ 45
5.3 UMAHLP: Instâncias de teste de problemas AP simétricos — $\chi = 1$, $\alpha = 0.1$ e $\delta = 1$ 46
5.4 UMAHLP: Instâncias de teste de problemas AP simétricos — $\chi = 1$, $\alpha = 0.5$ e $\delta = 1$ 47
5.5 UMAHLP: Instâncias de teste de problemas AP simétricos — $\chi = 1$, $\alpha = 0.9$ e $\delta = 1$ 48
5.6 UMAHLP: Instâncias de teste de problemas CAB 49

Lista de Figuras

2.1 Algoritmo de Pesquisa Local 6
2.2 Algoritmo genérico de Pesquisa Tabu 8
2.3 Algoritmo de Pesquisa por Dispersão 10

3.1 Modelo PD-RAMP 12

4.1 UFLP: Formulação de Programação Linear Inteira 15
4.2 UFLP: Modelo do procedimento Pesquisa por Dispersão 18
4.3 UFLP: Método para gerar soluções diversificadas 19
4.4 UFLP: Método de melhoramento 20
4.5 UFLP: Método para criar e atualizar o conjunto de referência 21
4.6 UFLP: Método para gerar os subconjuntos 21
4.7 UFLP: Método para combinar as soluções 22
4.8 UFLP: Estruturas de dados 24

5.1 UMAHLP: Formulação de Programação Linear Inteira – 1 28
5.2 UMAHLP: Formulação de Programação Linear Inteira – 2 29
5.3 UMAHLP: Exemplo – Problema de localização de hubs com alocação singular e múltipla 30
5.4 UMAHLP: Formulação condensada do problema dual 33
5.5 UMAHLP: Exemplo dos nós que estão mais perto 34
5.6 UMAHLP: Método *Dual-Ascent* 36
5.7 UMAHLP: Método heurístico para resolução do problema de transportes . 38
5.8 UMAHLP: Método de melhoramento 40
5.9 UMAHLP: Estruturas de dados 42

Capítulo 1

Introdução

Os Problemas de Localização de Instalações (FLP), são problemas muitos conhecidos e muito estudados pela comunidade científica, muito devido às aplicações que tem no mundo real. Os FLP aparecem em várias áreas de enorme importância, nomeadamente, redes de computadores, localização de hospitais, localização de escolas, localização de correios, localização de aeroportos, entre muitas outras.

Como podemos verificar pelas várias aplicações que os FLP cobrem, estas são maioritariamente compostas por problemas principalmente de grandes dimensões. Inicialmente os métodos exatos, conseguiam resolver estes problemas de forma eficiente e eficaz, uma vez que encontravam a solução ótima do problema em tempos considerados aceitáveis. No entanto, com a evolução dos mercados globais e o aumento da competitividade entre organizações, os métodos exatos começaram a comprometer, uma vez que demoram muito tempo a obter a solução, apesar desta ser ótima. Desta forma, vários algoritmos heurísticos têm sido propostos na literatura, com o objetivo de contrariar o excesso de tempo necessário por parte dos algoritmos exatos, ainda que a solução obtida não seja a solução ótima, mas uma boa solução para o problema.

Neste trabalho foram considerados dois problemas, o UFLP (problema de localização de instalações sem restrições de capacidade) e o UMAHLP (problema de localização de hubs com alocação múltipla e sem restrições de capacidade), tendo como principal objetivo o desenvolvimento de algoritmos de estado de arte com base na nova metaheurística RAMP [1]. Com este trabalho também se pretende adquirir bases para tratar outros problemas no futuro e contribuir com conhecimento para a comunidade científica, de forma que este estudo possa servir de base a investigações futuras.

A metaheurística RAMP, proposta por Rego em 2005, tem como base a exploração de forma integrada de ambos os lados, primal e dual do problema. O RAMP é uma *framework*

incremental que considera vários níveis de sofisticação. Inicialmente deve-se começar por desenvolver uma versão mais simples, e só passar para formas mais complexas, caso haja essa necessidade. Desta forma a *framework* é bastante flexível e adaptável aos objetivos e resultados que se pretendem atingir.
Apesar de ser uma metaheurística recente, já foi aplicada com sucesso a vários problemas de otimização combinatória, nomeadamente ao *Multi-Resource Generalized Assignment Problem* (MRGAP) [2], *Uncapacitated Facility Location Problem* (UFLP) [3], *Linear Ordering Problem* (LOP) [3] e *Resource Constraint Project Scheduling Problem* (RCPSP) [4].

O trabalho que se pretende desenvolver é um trabalho ambicioso, com objetivos claros e concisos e enquadra-se numa área de grande relevância. Ambos os problemas a tratar são considerados problemas NP-difíceis [5][6][7][8] e os algoritmos que se pretendem propor têm por base uma metaheurística nova, ainda com poucas aplicações, mas já com muito sucesso.
São propostos dois algoritmos para o UFLP, o primeiro, tendo por base a metaheurística Pesquisa por Dispersão (SS), e o segundo baseado numa versão mais sofisticada do método RAMP (PD-RAMP). Para o UMAHLP é proposto um algoritmo considerando o nível mais simples do método RAMP.

Os resultados atingidos superaram todos os objetivos. No UFLP, tanto o SS, como o PD-RAMP conseguem competir com os melhores algoritmos propostos na literatura, conseguindo para ambos os algoritmos valores muito próximos de zero, no que diz respeito ao desvio da solução encontrada em relação à solução ótima. O algoritmo RAMP proposto para o UMAHLP consegue obter excelentes resultados, obtendo duas soluções melhores do que as atualmente conhecidas. O algoritmo consegue encontrar 98% das melhores soluções conhecidas e descobre duas novas soluções.

Esta dissertação está estruturada da seguinte forma. Nos capítulos 2 e 3, são apresentados alguns conceitos teóricos que estão na base do trabalho desenvolvido, com especial ênfase no método RAMP. Nos dois capítulos seguintes, introduzem-se os problemas tratados nesta dissertação e descreve-se o trabalho desenvolvido. São apresentados os resultados obtidos e é feita a comparação com resultados apresentados por outros autores, que propuseram as melhores abordagens conhecidas. Finalmente, no capítulo 6, são apresentadas as conclusões do trabalho elaborado e o seguimento do trabalho efetuado.

Capítulo 2

Conceitos e Definições Gerais

Este capítulo tem como objetivo, introduzir alguns conceitos teóricos importantes acerca do trabalho desenvolvido, de forma ao leitor ter uma melhor perceção e conhecimento sobre o trabalho descrito posteriormente.

2.1 Problemas de Localização de Instalações

Um problema de localização de instalações (*Facility Location Problem* – FLP) consiste em determinar quais as localizações onde construir/abrir instalações de forma a servir com custo mínimo um conjunto de clientes. O problema é constituído por um conjunto de potenciais localizações, à qual cada instalação tem associado um custo de abertura. Adicionalmente, existe o custo de transporte entre as potencias localizações e cada cliente. As instalações podem ainda ter capacidade limitada, e dependendo da variante do problema, o cliente pode ser servido por apenas uma ou por várias instalações.

Para o FLP, existem várias variantes do problema, desde versões mais simples, sem restrições de capacidade até versões mais avançadas, com restrições de capacidade. Alguns exemplos são, o problema de localização de instalações sem restrições de capacidade (*Uncapacitated Facility Location Problem* – UFLP), o problema de localização de hubs com afetação múltipla e sem restrições de capacidade (*Uncapacitated Multiple Allocation Hub Location Problem* – UMAHLP) entre muitos outros, nomeadamente para problemas sem restrições de capacidade.

Para os problemas com restrições de capacidade temos os seguintes exemplos, o problema de localização de instalações com restrições de capacidade (*Capacitated Facility Location Problem* – CFLP), o problema de localização de instalações com restrições de capacidade e um único servidor (*Single-Source Capacitated Facility Location Problem* – SSCFLP),

entre outros. Para cada uma dessas variantes existem ainda muitos outros tipos de problemas, cada um com as suas próprias caraterísticas.

Para ambas as versões (CFLP e UFLP) existem restrições comuns, como é o caso da procura dos clientes, os custos de abertura das instalações e os custos das instalações servirem os clientes. A procura dos clientes não tem influência no UFLP, uma vez que as instalações têm capacidades ilimitadas, já no CFLP, e uma vez que as instalações têm capacidades limitadas, as mesmas têm grande influência, sendo necessário determinar para cada cliente, qual ou quais as instalações que o vão servir. A variante SSCFLP é muito parecida com o CFLP, apenas difere no facto de cada cliente apenas ser servido por uma única instalação.

O UMAHLP faz parte dos problemas de localização, designados por problemas de localização de hubs (*Hub Location Problems* – HLP), muito similares aos FLP, mas com caraterísticas próprias. Apesar do problema ser muito parecido com os FLP tem algumas diferenças importantes. O problema consiste em determinar a localização dos nós hub (de agora em diante apenas será designado por hub, em vez de nó hub) de um conjunto de potenciais localizações, sendo que apenas algumas dessas localizações podem ser hubs, e as restantes localizações são nós (não hubs) que deverão ser alocadas aos hubs.

Podemos fazer a analogia com o FLP, sendo que os hubs correspondem às instalações, enquanto que os nós são os clientes. De realçar que, desde já é percetível uma diferença, que apesar de parecer mínima é uma diferença que transforma em muito o problema, pois enquanto o FLP contém dois conjuntos de localizações, nomeadamente as instalações e os clientes, o HLP apenas tem um.

Outra grande diferença que faz com que o problema tenha uma formulação diferente da do FLP é que este problema é constituído por dois tipos de custos, a distância entre ambos os nós, e o fluxo de cada um dos nós para os restantes. A maior diferença entre o HLP e o FLP, reside no facto de o FLP apenas minimiza o custo de alocação dos clientes às possíveis instalações abertas, já o HLP tem como objetivo minimizar as distâncias entre todos os nós da rede.

O problema é ainda divido em vários subproblemas, cada um deles com as suas particularidades. Existe a versão mais simples sem restrições de capacidade (*Uncapacitated Hub Location Problem* – UHLP), podendo ser considerada alocação singular (*Uncapacitated Single Allocation Hub Location Problem* – USAHLP) ou múltipla (*Uncapacitated Multiple Allocation Hub Location Problem* – UMAHLP), ou seja, um nó pode estar alocado a um ou a mais hubs, respetivamente.

Pode ainda ser definido à priori o número de nós hub a alocar, sendo o problema designado na literatura por *Uncapacitated Single Allocation p-Hub Median Problem* (USApHMP) e *Uncapacitated Multiple Allocation p-Hub Median Problem* (UMApHMP) na versão com alocação múltipla. O HLP é ainda constituído por versões com restrições de capacidade para alocação singular (*Capacitated Single Allocation Hub Location Problem* – CSAHLP) e para alocação múltipla (*Capacitated Multiple Allocation Hub Location Problem* – UMAHLP) . Os problemas tratados e descritos neste trabalho são o UFLP e o UMAHLP.

2.2 Conceitos sobre métodos de resolução

Nesta secção apresentam-se algumas definições que se consideram importantes para melhor perceber os algoritmos propostos neste trabalho.

Espaço de soluções admissíveis: conjunto composto por todas as soluções válidas para o problema, isto é, que satisfazem todas as restrições do problema.

Estrutura de Vizinhança: considerando S o espaço de soluções admissíveis de um problema P. Sendo s uma solução de P, chama-se vizinhança de s, $V(s)$, ao conjunto de soluções que se podem alcançar a partir de s, através de uma operação, designada movimento. Cada elemento de $V(s)$ é denominado vizinho de s.

Ótimo local: chama-se ótimo local à melhor solução obtida por um algoritmo de melhoramento com uma determinada estrutura de vizinhança.

Ótimo global: chama-se ótimo global à solução ótima do problema.

Algoritmos exatos: garantem que a solução ótima é obtida, mas que podem necessitar de muito tempo para a conseguir, bem como também de muitos recursos computacionais. Posto isto, os algoritmos exatos podem apresentar grandes limitações, nomeadamente para problemas de otimização combinatória de grandes dimensões.

De forma a colmatar esta principal fraqueza, aparecem os algoritmos heurísticos, que apesar de não garantirem que a solução ótima é encontrada, permitem encontrar uma boa solução para o problema em tempos computacionais aceitáveis.

Algoritmos heurísticos

Os algoritmos heurísticos podem ser classificados em três grupos: algoritmos construtivos, algoritmos de melhoramento ou pesquisa local e algoritmos metaheurísticos.

Algoritmos construtivos: tal como o nome indica vão construindo a solução de forma iterativa e incremental, isto é, para cada iteração um componente é escolhido e inserido na solução em construção até se obter uma solução completa.

Algoritmos de melhoramento ou de pesquisa local: tentam melhorar a solução corrente perturbando a solução iterativamente, de forma a chegar a outra solução de melhor qualidade, utilizando uma estrutura de vizinhança. As vizinhanças são definidas sobre um modelo de representação do problema. Se o modelo consiste numa formulação matemática do problema, então a vizinhança baseia-se na atribuição de valores a variáveis da formulação.

Um algoritmo de pesquisa local começa com uma solução inicial admissível, e em cada iteração aplica um movimento definido na estrutura de vizinhança, passando para outra solução (melhor que anterior). O algoritmo termina quando não for possível melhorar a solução a partir de um movimento da estrutura de vizinhança definida, atingindo um ótimo local.

De seguida é apresentado, na figura 2.1, um modelo genérico de um algoritmo de pesquisa local, para o resolução de um problema de minimização com função objetivo f e onde $V(S_c)$ representa a vizinhança da solução corrente.

Procedimento de Pesquisa Local

Passo 1. Obter solução inicial admissível, S_0.

Passo 2. Inicializar solução corrente S_c, fazendo $S_c = S_0$.

Passo 3. Considerar uma solução $S_v \in V(S_c)$.

Passo 4. Se $f(S_v) < f(S_c)$, então $S_c = S_v$, senão $V(S_c) = V(S_c) - S_v$.

Passo 5. Se $V(S_c) \neq \oslash$ voltar para o **Passo 3**.

Passo 6. Terminar.•

Figura 2.1: Algoritmo de Pesquisa Local

Sendo assim, num problema com muitos ótimos locais, o ótimo local que o algoritmo de pesquisa local vai encontrar irá depender da solução inicial, e só será encontrado o ótimo global caso a pesquisa seja feita na vizinhança desse ótimo.

Devido a esta desvantagem, surgem as metaheurísticas que têm como objetivo conseguir escapar da vizinhança do ótimo local, de forma a conseguir passar para outra zona do espaço de soluções admissíveis com o intuito de encontrar o ótimo global, ainda que para tal seja necessário piorar a solução corrente.

Algoritmos metaheurísticos: complementam os procedimentos de pesquisa local permitindo a passagem por soluções de pior qualidade, tendo como objetivo explorar outra zonas do espaço de soluções, encontrando outros ótimos locais de melhor qualidade, e se possível conseguir alcançar o ótimo global.

2.3 Pesquisa Tabu

A metaheurística Pesquisa Tabu (*Tabu Search* – TS) foi proposta por Glover [9] [10] e é muito utilizada na resolução de problema complexos de otimização combinatória.

O método de Pesquisa Tabu é um procedimento adaptativo, que guia o algoritmo de pesquisa local na exploração do espaço de soluções. O algoritmo faz uso de uma estrutura de memória flexível que pode ter vários níveis (memória de curto, médio e longo prazo). Um algoritmo genérico de Pesquisa Tabu inclui os seguintes componentes:

Procedimento de Pesquisa Local: o algoritmo começa por utilizar um procedimento de pesquisa local para encontrar um ótimo local. O algoritmo continua a sua pesquisa deixando piorar a solução, de forma a passar para outra zona do espaço de soluções, com o objetivo de atingir outro ótimo local.

Lista Tabu: esta lista tem como objetivo ir guardando os últimos movimentos efetuados, de forma a impedir que a pesquisa local volte novamente ao ótimo local visitado anteriormente, isto é, permita que a solução piore o suficiente para posteriormente atingir outro ótimo local. Isso é assegurado pela exclusão dos movimentos pertencentes à lista tabu, dos que vão sendo considerados a cada iteração, a não ser que estejam definidos **Critérios de Aspiração** (por exemplo, caso o movimento escolhido melhore a melhor solução encontrada até ao momento). Para além de se definir a informação guardada pela lista tabu, também é necessário definir o tamanho da lista, isto é, o número de movimentos que são guardados, ou o número de iterações que esse movimento irá ser considerado como tabu.

Critério de Paragem: uma vez que o algoritmo não pára quando encontra um ótimo local, e o problema pode ter múltiplos ótimos locais, é necessário definir o critério de paragem. Alguns exemplos de critérios de paragem utilizados na literatura são, número fixo de iterações, número de iterações consecutivas sem melhoramento, tempo fixo de utilização de CPU, entre outros.

Na figura 2.2, é apresentada uma estrutura genérica de um algoritmo básico de Pesquisa Tabu para a resolução de um problema de minimização com função objetivo f, onde $V(S_c)$ representa a vizinhança da solução corrente, T representa a lista com os movimentos tabu e $S(T)$ as soluções da vizinhança $V(S_c)$ que são proibidas. De realçar que no modelo apresentado não é considerado o critério de aspiração.

Procedimento de Pesquisa Tabu simples

Passo 1. Obter solução inicial admissível, S_0.

Passo 2. Inicializar solução corrente S_c e a melhor solução encontrada S_m, fazendo $S_m = S_c = S_0$.

Passo 3. Inicializar o contador de iterações $k = 0$.

Passo 4. Se $V(S_c) - S(T) = \oslash$, ir para o **Passo 8**.

Passo 5. Incrementar k e selecionar a melhor solução $S_v \in V(S_c) - S(T)$.

Passo 6. Se $f(S_v) < f(S_m)$, então $S_m = S_v$. Senão $V(S_c) = V(S_c) - S_v$.

Passo 7. Se ainda não foi atingido o número fixo de iterações totais, ou de iterações consecutivas sem melhorar S_m, ir para o **Passo 4**.

Passo 8. Terminar.•

Figura 2.2: Algoritmo genérico de Pesquisa Tabu

É importante também referir que a Pesquisa Tabu permite diferentes níveis de sofisticação, são elas:

Intensificação: A técnica de intensificação permite ao algoritmo explorar com maior intensidade uma zona do espaço de soluções que seja identificada como promissora. Por exemplo, utilizar memória de longo prazo de forma a se identificarem caraterísticas comuns às melhores soluções encontradas, com o objetivo de se favorecer soluções com essas caraterísticas.

Diversificação: Já a técnica de diversificação tem como objetivo permitir à pesquisa explorar outras zonas do espaço de soluções. Por exemplo, não considerar soluções com caraterísticas das visitadas recentemente.

2.4 Pesquisa por Dispersão

A metaheurística Pesquisa por Dispersão (*Scatter Search* – SS) é um método evolutivo proposto por Glover [11]. O método de Pesquisa por Dispersão combina vetores de soluções, com o objetivo de capturar informação que não se consegue obter separadamente nos vetores originais e utiliza métodos heurísticos auxiliares para avaliar as combinações produzidas e gerar novos vetores. A população de soluções vai sempre evoluindo em gerações sucessivas, tal como é esperado num método evolutivo.

A população de soluções é representada por um conjunto de soluções de referência cuja constituição exige soluções de boa qualidade e soluções diversificadas. O método procede à combinação de soluções gerando novas soluções, através da aplicação de combinações lineares pesadas de subconjuntos de soluções que posteriormente são tratadas por procedimentos heurísticos auxiliares.

Um algoritmo genérico de Pesquisa por Dispersão é constituído pelos seguintes componentes:

Método de geração de soluções diversificadas: gera soluções diversificadas a partir de soluções semente.

Método de melhoramento: usado para aplicar movimentos de forma a tornar a solução melhor.

Método de atualização do conjunto de referência: tem como objetivo criar e atualizar o conjunto de soluções de referência, contendo soluções de qualidade e soluções diversificadas.

Método de geração de subconjuntos: serve para gerar subconjuntos do conjunto de referência para posteriormente se proceder à combinação de soluções.

Método de combinação de soluções: combina as soluções escolhidas de forma a atingir novas soluções com caraterísticas comuns às soluções selecionadas.

Na figura 2.3 é apresentado um modelo genérico do método de Pesquisa por Dispersão.

Procedimento de Pesquisa por Dispersão

Fase Inicial

Passo 1. Utilizar o Método de Geração de Soluções Diversificadas para criar soluções.

Passo 2. Tentar melhorar as soluções obtidas no **Passo 1** através da aplicação do Método de Melhoramento.

Passo 3. Atualizar o Conjunto de Referência.

Passo 4. Se ainda não foi atingido o nível pretendido de qualidade e diversidade de soluções, voltar ao **Passo 1**.

Fase de Pesquisa por Dispersão

Passo 5. Obter os subconjuntos do conjunto de referência (Refset) através do Método de Geração de Subconjuntos.

Passo 6. Aplicar o Método de Combinação de Soluções para obter novas soluções.

Passo 7. Tentar melhorar as soluções obtidas no **Passo 6** através da aplicação do Método de Melhoramento.

Passo 8. Atualizar o Conjunto de Referência.

Passo 9. Se esta fase produziu soluções melhores ou se ainda não foi alcançado um número pré-definido de iterações, ir para o **Passo 5**.

Passo 10. Terminar.•

Figura 2.3: Algoritmo de Pesquisa por Dispersão

Capítulo 3

Relaxation Adaptative Memory Programming – RAMP

RAMP é uma nova abordagem metaheurística proposta por Rego [1], em 2005. O método RAMP integra o conceito de programação em memória adaptativa (*Adaptative Memory Programming* – AMP) [12], com procedimentos de relaxação matemática com o objetivo de explorar a relação primal-dual do problema de otimização. O termo primal representa o problema original e o respetivo espaço de soluções, já o termo dual refere-se ao problema dual de uma relaxação do problema original.

O método RAMP tem presente vários níveis de sofisticação. Na versão mais simples do método a pesquisa desenvolve-se essencialmente no espaço de soluções dual, sendo a interação entre o lado primal e dual do problema obtida através de formas simples de projeção das soluções duais no espaço primal. Em versões mais avançadas do método, são explorados ambos os lados do problema, dual e primal, de forma intensiva. Para tal o método incorpora técnicas mais elaboradas, permitindo uma pesquisa mais alargada no espaço de soluções primal e a criação de estruturas de memória mais complexas. A versão mais simples do método costuma ser designada apenas de RAMP (ou Dual-RAMP), sendo a mais elaborada designada de PD-RAMP, de forma a realçar a ligação entre o lado primal e dual do problema.

Para os vários níveis de sofisticação do método podem ser implementadas várias estratégias para a pesquisa de cada um dos lados, do lado dual salientam-se, como técnicas base, a utilização da relaxação Lagrangeana [13] e da relaxação por restrições substitutas [1], sendo no entanto possível a utilização de outros tipos de relaxação, incluindo técnicas de relaxação compostas, tais como, a técnica de relaxação paramétrica cruzada [1], também proposta por Rego quando apresentou o RAMP. Esta relaxação combina

fundamentos da relaxação lagrangeana com a relaxação por restrições substitutas.

Do lado primal do problema é sugerido o método de pesquisa por dispersão ou a pesquisa por ligação de caminhos (*Path-relinking* – PR) [14] como processos adequados à utilização de memória adaptativa. No entanto, em ambos os lados podem ser utilizadas estratégias alternativas , que incluem outras formas de relaxação e outros métodos evolutivos, tais como algoritmos genéticos (*Genetic Algorithms* – GA) e outras abordagens inspiradas em processos de evolução natural.
Os vários níveis de sofisticação são distintos e incrementais, deve-se iniciar pela versão mais simples, e só avançar para formas mais complexas caso seja necessário, ajustando a conceção do algoritmo em função dos resultados obtidos, de forma a se obterem melhores resultados.

Na figura 3.1 é apresentado um possível modelo do PD-RAMP. Este é um modelo genérico e é constituído por uma relaxação do problema original, que explora o lado dual, enquanto que para efetuar a exploração do lado primal é utilizado o método de pesquisa por dispersão.

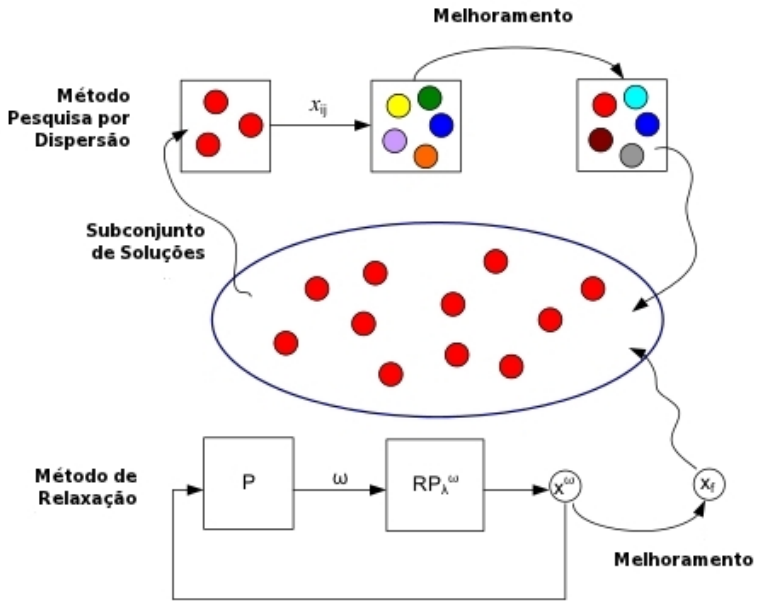

Figura 3.1: Modelo PD-RAMP

O método começa por criar e inicializar o conjunto de referência com as soluções produzidas por um método de relaxação. Nesta etapa o conjunto de referência contém informação relevante da abordagem dual, de seguida é utilizada essa informação e é feita a projeção das soluções para o espaço de soluções do lado primal.
O método de pesquisa por dispersão começa por escolher e combinar as soluções. De

seguida, é aplicado um método de melhoramento às soluções resultantes, dando origem a novas soluções. O método alterna entre a abordagem primal e dual, atualizando o conjunto de referência e terminando quando já não é possível melhorar as soluções encontradas, ou quando um determinado critério de paragem é alcançado.

A metaheurística RAMP já foi aplicada a vários problemas complexos tendo obtido bons resultados. Versões simples do RAMP foram aplicadas ao problema de localização de instalações sem restrições de capacidade (UFLP) [3] e ao *Resource Constraint Project Scheduling Problem* (RCPSP) [4], com excelentes resultados. Foram propostas abordagens PD-RAMP para o Problema de Ordenação Linear (*Linear Ordering Problem* – LOP) [3] e para o *Multi-Resource Generalized Assignment Problem* (MRGAP) [2], também com ótimos resultados.

Capítulo 4

Problema de Localização de Instalações sem Restrições de Capacidade – UFLP

4.1 Introdução

O problema de localização de instalações sem restrições de Capacidade (*Uncapacitated Facility Location Problem* – UFLP, também conhecido por *Uncapacitated Warehouse Location Problem* e *Uncapacitated Plant Location Problem*) é um problema muito estudado e com uma grande variedade de aplicações, tais como, na área das telecomunicações, localização de aeroportos, escolas, armazéns, centrais de tratamento de resíduos, fábricas, postos de correio, hospitais, livrarias, entre outros.

Um UFLP com m instalações e n clientes, em que cada instalação i está associado um custo fixo de instalação F_i e um custo de transporte c_{ij} para servir completamente um cliente j a partir da instalação i. O seu objetivo consiste em selecionar o conjunto de instalações a abrir de forma a satisfazer todos os clientes e minimizar os custos.
Na figura 4.1 é apresentada uma possível formulação de programação linear inteira do problema.
Nesta formulação, $C = \{1, ..., n\}$ e $A = \{1, ..., m\}$ correspondem respetivamente, aos conjuntos de clientes e de instalações, as variáveis x_{ij} indicam se j é ou não servido pela instalação i, sendo verificado pelas restrições (4.2), (4.3) e (4.4). A variável y_i, presente na restrição (4.5), especifica se a instalação i está aberta ou fechada.

Formulação de Programação Linear Inteira – UFLP

$$Min \qquad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} F_i y_i \tag{4.1}$$

$s.a$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in C, \tag{4.2}$$

$$x_{ij} \leq y_i \qquad \forall i \in A, j \in C, \tag{4.3}$$

$$x_{ij} \geq 0 \qquad \forall i \in A, j \in C, \tag{4.4}$$

$$y_i \in \{0,1\} \qquad \forall i \in A. \tag{4.5}$$

Figura 4.1: UFLP: Formulação de Programação Linear Inteira

Uma vez que se admite que as instalações têm capacidade ilimitada, quando selecionado o conjunto de localizações com instalação aberta, a afetação ótima é imediata, bastando servir cada cliente pela instalação mais próxima. Desta forma, pode-se definir o problema UFL-IP apenas com variáveis binárias (x_{ij} toma o valor 1 ou 0), uma solução UFLP fica totalmente definida pelo conjunto de instalações abertas (a solução pode ser representado por um vetor $Y = (y_1, ..., y_i, ..., y_m)$ onde y_i indica que a instalação está aberta e 0 caso esteja fechada).

A localização de instalações é um fator importante no sucesso ou fracasso dos negócios, pois aplicando uma correta distribuição das localizações das instalações pode se reduzir os custos operacionais. Analisando a literatura encontramos várias abordagens para a resolução deste problema. Tratando-se de um problema de otimização combinatória NP-difícil [5][6], a utilização de métodos exatos para a resolução de problemas de grande dimensão pode comprometer no encontro de uma boa solução em tempo útil devido aos tempos de computação elevados para se encontrar a solução ótima.
Uma vez que que os métodos exatos não revelam ser um caminho viável devido aos tempos de computação elevados, tem sido propostos vários algoritmos heurísticos, de forma a se encontrar soluções de boa qualidade em tempos tão reduzidos quanto possível.

Em 1963, Kuehn e Hamburger [16] propõem uma abordagem baseada em *Greedy Heuristics*, que em 1995 dá origem a um novo algoritmo proposto pelo autor Daskin [17]. Erlenkotter [18], em 1978, e Korkel [19] em 1989, apresentam uma abordagem baseada em *Branch and Bound*. Almeida e Alves, em 1992, propõem um algoritmo com base em *Simulated Annealing* e em 1989 e 1990, surgem as primeiras abordagens *Tabu Search* propostas por Glover [9] [10]. Em 1999, pelos autores Al-Sultan e Al-Fawzan [20], em 2003 por Ghosh [21], em 2004 pelos autores Michel e Hentenryck [22] e em 2006 por Sun [15] são apresentas outras abordagens *Tabu Search*. Kratica, Tosic, Filipovic e Ljubic [23], em 2001, apresentam um algoritmo genético e em 2006, Resende e Werneck [24] propõem uma abordagem híbrida (*Hybrid Multistart Heurist*). Em 2008 Gamboa e Rego [3] aplicam a abordagem RAMP ao UFLP. Em 2009, Pullan [25] propõem uma abordagem *Population Based Search*.

4.2 Algoritmos Metaheurísticos para o UFLP

Analisando a literatura existente encontramos vários algoritmos para o UFLP, baseados em diferentes metaheurísticas.

Em 2004, Michel e Hentenryck [22] apresentaram um algoritmo baseado em Pesquisa Tabu que sendo uma abordagem simples, foi designada *Simple Tabu Search* (TS-MH). O algoritmo utiliza um vetor para representar as localizações que estão abertas (valor 1 - a localização está aberta, valor 0 - a localização está fechada) e outro vetor para ir guardando o histórico das trocas de estado.

Também baseada em Pesquisa Tabu, Sun [15] apresenta, em 2006, uma abordagem mais sofisticada (TS-S). O algoritmo explora o espaço de soluções de uma forma mais complexa, fazendo uso de memória dinâmica e exploração responsiva para orientar o processo de solução e deste modo passar de uma tentativa de solução para outra. O algoritmo faz uso da memória dinâmica de curto, médio e longo prazo.
Já a exploração responsiva, que tem como objetivo encaminhar a orientação de pesquisa no espaço de soluções, baseia-se nas propriedades da solução corrente e no historial de pesquisa para encaminhar a solução na melhor direção.

Ao longo dos anos têm-se destacado os algoritmos híbridos [26]. De seguida são apresentados os mais relevantes.

Resende e Werneck [24] propõem, em 2006, um algoritmo designado por *Hybrid Multi-start Heuristic* (H-RH), o qual combina elementos de várias metaheurísticas, tais como, Pesquisa Tabu, Algoritmos Genéticos e Pesquisa por Dispersão. Este algoritmo divide-se em duas fases.
A primeira fase, tem como base a intensificação. Em cada iteração, é construída uma solução aleatória e aplicado um método de pesquisa local. A solução resultante (S) é combinada com outras soluções do conjunto de soluções de alta qualidade (representa as melhores soluções encontradas), resultando numa nova solução S'. De seguida, o algoritmo analisa as soluções S' e S, e insere-as no conjunto das soluções de alta qualidade, caso estas verifiquem determinadas critérios. A segunda fase é uma pós-otimização, que combina soluções do conjunto de soluções de qualidade com soluções do processo iterativo, onde se espera que resulte em melhores soluções.

Em 2008, surge um novo algoritmo desenvolvido por Gamboa e Rego [3], baseado numa nova metaheurística conhecida por *Relaxation Adaptative Memory Problem* (RAMP) [1]. A abordagem baseia-se na exploração da relação primal-dual do problema, orientando a pesquisa tendo por base princípios de memória adaptativa.
O primal corresponde ao problema original e correspondente espaço de soluções. Já o dual deriva de uma relaxação do problema original. Na abordagem RAMP para o UFLP, a pesquisa trabalha essencialmente no espaço de soluções dual, sendo a interação entre o primal e o dual obtida através de formas simples de projeção de soluções duais para o espaço primal.

Pullan [25], em 2009, apresenta um algoritmo designado por *Population Based Search* (PBS) que tem por base o trabalho desenvolvido por Resende e Werneck em 2006.
O objetivo da metaheurística é, ter um grande conjunto de bons pontos iniciais para posteriormente aplicar o método de pesquisa local, de forma a se explorar outros espaços de soluções para fechar esses pontos iniciais.
O método começa por criar a população inicial. De seguida, é executado repetidamente o algoritmo sendo aplicadas mutações e cruzamentos na população. A população é atualizada sempre que é encontrada uma solução de menor custo.
O algoritmo termina quando não é atingido um custo alvo ou é atingido o número máximo de gerações.

4.3 Descrição do algoritmo de Pesquisa por Dispersão (SS)

O algoritmo aqui proposto é baseado em Pesquisa por Dispersão [11][14][27]. O algoritmo é constituído por duas fases e cinco métodos, sendo que cada um dos métodos podem contribuir para apenas uma ou ambas as fases. De seguida é apresentado um modelo do algoritmo proposto.

Na figura 4.2 é apresentado um modelo do procedimento de Pesquisa por Dispersão implementado, onde se destacam as duas fases e os cinco métodos, a primeira fase termina quando for atingido o nível de qualidade pretendido, a segunda fase começa logo de seguida e termina quando não se verificarem alterações ao conjunto de referência (Refset).

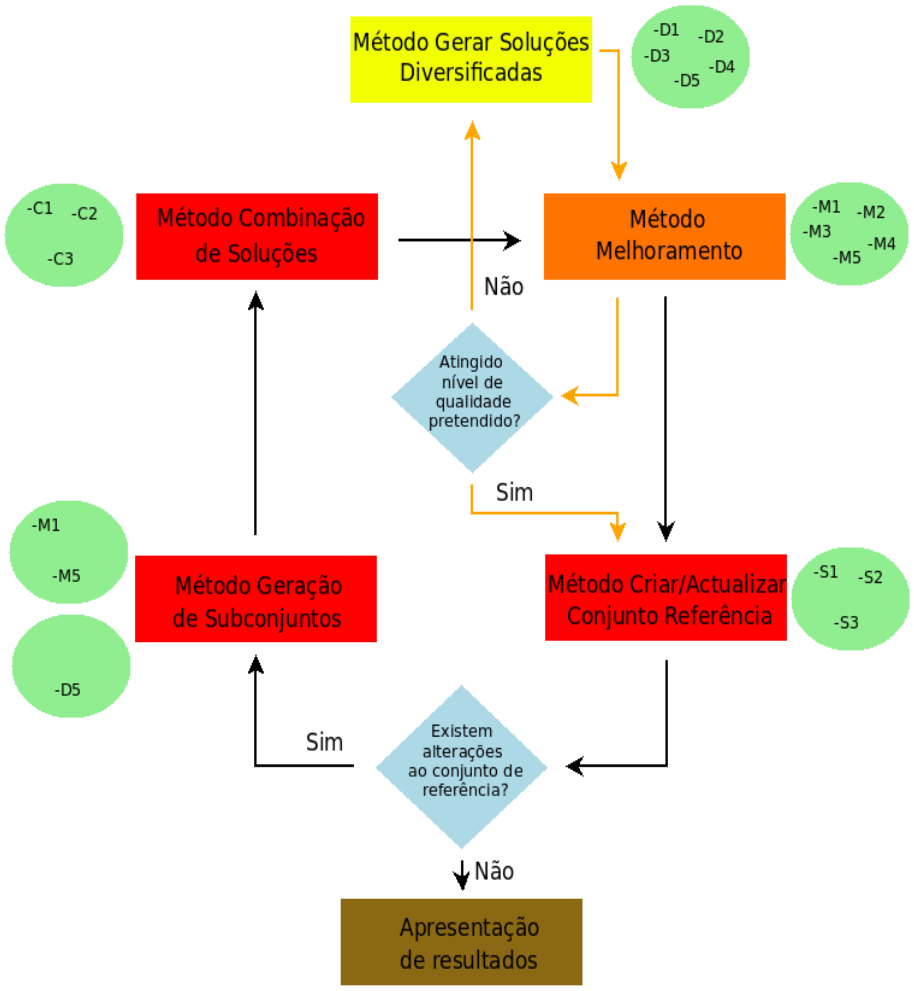

Figura 4.2: UFLP: Modelo do procedimento Pesquisa por Dispersão

Método de Geração de Soluções Diversificadas

Este método tem como objetivo gerar soluções diversificadas, de forma ao algoritmo poder passar por várias soluções alternativas e conseguir encontrar uma boa solução.
É utilizado um método pseudo-aleatório para a geração das soluções aleatórias, de forma a que a cada nova execução do algoritmo, as soluções aleatórias sejam sempre as mesmas.
Na figura 4.3 é apresentada uma descrição genérica do método.

Gerar soluções diversificadas

Passo 1. Inicializar vetor de soluções.

Passo 2. Inicializar semente para pseudo-aleatório.

Passo 3. Gerar instalação.

Passo 4. Se a instalação ainda não existe na solução, adicionar à solução.

Passo 5. Se ainda não foi atingido o número de instalações abertas por solução, ir para **Passo 3**.

Passo 6. Se ainda não foi atingido o número total de soluções, ir para **Passo 2**.

Passo 7. Terminar.•

Figura 4.3: UFLP: Método para gerar soluções diversificadas

Método de Melhoramento

O método tem por base a abordagem de Pesquisa Tabu proposta por Michel e Hentenryck [22]. O objetivo é melhorar as soluções diversificadas, bem como também as que resultem da combinação de soluções. Este método faz uso de uma estrutura de vizinhança que consiste em abrir ou fechar um armazém e utiliza uma lista tabu que vai guardando os movimentos que vão sendo efetuados, marcando-os como movimentos tabu, durante um determinado número de iterações.

Na figura 4.4 é apresentado o modelo para o método de melhoramento: As listas de armazéns fechados e abertos, $FechadasInstalacoes^+$ e $AbertosInstalacoes^+$, são ordenadas por ordem crescente de custo e as funções **Ganho**(S^+), $S^+ \in FechadasInstalacoes^+$, e **Ganho**($S^-$), $S^- \in AbertosInstalacoes^+$, retornam o menor custo, tanto dos armazéns abertos como dos que estão fechados. A função **Ganho**(S) retorna verdadeiro se houver uma solução na vizinhança de S que melhora a solução S, isto é se nas listas $FechadasInstalacoes^+$ e $AbertosInstalacoes^+$ existe algum movimento que melhor a melhor solução até ao mo-

mento. A função **Melhor**(S^+) e **Melhor**(S^-) retorna o melhor custo de cada uma das listas que piore menos a solução e que não faça parte da lista tabu.

Método de Melhoramento

Passo 1. Obter uma solução inicial admissível, S_0 com valor da função objetivo vs_0.

Passo 2. Inicializar a solução $S' = S = S_0$, $vs' = vs = vs_0$.

Passo 3. Inicializar $tabuLength = 3$, $maxIter = 10$, $iter = 0$ e $noImprove = 0$.

Passo 4. Se $noImprove \geq maxIter$, ir para **Passo 13**.

Passo 5. $vs_{old} = vs$.

Passo 6. Se !Ganho(S), ir para **Passo 10**.

Passo 7. Se Ganho(S^+) $\geq$ Ganho(S^-), então abre a melhor instalação do conjunto $FechadasInstalacoes^+$.

Passo 8. Se Ganho(S^+) $<$ Ganho(S^-), então fecha a melhor instalação do conjunto $AbertosInstalacoes^+$.

Passo 9. Se $vs \leq vs'$ fazer $vs' = vs$, $S' = S$ e $noImprove = 0$.

Passo 10. Se Melhor(S^+) $\geq$ Melhor(S^-), então abre instalação, senão fecha instalação.

Passo 11. $noImprove = noImprove + 1$.

Passo 12. Voltar ao passo **Passo 4**.

Passo 13. Terminar.•

Figura 4.4: UFLP: Método de melhoramento

O método tem a capacidade de explorar com maior intensidade uma região do espaço de soluções que se verifique como promissora — intensificação, bem como também fugir e explorar outras regiões admissíveis que ainda não tenham sido exploradas — diversificação.

Método para Criar e Atualizar o Conjunto de Referência

Método para criar e atualizar o conjunto de referência (Refset). A partir do conjunto de soluções diversificadas, extrai-se o conjunto de referência que é constituído por dois subconjuntos, um com soluções de qualidade e outro com soluções diversificadas.
Na figura 4.5 é apresentado um procedimento genérico deste método.

Criar e Atualizar o Conjunto de Referência

Passo 1. Inicializar o conjunto de referência (Refset).

Passo 2. Se a solução ainda não existe no conjunto de referência, adiciona-a.

Passo 3. Se ainda não atingiu o número de soluções total, voltar para o **Passo 2**.

Passo 4. Terminar.•

Figura 4.5: UFLP: Método para criar e atualizar o conjunto de referência

Geração de Subconjuntos

O método de geração de subconjuntos especifica a forma como são selecionados os subconjuntos de soluções aos quais será aplicado o método de combinação.
O grande objetivo deste método é conseguir encontrar o equilíbrio entre a diversificação e a intensificação, isto é, a escolha do número de soluções diversificadas e das soluções melhoradas para proceder à combinação das soluções. Se queremos diversificar as soluções, teremos que incluir mais soluções diversificadas e se pretendemos intensificar temos que incluir mais soluções melhoradas.
Na figura 4.6 é apresentado um algoritmo simples para este método.

Gerar os Subconjuntos

Passo 1. Inicializar o subconjunto.

Passo 2. Verificar as soluções melhores/piores a escolher.

Passo 3. Se a solução ainda não existe no subconjunto, adiciona-a.

Passo 4. Passar à solução seguinte a adicionar.

Passo 5. Se ainda não atingiu o número de soluções, voltar para o **Passo 3**.

Passo 6. Terminar.•

Figura 4.6: UFLP: Método para gerar os subconjuntos

Combinação de Soluções

Este método tem como objetivo combinar as soluções escolhidas anteriormente. O método combinará as soluções com base em rácios, de forma a que cada solução tenha um peso diferente dependente do valor da função objetivo para essa solução. Este parâmetro também irá permitir aplicar as técnicas de diversificação e intensificação, faladas anteriormente.
Na figura 4.7 apresenta-se um possível algoritmo para este método.

Combinar as Soluções

Passo 1. Inicializar o conjunto que guardará as soluções combinadas.

Passo 2. Obter as soluções a combinar.

Passo 3. Calcular rácios para cada solução.

Passo 4. Combinar soluções e adicionar ao conjunto das soluções combinadas.

Passo 5. Se ainda não atingiu o número de soluções total a combinar, voltar para o **Passo 2**.

Passo 6. Terminar.•

Figura 4.7: UFLP: Método para combinar as soluções

4.4 Descrição do algoritmo PD-RAMP

Nesta secção descreve-se o algoritmo PD-RAMP aqui proposto para o UFLP, que explora ambos os lados do problema (primal e dual) de forma intensiva.
O PD-RAMP tem como objetivo alternar entre ambos os lados do problema, de forma a guiar o melhoramento das soluções e conseguir atingir bons resultados. Ambos os lados partilham as informações recolhidas, para que ao alternar consigam tomar os melhores caminhos e fugir de caminhos já explorados.

4.4.1 Método Dual

O método dual é basicamente o mesmo do apresentado em Gamboa e Rego [3], por isso, optou-se por não se descrever aqui detalhadamente, uma vez que não acrescenta valor ao trabalho desenvolvido.

4.4.2 Método Primal

No lado primal é necessário um método de projeção, que tal como o nome indica, tem como objetivo fazer a projeção das soluções que o método dual fornece para o espaço de soluções do primal. Após a projeção, as soluções são melhoradas através do método de melhoramento. Pelo mesmo motivo, referido acima, não se descreve com detalhe o método de projeção.

O lado primal do problema é explorado intensivamente pelo método de pesquisa por dispersão falado anteriormente. De realçar que não utiliza a versão tal e qual como é descrita anteriormente, mas sim uma versão mais leve e simples, uma vez que se explora ambos os lados do problema e é necessário mais tempo de processamento. Como tal, foi simplificada a exploração do lado primal, com o objetivo de reduzir o tempo de execução, sendo o conjunto de referência constituído por menos soluções "primais", isto é, as soluções que são geradas pelo algoritmo pseudo-aleatório. Deste modo, são geradas menos soluções diversificadas e consequentemente melhoradas para o lado primal.

4.5 Estruturas de Dados

De seguida são descritas as estruturas de dados representadas na figura 4.8.
Temos uma matriz de custos, com os custos associados ao serviço de cada cliente por cada armazém. Um vetor com os custos de abertura de cada armazém e um vetor de listas duplamente ligadas, onde para cada um dos clientes são guardados, por ordem crescente de custo, os custos de servir para cada um dos armazéns abertos.

Esta última estrutura é de grande utilidade pois permite fazer a atribuição direta de cada um dos clientes aos armazéns, isto é, uma vez que o problema não tem restrições de capacidade é só percorrer cada um dos clientes e verificar qual o armazém à cabeça, sendo esse o que fica a servir o cliente. Quando é efetuado um movimento (fechar ou abrir armazém) essa estrutura é atualizada, de forma a que a mesma esteja sempre coerente com a solução que estamos a trabalhar e possamos fazer a atribuição direta dos clientes pelos armazéns abertos.

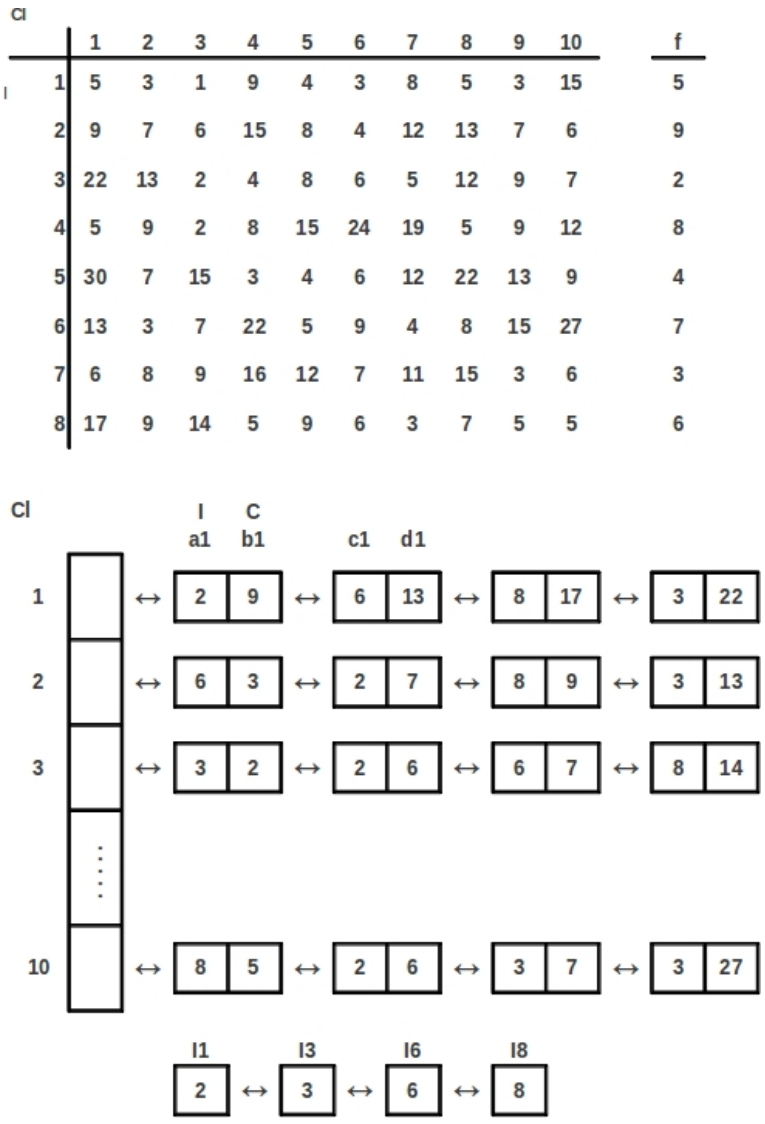

Figura 4.8: UFLP: Estruturas de dados

4.6 Resultados Computacionais

Foram realizados testes computacionais, utilizando os seguintes problemas de teste: 15 instâncias da biblioteca ORLIB propostos por Beasley [28], 22 problemas apresentados por Katrica, Filipovic e Ljubic [23] (M*), 60 instâncias pertencentes a dois grupos de problemas criados por Kochetov [29] (FPP) e 90 instâncias apresentados por Diptesh Ghosh [30] (GHOSH).

Para melhor se perceber o potencial das duas versões do algoritmo implementado, é feita uma comparação com os melhores resultados apresentados pelos melhores algoritmos conhecidos para o UFLP. Os algoritmos são todos híbridos e utilizam metaheurísticas como *Pesquisa Tabu*, *Algoritmos Genéticos* e *Pesquisa por Dispersão*. Os algoritmos utilizados para fazer a comparação são, H-RW proposto por Resende e Werneck em 2006 [24], RAMP apresentado por Gamboa em 2008 [3] e PBS proposto por Pullan em 2009 [25].

As duas versões propostas foram ambas implementadas em linguagem C, mas os seus resultados foram obtidos em máquinas distintas, sendo o SS executado numa máquina com o sistema operativo Linux, com as seguintes caraterísticas, processador Intel i5 2,2 GHz, com 4 GB de memória RAM e 7 GB de memória swap, já para o PD-RAMP os resultados foram obtidos num Intel Core 2 Duo 2.5 GHz, com 4 GB de memória RAM e 7 GB de memória swap, com sistema Linux, sendo que apenas foi utilizado um processador nos resultados obtidos em ambos os algoritmos. Os restantes algoritmos foram testados nas seguintes máquinas: o RAMP foi numa máquina com processador Intel Centrino a 1.5 GHz, ambos os algoritmos H-RW e PBS foram executados numa máquina AMD Opteron 252 2.6 GHz com 4 GB de memória RAM, uma vez que se consideram os resultados do algoritmo H-RW fornecidos em [25]. Pullan correu o algoritmo proposto por Resende e Werneck [24] com algumas alterações, tendo sido obtidos os resultados apresentados na tabela 4.1, à exceção das instâncias ORLIB, cujos resultados foram retirados do original proposto por Resende e Werneck [24].

A comparação entre os melhores resultados conhecidos é apresentada na tabela 4.1, onde estão identificados os desvios e os tempos de processamento respetivos.

	H-RW		RAMP		PBS		**SS**		**PD-RAMP**	
Classe	Dev.[a]	CPU[b]	Dev.[a]	CPU[b]	Dev.[a]	CPU[b]	Dev.[a]	CPU[b]	Dev.[a]	CPU[b]
ORLIB	0	0.048	0	0.93	-	-	0	0.21	0	1.68
M*	0	0.590	0	17.3	0	0.16	0	4.83	0	37.93
FPP	0	94.430	0.037	2.92	0	12.26	0.015	1.67	0	8.94
GHOSH	0	26.280	0.002	53.115	0	15.34	0.001	9.19	0.001	41.26
Média	0	30.337	0.020	21.723	0*	9.253*	0.005	4.460	≈ 0	22.450

[a] Desvio em relação ao ótimo em %
[b] Tempo de processamento em segundos
* Média efetuada apenas para as 3 instâncias (M*, FPP, GHOSH)

Tabela 4.1: UFLP: Resultados dos melhores algoritmos conhecidos

Podemos verificar que ambos os algoritmos SS e PD-RAMP conseguem ser competitivos com os restantes. Verificamos que o, SS, consegue obter bons resultados em tempos bastante reduzidos, quando comparados com o algoritmo mais complexo, PD-RAMP. Por sua vez, o PD-RAMP consegue obter melhores resultados tendo de média um valor muito próximo de zero (0,00025) para o desvio em relação ao ótimo. Se compararmos com o melhor algoritmo da literatura, PBS, podemos verificar que ambas as versões são competitivas, conseguindo a versão mais simples consumir menos tempo de CPU do que o algoritmo PBS.

Tendo em conta que o algoritmo RAMP, proposto por Gamboa e Rego [3], é a versão mais simples da abordagem RAMP, e o algoritmo PD-RAMP é uma versão mais complexa, podemos verificar que a *framework* RAMP é muito robusta e com grande flexibilidade para se adaptar às necessidades e aos objetivos que se pretendem alcançar, podendo ser efetuados incrementos de forma a alcançar melhores resultados.

Capítulo 5

Problema de Localização de Hubs com Afectação Múltipla e Sem Restrições de Capacidade – UMAHLP

5.1 Introdução

O problema de localização de hubs é um problema NP-difícil [7][8]. É constituído por um conjunto N de localizações de nós e por um conjunto H de r potenciais localizações de hubs. O problema tem associado um custo de transporte C_{ij} por unidade de fluxo entre os nós i e j e uma quantidade de fluxo W_{ij} do nó origem i para o nó destino j.
O caminho do nó de origem i para o nó destino j inclui três componentes:

- coleção — do nó origem i para o hub k a que está alocado;
- transferência — entre o hub k ao qual o nó i está alocado, e o hub m ao qual o nó j está alocado;
- distribuição — do hub m para o nó destino j a que está alocado.

O custo por unidade de fluxo para a rota $(i \to k \to m \to j)$ é $C_{ikmj} = \chi C_{ik} + \alpha C_{km} + \delta C_{mj}$, onde χ, α e δ são os componentes de custo, coleção, transferência e distribuição. Geralmente o α é mais pequeno que o χ e o δ, sendo um fator de desconto entre ligações de hubs. O'Kelly [7] propôs $\alpha < 1$ para representar a economia de escala no custo de transporte entre hubs. Ernst e Krishnamoorthy [31] [32] introduziram os outros dois fatores, χ e δ, de forma a representar a realidade dos custos de serviço postal (correios), representando

o custo de recolher o correio e de o distribuir. Cada nó definido como hub tem associado um custo fixo, sendo representado por f_k.

O UMAHLP pode ser definido pela formulação de programação linear inteira, representada na figura 5.1:

Formulação de Programação Linear Inteira – UMAHL-IP1

$$Min \quad \sum_i \sum_j \sum_k \sum_m C_{ikmj} x_{ikmj} + \sum_k f_k y_k \tag{5.1}$$

$s.a$

$$\sum_k \sum_m x_{ikmj} = 1, \qquad \forall i, j, \tag{5.2}$$

$$x_{ikmj} \leq y_k \qquad \forall i, j, k, m, \tag{5.3}$$

$$x_{ikmj} \leq y_m \qquad \forall i, j, k, m, \tag{5.4}$$

$$y_k \in \{0, 1\} \qquad \forall k, \tag{5.5}$$

$$x_{ikmj} \geq 0 \qquad \forall i, j, k, m. \tag{5.6}$$

Figura 5.1: UMAHLP: Formulação de Programação Linear Inteira – 1

Nesta formulação, a função objetivo (5.1) tem como meta minimizar o custo total, determinando a localização dos hubs e alocar os restantes nós aos hubs, tendo em conta os custos de coleção, transferência, distribuição e o custo fixo de alocar o nó como hub, de forma a encaminhar o tráfego entre o par de nós, origem — destino.

A restrição (5.2), mais concretamente a variável x_{ikmj}, impõe que para um par de nós (i — origem, j — destino) apenas existe uma rota, enquanto que a (5.3) e (5.4) garantem que o fluxo é apenas enviado por nós alocados como hubs. A restrição (5.5) representa os nós k que são alocados como hubs, assumindo o valor 0 (nó não alocado como hub) e o valor 1 (nó alocado como hub). A restrição (5.6) garante que a variável x_{ikmj}, seja não negativa. Esta variável define se a rota entre o par de nós (i e j) passa pelos hubs (k e m), em conjunto com a restrição (5.2) obriga a que só exista uma rota para cada par de nós (i e j).

Consideremos agora uma formulação alternativa proposta por Klincewicz [33], que é equivalente à anterior mas tem a vantagem de permitir uma melhor perceção da posterior formulação do problema dual.
Os índices i e j de pares ordenados de nós (i,j) é substituído pelo índice único h e os índices k e m de pares ordenados de hubs (k,m) são substituídos pelo índice único l. É introduzido o conjunto $H_l = \{k|l = (k,m) \vee l = (m,k)\}$, que contém os hubs que estão presentes no par de hubs l. O custo C_{ikmj} para os pares de nós e pares de hubs é substituído por C'_{hl}. O UMAHLP pode então ser formulado como representado na figura 5.2:

Formulação de Programação Linear Inteira – UMAHL-IP2

$$Min \qquad \sum_h \sum_l C'_{hl} x_{hl} + \sum_k f_k y_k \tag{5.7}$$

$s.a$

$$\sum_l x_{hl} = 1, \qquad \forall h, \tag{5.8}$$

$$x_{hl} \leq y_k \qquad \forall h, l, ek \in H_l, \tag{5.9}$$

$$y_k \in \{0,1\} \qquad \forall k, \tag{5.10}$$

$$x_{hl} \geq 0 \qquad \forall h, l. \tag{5.11}$$

Figura 5.2: UMAHLP: Formulação de Programação Linear Inteira – 2

Nesta formulação, o índice h assume os valores do conjunto $M = N \times N$, enquanto que o índice l toma valores do conjunto $L = H \times H$. A variável x_{hl} representa o tráfego entre os nós, formados pelo par de nós h e o par de hubs l. As restrições (5.8) a (5.11) são análogas às restrições (5.2) a (5.6). A única diferença reside no facto das restrições (5.3) e (5.4) terem dado origem a uma única restrição (5.9). A restrição (5.10) também poderia ser substituída por:

$$y_k \geq 0 \qquad \forall k.$$

Apresentado o problema, podemos acrescentar que o UMAHLP é uma versão mais simples do *Capacitated Multiple Allocation Hub Location Problem* (CMAHLP).
Para o problema *Uncapacitated Hub Location Problem* (UHLP) ainda existe outro tipo de redes de hubs, a alocação singular. A diferença entre os dois tipos está nos hubs.

Na alocação singular, o encaminhamento do tráfego tem apenas uma rota para o par de nós (origem — destino), ou seja cada nó apenas está alocado a um único hub.
Já na alocação múltipla podem existir várias rotas para os pares de nós, pois um nó pode estar alocado a mais do que um hub, tendo sempre o objetivo de minimizar o custo total.

Para estes dois tipos de problemas, existem ainda duas versões, onde é definido à priori o número exato de hubs a serem atribuídos, designados por *Uncapacitated Single Allocation p-Hub Median Problem* (USApHMP) e *Uncapacitated Multiple Allocation p-Hub Median Problem* (UMApHMP).
Os problemas de localização de hubs assumem três pressupostos:

1. A rede de hubs é completamente ligada, isto é, todo o par de hubs tem ligação entre si;

2. Atendendo a economias de escala é considerado um fator de desconto (símbolo alfa) nas ligações entre os hubs;

3. Não é permitido o serviço direto entre nós, têm obrigatoriamente de passar por um ou dois hubs.

Podemos então concluir que uma rota entre um par de nós (origem — destino) tem que passar obrigatoriamente por um ou dois hubs.
Consideremos o exemplo presente na figura 5.3:

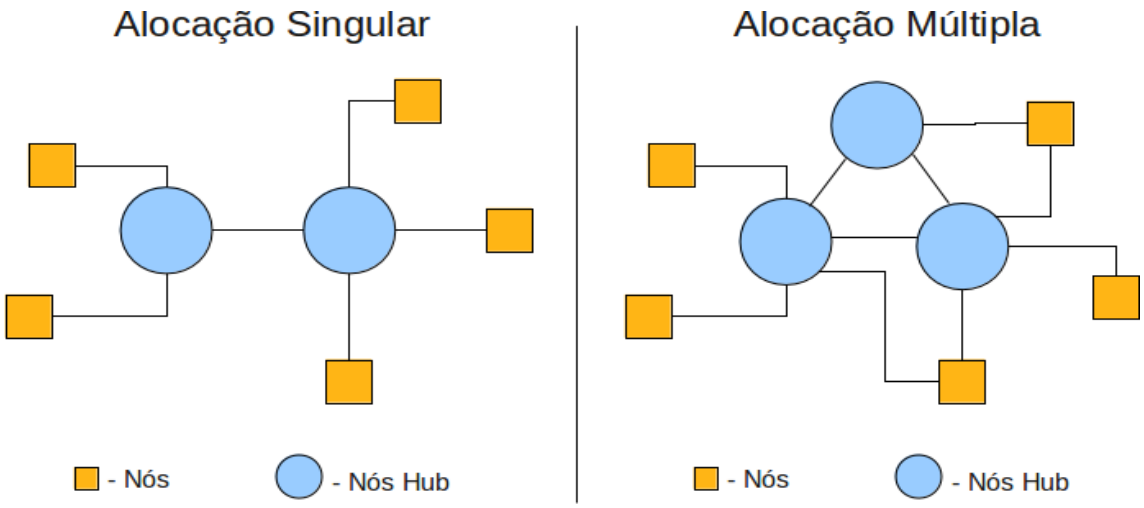

Figura 5.3: UMAHLP: Exemplo – Problema de localização de hubs com alocação singular e múltipla

No exemplo da figura 5.3, estão representados dois possíveis exemplos dos dois tipos de alocação que existem no HLP. Na imagem da esquerda está representada a alocação singular, onde cada nó apenas está alocado a um único hub. Na imagem da direita é apresentado um exemplo de alocação múltipla, onde cada nó pode estar alocado a mais do

que um hub. Em ambas as imagens constatamos que os hubs são completamente ligados entre si, sendo uma caraterística do HLP.

Os problemas de localização de hubs têm várias áreas de aplicação, tais como, transportes (aéreos de passageiros, terrestres de carga e correio postal), telecomunicações (redes de telefones e vídeo conferência), redes de computadores, entre outras.

5.2 Algoritmos Heurísticos para o UMAHLP

O problema de localização de hubs é um dos problemas mais estudados pela comunidade científica, sendo que nos últimos anos tem assumido um grande foco, e sofrido um grande crescimento. Ao longo destes anos têm sido propostos vários algoritmos. De seguida são apresentados alguns dos algoritmos mais importantes existentes na literatura para o UMAHLP.

Campbel [34], em 1994, propôs a primeira formulação de programação linear para o problema de localização de Hubs, com e sem restrições de capacidade, e para a alocação singular e múltipla.

Em 1996, Klincewicz [33] apresentou um algoritmo que aplica o método *Dual-Ascent* e *Dual Adjustment* em conjunto com o método *branch-and-bound*. Ambos os métodos *Dual-Ascent* e *Dual Adjustment* exploram o lado dual do problema, enquanto que o método *branch-and-bound* explora o lado primal do problema. O método *Dual-Ascent* utilizado tem por base o apresentado por Erlenkotter [18] ao UFLP – problema de localização de instalações sem restrição de capacidades.

Mayer e Wagner [35], em 2002, implementaram um novo algoritmo com base no método *branch-and-bound* falado anteriormente, designado por "HubLocator". O algoritmo utiliza o *Dual-Ascent* para obter limites inferiores. A vantagem em relação ao proposto por Klincewicz [33] é que o valor desses limites é mais pequeno e assim reduz o tempo computacional de execução do método *branch-and-bound*.

Em 2004, Boland et al. [36] verificaram que apesar da formulação proposta por Ernst e Krishnamoorthy [32] para o UMApHMP, ter tempos computacionais pequenos e requerer pouca memória, os limites inferiores são um aspeto a melhorar. De acordo com essa deficiência, os autores identificaram algumas caraterísticas das soluções ótimas para implementar técnicas de pré-processamento e reduziram as restrições.

Ainda no mesmo ano, Hamacher et al. [37] propuseram um estudo poliédrico sobre o problema. Os autores determinaram a dimensão e derivaram algumas classes das faces do poliedro. De seguida, implementaram uma regra geral para fazer o levantamento das faces do UFLP, para este este problema (UMAHLP), desenvolvendo uma nova formulação cujas restrições são as que eles definem.

Marín [38] propôs, em 2005, algumas desigualdades válidas de definição de faces para o UHLP, com custos que satisfazem a desigualdade do triângulo, que representa os custos de transporte entre hubs. O autor resolveu o problema com um algoritmo baseado em *relax-and-cut*.
O mesmo Marín em conjunto com outros autores [39], em 2006, apresentaram uma nova formulação que é uma generalização de formulações anteriores e o pressuposto de ter uma estrutura de custos que satisfaz a desigualdade do triângulo. Usando alguns resultados poliédricos, conseguiram reduzir o número de restrições do problema.

Em 2007, Cánovas et al. [40] propuseram uma nova heurística com base no *dual-ascent*, juntamente com o método *branch-and-bound*. Os autores testaram o algoritmo usando os problemas de teste *Civil Aeronautics Board (CAB)* e *Australian Post (AP)*, obtendo bons resultados em tempos reduzidos.

Camargo et al. [41], em 2008, apresentaram um algoritmo com base na formulação de Hamacher [37] e utilizaram a abordagem *Benders* [42] para resolver o problema. A abordagem *Benders* divide o problema em dois problemas mais simples: um de nível superior designado por MP (problema principal), que determina os hubs que são escolhidos e outro de nível inferior conhecido como SP (subproblema) que define a alocação dos nós aos hubs. O MP é uma versão relaxada do problema original com o conjunto de variáveis inteiras e as restrições associadas. O SP é o problema original com os valores das variáveis inteiras temporariamente fixas para o MP, mais concretamente o problema de transportes.

5.3 Descrição do Algoritmo

De seguida é descrito o algoritmo RAMP proposto para resolver o problema UMAHLP. O algoritmo é composto por dois procedimentos principais, a resolução do problema dual, utilizando o método *Dual-Ascent* e a resolução do lado primal, utilizando o método de pesquisa tabu. O algoritmo começa por determinar uma solução dual admissível para o problema dual. Posteriormente projeta essa solução dual no espaço de soluções do primal, sendo-lhe aplicado um método de melhoramento, com base no método de pesquisa tabu.

O processo repete-se iterativamente, até ser atingido um número de iterações consecutivas sem melhorar a melhor solução encontrada.

5.3.1 Método Dual

Na abordagem aqui proposta, utiliza-se a relaxação e dualização usada por Mayer e Wagner [35] no algoritmo HubLocator. O HubLocator aplica o procedimento *Dual-Ascent* a uma formulação condensada do problema dual, e posteriormente o procedimento *branch-and-bound*.
A abordagem RAMP aqui descrita, utilizou o procedimento *Dual-Ascent*, uma vez que, a sua integração no algoritmo RAMP forneceu logo bons resultados.

A formulação condensada do problema dual (UMAHL-D) está representada na figura 5.4 e obtém-se como se segue. Começa-se por obter a relaxação de programação linear inteira do problema UMAHL-IP2, associando as variáveis duais v_h e w_{khl} às restrições (5.8) e (5.9) respetivamente, obtemos seguinte formulação do problema dual:

Formulação condensada do problema dual – UMAHL-D

$$Max \qquad \sum_{h} v_h \tag{5.12}$$

$s.a$

$$v_h - \sum_{k \in H_l} w_{khl} \leq C'_{hl}, \qquad \forall h, l, \tag{5.13}$$

$$\sum_{h} \sum_{l \in P_k} w_{khl} \leq f_k \qquad \forall k, \tag{5.14}$$

$$w_{khl} \geq 0 \qquad \forall k, h, l. \tag{5.15}$$

Figura 5.4: UMAHLP: Formulação condensada do problema dual

O conjunto P_k contém todos os pares de hubs l, onde o hub k está presente. A função objetivo da formulação do problema dual apenas depende da variável v_h. Desta forma, caso estas variáveis sejam conhecidas à priori, à variável w_{khl} pode ser atribuído qualquer valor, desde que respeite as restrições (5.13) e (5.14).
O objetivo é maximizar o valor de v_h, de forma a respeitar as restrições (5.13) e (5.14), de modo a que a solução seja admissível.

De seguida, descreve-se o procedimento *Dual-Ascent*.
Antes de se iniciar a execução do método *Dual-Ascent*, são escolhidos os nós mais promissores a serem definidos como hubs.
A forma como são determinados os hubs mais promissores tem por base a distância entre os nós, ou seja, é feita a média de todas as distâncias entre os vários nós. De seguida para cada um dos nós verifica-se qual dos restantes nós se encontram a uma distância inferior à média total das distâncias, sendo guardado os nós, e o número total de nós em que a distância é inferior à distância média total.
Posteriormente, percorrem-se os nós, por ordem decrescente do total de nós inferiores à distância média, e verifica-se para cada nó, quais os nós que não estão perto. Sabendo os nós que não estão perto, são analisados esses nós e os respetivos nós que estão perto, de modo a se verificar os nós que conseguem cobrir todos os restantes nós com uma distância inferior à média. Na figura 5.5 é apresentado um exemplo.

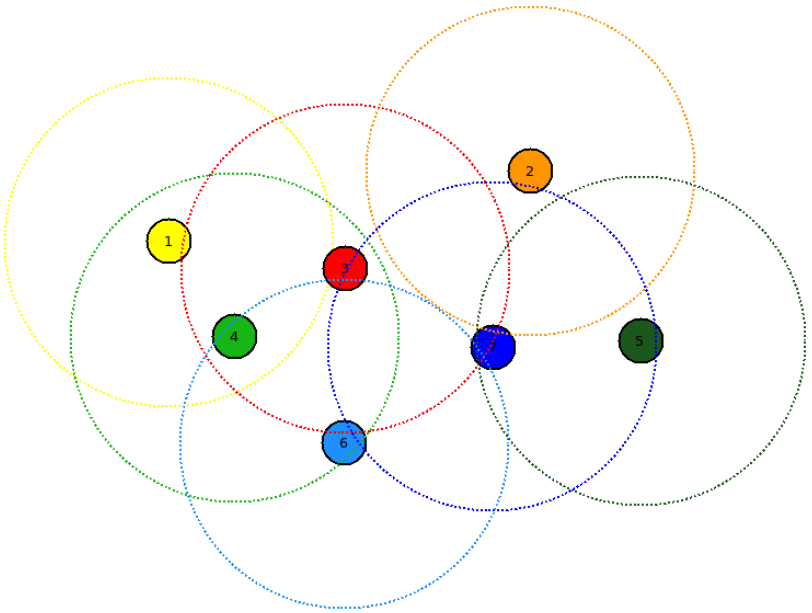

Figura 5.5: UMAHLP: Exemplo dos nós que estão mais perto

Para o exemplo representado na figura 5.5, podemos verificar para cada um dos nós, os nós que estão mais perto. O raio de ação é igual para cada um dos nós, sendo equivalente à distância média total entre os nós, considerada no exemplo. De seguida são indicados para cada um dos nós, os nós que estão mais perto.

Para o nó 1, estão perto os nós 3 e 4; Para o nó 2, está perto o nó 7; O nó 3 tem perto dele os nós 1, 4, 6 e 7; Para o nó 4, estão perto os nós 1, 3 e 6; O nó 5, está perto do nó 7; Do nó 6 estão perto, o nó 3, 4 e 7; E por fim o nó 7, está perto dos nós 2, 3, 5, 6.

Neste exemplo podemos verificar que no mínimo dois nós, nomeadamente os nós 3 e 7, são necessários para cobrir toda a rede com a distância inferior à média total. Também com outras combinações de nós podemos cobrir toda a rede, por exemplo, considerando definidos os conjuntos de hubs {4, 6, 7}, {1, 4, 7}, {3, 4 e 7}, entre outros.

De seguida, são então escolhidos os r potenciais nós para serem definidos como hubs, tendo por base o conceito apresentado anteriormente. É dada maior prioridade aos nós que conseguem cobrir toda a rede com distâncias inferiores, com número mínimo de nós, sendo neste exemplo apenas os nós 3 e 7. Deste modo, apenas estes conseguem cobrir toda a rede com apenas 2 hubs definidos, as restantes combinações para cobrir toda a rede para o exemplo dado necessitam de mais do que 2 hubs definidos. Os restantes nós têm maior prioridade de ser atribuídos como hubs, tendo por base os seus fluxos e distâncias.

Posteriormente, é construída a matriz de custos, tendo como base todas as soluções combinadas dos pares de hubs promissores l, com os pares de nós h. Esta matriz de custos engloba todas as rotas possíveis entre os pares de nós e os hubs escolhidos.

Depois de todos os passos anteriores estarem terminados, o procedimento *dual-ascent* é iniciado. O procedimento começa por construir a matriz transposta dos custos $C_h'^q$, isto é, os valores de C'_{hl} para todo o h são ordenados de forma crescente para $q = 1, ..., r^2$ e $C_h'^q = +\infty$ para $q = r^2 + 1$.
A variável v_h é inicializada com os menores custos da matriz transposta de custos $C_h'^1$.
Posteriormente são percorridos repetida e sequencialmente todos os pares de nós h, tentando aumentar o valor de v_h para o seguinte valor mais elevado de C'_{hl}. De realçar que são percorridos apenas os pares de nós h, para quais o v_h possa ser aumentado, isto é, quando em algum momento, não for possível aumentar v_h esse h não volta a ser considerado. O procedimento termina quando não é possível aumentar mais nenhum v_h.

Na figura 5.6 está representado o algoritmo *Dual-Ascent* implementado.
No passo 4, o problema de transportes é resolvido por um algoritmo heurístico que tem a vantagem de consumir menos recursos computacionais e ser mais rápido que o método clássico de resolução do problema de transportes. De seguida é explicado como são obtidos os valores das variáveis necessárias para resolução do problema de transportes.

Sendo o problema de transportes utilizado para verificar a integridade das restrições (5.13) e (5.14), e uma vez conhecido o valor de v_h, a restrição (5.13) equivale à procura e a restrição (5.14) equivale à capacidade. Posto isto, apenas ficam a faltar os custos para a resolução do problema de transportes. Esta matriz de custos é constituída por r (potenciais hubs) linhas e $R = n^2r^2$ colunas ($r \times R$). Para cada hub potencial k, existe

Procedimento *Dual-Ascent*

Passo 1. Começar com uma solução dual admissível v_h, tal que $v_h \geq C'^1_h$. Inicializar $g = 1$ e $w_{khl} = 0 \qquad \forall h \in M, l \in L, k \in H_l$.

Passo 2. Definir $J = P$ e $g = g + 1$.

Passo 3. Escolher um $h \in J$ e definir $v_h = C'^g_h$.

Passo 4. Resolver o problema de transportes.

Passo 5. Se custo retornado pelo problema de transportes for diferente 0, fazer $P = P - \{h\}$ e $v_h = C'^{g-1}_h$.

Passo 6. Fazer $J = J - \{h\}$.

Passo 7. Se o conjunto J não está vazio, então voltar ao **Passo 3**.

Passo 8. Se o conjunto P não está vazio, então voltar ao **Passo 2**.

Passo 9. Terminar.•

Figura 5.6: UMAHLP: Método *Dual-Ascent*

uma rota constituída por todas as combinações possíveis de nós h e hubs l (h, l). A capacidade S_k de cada k, para cada $k = 1, ..., r$ vai ser igual ao seu custo de alocação do nó como hub (f_k). A procura d_{hl} para a rota (h, l), onde (h, l) = 1, ..., R, vai ser igual ao max$\{0,\ v_h - C'_{hl}\}$. Os custos de transporte para $r \times R$ assumem o valor 0. A variável w_{khl} representa o tráfego de origem do hub k para o par destino (h, l). Para que o problema de transportes esteja totalmente correto é necessário que este esteja equilibrado. Para garantir que está equilibrado são adicionados um nó e um hub fictício. Para o nó fictício é também atribuída a procura, que vai ser igual ao $\sum_k S_k$. A capacidade do hub fictício vai ser igual ao $\sum_h \sum_l d_{hl}$. Aos custos de transporte é atribuído um custo positivo (ex: 1), exceto para a ligação entre o nó fictício e o hub fictício que assume valor 0.

Desta forma garantimos um problema de transportes equilibrado e asseguramos que se o valor retornado for igual a zero, as restrições (5.13) e (5.14) são respeitadas, caso contrário não são, e tem que se proceder ao decremento do v_h, de forma a tornar a solução admissível.

De seguida é descrito, de forma sucinta, o algoritmo heurístico do problema de transportes. O algoritmo em cada iteração do procedimento *Dual-Ascent*, verifica a procura necessária para o par de nós h para cada par de hubs l, verificando se existe capacidade nos hubs k e m. Caso exista capacidade, é necessário proceder às alterações da variável w_{khl}. Deste modo, para todos os índices l onde o d_{hl} foi aumentado Δ, são efetuadas as seguintes modificações:

Se $w_{k,R+1} \geq \Delta$, onde $l = (k, m)$ ou $l = (m, k)$:

- $w_{k,R+1} = w_{k,R+1} - \Delta$,
- $w_{khl} = w_{khl} + \Delta$,
- $w_{r+1,R+1} = w_{r+1,R+1} + \Delta$.

Senão se $w_{k,R+1} + w_{m,R+1} \geq \Delta$, onde $l = (k, m)$:

- $w_{m,R+1} = w_{m,R+1} - (\Delta - w_{k,R+1})$,
- $w_{mhl} = w_{mhl} + (\Delta - w_{k,R+1})$,
- $w_{khl} = w_{khl} + w_{k,R+1}$,
- $w_{k,R+1} = 0$,
- $w_{r+1,R+1} = w_{r+1,R+1} + \Delta$.

Na figura 5.7 é apresentado o procedimento heurístico implementado para a resolução do problema de transportes.
Acabada a execução do procedimento *Dual-Ascent* é iniciado o método primal.

5.3.2 Método Primal

Terminado o método dual é necessário projetar a solução dual encontrada no espaço de soluções do problema primal, de forma a que se possa aplicar o método de melhoramento. O procedimento *Dual-Ascent* fornece uma solução dual, a partir da qual é possível construir uma solução primal que satisfaça as variáveis de folga correspondentes à restrição (5.14) e à variável y_k:

$y_k(f_k - \sum_h \sum_{l \in P_k} w_{khl}) = 0$

Sendo conhecidos os valores da variável w_{khl}, onde cada posição corresponde à quantidade afeta para cada hub k nos pares h e l e a variável f_k representa o respetivo custo de abertura do hub k. Onde se verificar que o hub k esgotou a sua capacidade, isto é, a soma das afetações dos nós que lhe estão alocados for igual à sua capacidade, esses nós k são escolhidos para hubs.

Feita a projeção, são fechados os hubs que melhoram a solução, enquanto o número de hubs definido seja superior ou igual ao número de hubs que devem ser definidos. Isto é, todos os nós estejam a uma distância do hub alocado inferior à da média total.

Procedimento heurístico do problema de transportes

Passo 1. Inicializar $somaProcura = 0$, $ClientesProcura = 0$ e $ClientesSatisfeitos = 0$. Para cada $h \in M$ definir $procura(h) = 0$.

Passo 2. Se $l \notin L$, passar para o **Passo 6**.

Passo 3. Se $v_h > C_{hl}$, então $somaProcura = somaProcura + v_h - C_{hl}$.

Passo 4. $somaProcura = somaProcura - procura(h)$ e $procura(h) = somaProcura$.

Passo 5. Se $l \neq H \times H$, incrementar 1 a l e voltar ao **Passo 2**.

Passo 6. Se $somaProcura \leq restoProcura$, então $restoProcura = restoProcura - somaProcura$, senão retornar 1 e passar para o **Passo 14**.

Passo 7. Se $l \notin L$, passar para o **Passo 13**.

Passo 8. Definir $\Delta = v_h - C_{hl}$.

Passo 9. Se $\Delta > 0$, então $ClientesProcura = ClientesProcura+1$, senão passar para o **Passo 12**.

Passo 10. Se $W_{k,R+1} \geq \Delta$, fazer:

- $W_{k,R+1} = W_{k,R+1} - \Delta$;
- $W_{khl} = W_{khl} + \Delta$;
- $W_{r+1,R+1} = W_{r+1,R+1} + \Delta$;
- $ClientesSatisfeitos = ClientesSatisfeitos + 1$.

Passo 11. Se $W_{k,R+1} + W_{m,R+1} \geq \Delta$, fazer:

- $W_{m,R+1} = W_{m,R+1} - (\Delta - W_{k,R+1})$;
- $W_{mhl} = W_{mhl} + (\Delta - W_{k,R+1})$;
- $W_{khl} = W_{khl} + W_{k,R+1}$;
- $W_{k,R+1} = 0$;
- $W_{r+1,R+1} = W_{r+1,R+1} + \Delta$;
- $ClientesSatisfeitos = ClientesSatisfeitos + 1$.

Passo 12. Se $l \neq H \times H$, incrementar 1 a l e voltar ao **Passo 7**.

Passo 13. Se $ClientesProcura = ClientesSatisfeitos$, então retorna 0, senão retorna 1.

Passo 14. Terminar.•

Figura 5.7: UMAHLP: Método heurístico para resolução do problema de transportes

De seguida é aplicado o método de melhoramento à solução projetada.

O algoritmo de pesquisa local utilizado para melhorar a solução tem por base o método de pesquisa tabu. O algoritmo faz uso de uma estrutura de vizinhança que consiste em trocar o estado do nó, isto é, caso o nó esteja definido com hub, deixa de estar, e vice-versa.
O algoritmo também utiliza uma lista tabu, que vai guardando os movimentos (nós) tabu. Desta forma assegura que não são efetuados os mesmos movimentos, sendo o movimento retirado da lista tabu após algumas iterações, tendo como objetivo a passagem para outra zona do espaço de soluções.

Na figura 5.8 é apresentado o modelo do algoritmo de melhoramento implementado.
O algoritmo começa com uma solução inicial admissível, nomeadamente a retornada pelo método dual, com as alterações que foram descritas anteriormente. Em cada iteração o algoritmo verifica se existe algum movimento na estrutura de vizinhança que melhore a solução corrente, sendo isso verificado pela função **Ganho()**. Caso exista, é escolhido o movimento que melhora mais a solução, sendo o movimento retornado pela função **melhorTroca()**. De seguida é diminuída uma unidade no tamanho da lista tabu, caso exista algum movimento que melhore a qualidade da solução, se não existir nenhum movimento, é incrementado uma unidade ao tamanho da lista tabu, caso esta seja inferior a 10.

Posteriormente é escolhido um movimento, de forma aleatória, ou seja, caso estejam definidos mais do que um hub, é escolhido um hub para fechar. Se apenas estiver um hub definido, é escolhido um nó para ser definido como hub, sendo em ambos os casos o movimento escolhido de forma aleatória.

Para cada iteração é sempre verificado se a solução corrente S é melhor do que a melhor solução até ao momento S^*, e caso isso se verifique, a melhor solução até ao momento é atualizada, bem como o valor da função objetivo vs^*. Ainda para cada iteração do método, a lista tabu é atualizada, inserindo o movimento escolhido na lista. Caso a lista se encontre completa (o número de elementos é superior ao tamanho definido) é removido o movimento mais antigo. O método termina quando se atingir um determinado número de iterações consecutivas ($maxIter$) sem que a melhor solução seja melhorada.

Procedimento Método de Melhoramento

Passo 1. Começar com uma solução inicial S_0, com o valor da função objetivo vs_0.

Passo 2. Inicializar $S^* = S = S_0, vs^* = vs = vs_0$.

Passo 3. Inicializar $tabuLength = 3$, $maxIter = nNos$, $iter = 0$.

Passo 4. Se $iter \geq maxIter$, então ir para **Passo 17**.

Passo 5. $vs_{old} = vs$.

Passo 6. Se !Ganho(S) ir para **Passo 9**.

Passo 7. no = melhorTroca(S).

Passo 8. Se $vs < vs_{old}$ e $tabuLength > 2$, fazer $tabuLength = tabuLength - 1$ e passar para o **Passo 12**.

Passo 9. Se $vs \geq vs_{old}$ e $tabuLength < 10$, fazer $tabuLength = tabuLength + 1$.

Passo 10. Se número de hubs > 1, fazer no = random(AlocadosNoHub(S)).

Passo 11. Se número de hubs ≤ 1, fazer no = random(NosNaoAlocadosHub(S)).

Passo 12. $iter = iter + 1$.

Passo 13. Se $vs < vs^*$, fazer $vs^* = vs$, $S^* = S$ e $iter = 0$.

Passo 14. Adicionar movimento lista tabu.

Passo 15. Se tamanho lista tabu $> tabuLength$, então eliminar os movimentos mais antigos.

Passo 16. Voltar ao **Passo 4**.

Passo 17. Terminar.•

Figura 5.8: UMAHLP: Método de melhoramento

5.4 Estruturas de Dados

Para se conseguir o nível de eficiência desejado para o algoritmo implementado foram utilizadas estruturas de dados específicas. Na figura 5.9 estão representadas as estruturas de dados utilizadas, que descrevemos de seguida.

As estruturas de dados básicas consistem numa matriz de distâncias (para cada nó é guardada a distância a que se encontra dos restantes nós), uma matriz de fluxos (cada nó tem associado um fluxo que pretende enviar para cada um dos restantes nós), um vetor de custos fixos (representa o valor que vai ser adicionado ao valor da função objetivo caso

o respetivo nó seja alocado como hub) e uma lista com os nós que se encontram alocados como hubs.
Adicionalmente, temos um vetor de listas duplamente ligadas e ordenadas (nosPerto), onde para cada nó é armazenado um conjunto de nós que se encontram mais perto dele exceto o próprio, isto é, são armazenados os nós que se distanciam do nó a uma distância inferior à média total, por ordem crescente. Existe ainda, uma lista ordenada por ordem crescente (numNosPerto), com o número total de nós que estão perto de cada nó.
Existe também um vetor de apontadores para os nós que estão alocados como hubs (Hubs.ap), de forma a facilitar o acesso e a minimizar o tempo de manipulação.

A estrutura de dados mostrou ser eficiente para o problema que se está a tratar. Conseguiu-se minimizar o tempo no acesso aos dados ao utilizar a estrutura total em comparação com uma utilização que considere apenas a estrutura básica. No caso de ficheiros de testes de maiores dimensões, esta dinâmica é facilmente percetível, uma vez que existem muitos nós.

5.5 Resultados Computacionais

Para se avaliar o desempenho do algoritmo RAMP proposto para o UMAHLP, comparam-se os resultados obtidos com os melhores algoritmos existentes na literatura, para as instâncias de teste standard do problema. As instâncias de teste utilizadas para recolher os resultados foram, Australian Post (AP) propostas por Ernst et al. [31] em 1996, e Civil Aeronautics Board propostas por O'Kelly [7] em 1987.

O conjunto de testes AP, corresponde a um problema real de 200 distritos de códigos postais do correio Australiano. A partir desses dados de 200 distritos, são gerados problemas mais pequenos, sendo o número de nós dos ficheiros gerados, um múltiplo de cinco. São também considerados custos fixos pequenos (L) ou grandes (T) de acordo com o proposto por Ernst et al. [43]. Esta instância quando proposta por Ernst e Krishnamoorthy [31], assume os seguintes valores para os fatores rácio de coleção – χ, rácio de transferência – α e rácio de distribuição – δ: 3, 0.75 e 2, respetivamente, sendo que os autores consideram os fluxos assimétricos.
Em 2007, Cánovas et al. [40], obtiveram resultados para esses ficheiros e geraram subproblemas dos mesmos, com as seguintes alterações: consideraram os fluxos simétricos e os seguintes valores para os fatores, $\chi = 1$, $\delta = 1$, e α a tomar valores 0.1, 0.5 e 0.9.

As instâncias de teste CAB contêm informação acerca de 25 cidades dos EUA, com um elevado tráfego de passageiros das companhias aéreas, referente ao ano de 1970. De realçar

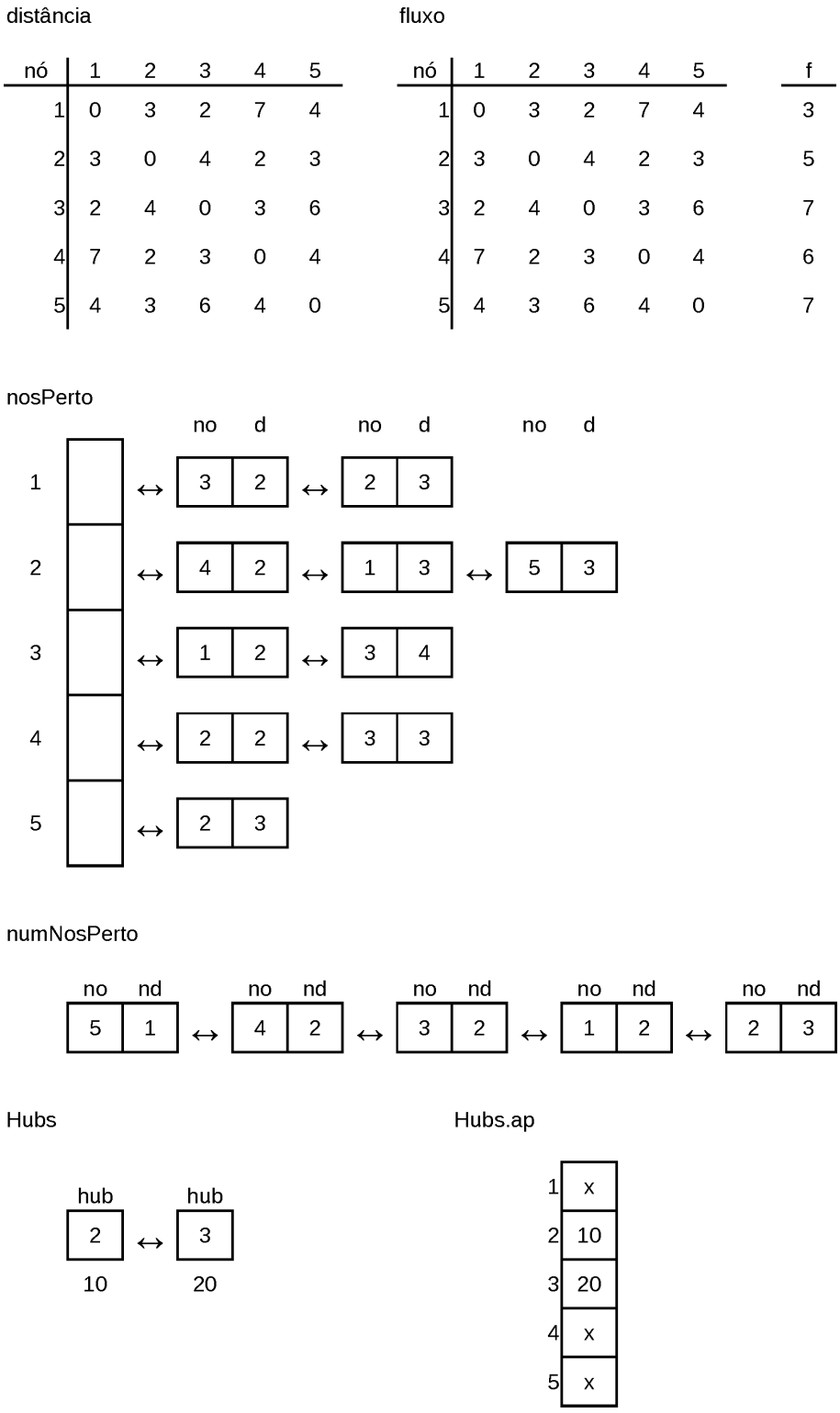

Figura 5.9: UMAHLP: Estruturas de dados

que os fluxos destas instâncias são simétricos.

Como já foi referido foram obtidos resultados, tendo sido contempladas todas as instâncias de teste. Para comparar os resultados obtidos é feita a comparação com os melhores resultados reportados na literatura, para estas instâncias de teste. De realçar, que apesar

do algoritmo proposto por Camargo et al. [41], ser um algoritmo mais atual e demonstrar bons resultados, não vai ser feita a comparação com o mesmo, devido às instâncias utilizadas evidenciarem diferenças em relação às anteriores e não estarem disponíveis. Logo, impossibilitou a obtenção de resultados e consequentemente a sua comparação. Posto isto, a comparação vai ser feita para o algoritmo proposto por Boland et al. [36] e com o algoritmo apresentado por Cánovas et al. [40].

Para facilitar a identificação dos resultados, o algoritmo proposto por Boland et al. foi designado por "M1"e o apresentado por Cánovas et al. por "M2". O algoritmo M1 apenas é apresentado na tabela 5.1, devido ao facto de o autor apenas apresentar resultados para essas instâncias. Nas tabelas seguintes 5.2, 5.3, 5.4, 5.5 e 5.6, a comparação é feita apenas com o algoritmo M2.

É importante realçar que os resultados e tempos aqui apresentados para ambos os algoritmos, foram recolhidos em máquinas distintas, e consequentemente com um poder de computação diferente. O algoritmo M1 foi executado numa máquina com 500 MHz de processador, nomeadamente Digital Personal workstation. Já o algoritmo M2 foi executado numa máquina Pentium com 1500 MHz de processador, 2 GB de memória RAM e 1.5 GB de memória swap, sendo o sistema operativo o Linux, e escrito na linguagem C++. Os resultados apresentados para o algoritmo RAMP proposto neste trabalho, foram obtidos num computador portátil com as seguintes caraterísticas: processador Intel Core i5-560M (3M Cache, 2660 MHz), com 3.7 GB de memória RAM e 7.4 GB de memória swap, com sistema operativo Linux. O algoritmo foi escrito em linguagem C.

O cabeçalho das tabelas é constituído por:

- Instância — $n.f$, onde n corresponde ao número de nós, e f ao custo fixo (sendo L e T). Na tabela 5.6 são apresentados símbolos extra, 1, 5 e 9, que corresponde respetivamente ao valor que α assume, sendo χ e δ igual a 1.
- Solução — Corresponde à solução ótima/melhor resultado conhecido.
- Desvio — Representa o desvio (em percentagem) do valor encontrado em relação ao ótimo ou melhor conhecido.
- CPU — Corresponde ao tempo de computacional (em segundos).

Para cada uma das tabelas de resultados são apresentadas as médias de desvio e de tempo computacional de cada um dos algoritmos. A média da tabela 5.2 é feita tendo por base os resultados obtidos na tabela 5.1 em conjunto com a tabela 5.2, uma vez que o conjunto de teste é o mesmo. Foi feita essa divisão devido ao facto de no algoritmo M1 apenas se conhecer os resultados para esse subconjunto de testes apresentado na tabela 5.1.

De seguida são apresentados os resultados para cada uma das instâncias enumeradas anteriormente.

Na tabela 5.1 são apresentados os resultados para a instância de teste AP, com o número de nós a variar entre 10 e 50. Como podemos verificar, o método M1 consegue encontrar todos os melhores resultados conhecidos até ao momento, bem como o algoritmo RAMP, proposto neste trabalho. O algoritmo M2 não consegue encontra todos os melhores resultados conhecidos, mas consegue encontrar bons resultados em tempos bastante reduzidos quando comparados com o algoritmo M1. Posto isto, também verificamos que o RAMP consegue encontrar todos os melhores resultados conhecidos e com tempos ainda mais reduzidos que o algoritmo M2. Tal como já referido temos de ter em atenção que as máquinas na qual foram executados os algoritmos são bastante distintas, e como tal, nota-se essa disparidade, principalmente a nível dos tempos computacionais.

Instância	Solução	M1		M2		**RAMP**	
		Desvio(%)	CPU(s)	Desvio(%)	CPU(s)	Desvio(%)	CPU(s)
10L	221 033	0	1	0	0	0	0,01
10T	257 558	0	1	0	0	0	0,01
20L	230 385	0	20	1,53	0	0	0,13
20T	266 877	0	52	3,16	0	0	0,14
25L	232 407	0	96	0	0	0	0,20
25T	292 032	0	324	3,07	0	0	0,21
40L	237 115	0	5359	0,62	5	0	0,89
40T	293 165	0	7854	0	8	0	0,84
50L	233 905	0	29 624	3,89	31	0	2,23
50T	296 025	0	48 044	0,72	34	0	1,12
	Média	0	1713,38	1,30	7,80	0	0,58

Tabela 5.1: UMAHLP: Instâncias de teste de problemas AP assimétricos — $\chi = 3$, $\alpha = 0.75$ e $\delta = 2$

Para a tabela 5.2 são apresentados os resultados para o mesmo conjunto de instâncias AP, onde a diferença reside no tamanho dos problemas a resolver, uma vez que são maiores, e o número de nós vai desde os 60 até aos 120.
Para este conjunto de testes podemos notar uma diferença mais acentuada ao nível dos tempos de computação, sendo que o algoritmo M2 tem como média 757.250 segundos, enquanto o RAMP se fica pelos 6.825 segundos, conseguindo obter melhores resultados do que o algoritmo M2.

Abaixo são apresentados os resultados para as instâncias de teste AP com os fluxos simétricos, presentes nas tabelas 5.3, 5.4 e 5.5. Para esses conjuntos de testes pode-

Instância	Solução	M2		**RAMP**	
		Desvio(%)	CPU(s)	Desvio(%)	CPU(s)
60L	225 042	0	57	0	2,16
60T	243 416	0	71	0	2,05
70L	229 874	3,84	144	0	4,24
70T	249 603	2,13	190	0	3,55
80L	225 167	2,88	291	0	7,21
80T	268 209	0	440	0	6,21
90L	226 857	1,42	597	0	9,06
90T	277 418	1,08	915	0	8,71
100L	235 097	0	883	0	12,51
100T	305 098	0	1534	0	11,86
110L	218 662	0	1568	0	16,64
110T	223 892	1,06	2057	0	19,15
120L	222 239	0,31	4706	0	23,34
120T	229 582	1,18	4643	0	31,34
	Média	1,12	757,25	0	6,83

Tabela 5.2: UMAHLP: Instâncias de teste de problemas AP assimétricos grandes — $\chi = 3$, $\alpha = 0.75$ e $\delta = 2$

mos verificar que o algoritmo M2, consegue obter melhores resultados em comparação com os anteriores presentes nas tabelas 5.1 e 5.2. Podemos também verificar que para os resultados apresentados nas tabelas 5.3, 5.4 e 5.5, o algoritmo M2 consegue resultados idênticos para essas instâncias, apesar de que os tempos computacionais vão reduzindo quando o valor de α cresce. Essa curiosidade já não se verifica no algoritmo proposto, sendo os tempos computacionais para essas instâncias muito idênticos e não se reflete que quanto menor for o valor de α mais difícil é encontrar a solução ótima, apesar que os desvios de ambos os algoritmos vão decrescendo quando o α aumenta.

Na tabela 5.6 são apresentados os resultados para o conjunto de teste CAB. Podemos verificar que o algoritmo RAMP consegue encontrar dois resultados melhores dos que atualmente conhecidos, para as instâncias 25T1 e 25T5, sendo os valores encontrados 258556.8 e 279140.88, respetivamente. Nas restantes instâncias são encontrados os melhores resultados conhecidos.

Através dos resultados apresentados pode verificar-se que das 104 instâncias testadas, o algoritmo proposto encontra 100 resultados ótimos, 2 melhores dos que os resultados conhecidos atualmente, sendo que não consegue obter a solução ótima para 2 ficheiros de teste. Posto isto, podemos concluir que o algoritmo consegue encontrar 98% dos resultados ótimos ou melhores conhecidos.

Instância	Solução	M2		**RAMP**	
		Desvio(%)	CPU(s)	Desvio(%)	CPU(s)
10L	122 039	0	0	0	0,01
10T	127 426	0	0	0	0,01
20L	125 310	3,32	0	0	0,12
20T	129 080	1,53	0	1,53	0,11
25L	126 822	0,06	1	0	0,21
25T	143 422	0	1	0	0,21
40L	124 994	0,16	18	0	0,96
40T	140 963	0	15	0	0,78
50L	120 872	0	59	0	1,97
50T	152 294	4,87	81	0	1,56
60L	112 992	0	147	0	1,99
60T	124 961	0	184	0	2,00
70L	114 596	0	402	0	3,42
70T	134 324	0	625	0	3,31
80L	116 506	0	991	0	5,79
80T	138 971	0	1460	0	5,42
90L	115 226	0	1989	0	8,47
90T	130 559	0	2030	0	8,22
100L	123 823	1,44	4750	0	12,68
100T	143 120	0	4400	0	12,05
110L	110 193	0	5794	0	16,52
110T	114 895	0	5976	0	17,04
120L	111 758	0	16 122	0	23,37
120T	118 377	0	20 578	0	23,76
	Média	0,47	1314,68	0,06	6,25

Tabela 5.3: UMAHLP: Instâncias de teste de problemas AP simétricos — $\chi = 1$, $\alpha = 0.1$ e $\delta = 1$

Instância	Solução	M2		**RAMP**	
		Desvio(%)	CPU(s)	Desvio(%)	CPU(s)
10L	125 592	0,93	0	0	0,01
10T	127 426	0	0	0	0,01
20L	126 058	0	0	0	0,11
20T	129 080	0	0	0	0,12
25L	126 901	0	0	0	0,18
25T	143 422	0	0	0	0,20
40L	125 200	0	5	0	0,78
40T	140 963	0	5	0	0,73
50L	124 917	4,71	20	0	1,31
50T	152 294	0	21	0	1,37
60L	116 799	0	39	0	2,04
60T	124 961	0	44	0	2,07
70L	120 503	0	113	0	3,38
70T	135 017	0	112	0	3,27
80L	119 406	0	227	0	5,46
80T	138 971	0	295	0	5,33
90L	118 612	4,55	475	0	8,18
90T	130 559	0	448	0	8,12
100L	125 484	0,10	836	0	16,40
100T	143 120	0	854	0	12,05
110L	116 255	0	1282	0	16,34
110T	121 485	0	1466	0	18,53
120L	118 048	0	3167	0	23,39
120T	122 850	0	2971	0	24,02
	Média	0,43	515,83	0	6,39

Tabela 5.4: UMAHLP: Instâncias de teste de problemas AP simétricos — $\chi = 1$, $\alpha = 0.5$ e $\delta = 1$

Instância	Solução	M2		**RAMP**	
		Desvio(%)	CPU(s)	Desvio(%)	CPU(s)
10L	125 592	0	0	0	0,01
10T	127 426	0	0	0	0,01
20L	126 058	0	0	0	0,10
20T	129 080	0	0	0	0,10
25L	126 901	0	0	0	0,17
25T	143 422	0	0	0	0,17
40L	125 200	0	3	0	0,68
40T	140 963	0	4	0	0,82
50L	124 917	4,71	12	0	1,32
50T	152 294	0	14	0	1,37
60L	116 799	0	20	0	1,99
60T	124 961	0	24	0	1,90
70L	121 859	0	47	0	3,22
70T	135 017	0	59	0	3,19
80L	119 406	0	105	0	5,69
80T	138 971	0	140	0	5,39
90L	118 612	4,55	205	0	8,01
90T	130 559	0	205	0	8,18
100L	125 484	0,39	335	0,10	11,34
100T	143 120	0	395	0	12,15
110L	119 008	0	530	0	16,16
110T	122 257	0	470	0	16,44
120L	119 561	0	747	0	23,31
120T	122 850	0	807	0	23,13
	Média	0,40	171,75	≈ 0	6,04

Tabela 5.5: UMAHLP: Instâncias de teste de problemas AP simétricos — $\chi = 1$, $\alpha = 0.9$ e $\delta = 1$

Instância	Solução	M2		**RAMP**	
		Desvio(%)	CPU(s)	Desvio(%)	CPU(s)
25La	390 369	2,020	0	0	0,396
25L1	196 903	0	0	0	0,310
25L5	234 523	2,600	0	0	0,284
25L9	256 691	0,380	0	0	0,268
25Ta	484 591	1,950	0	0	0,318
25T1	258 577	0,420	1	-0,008	0,277
25T5	279 144	0	0	-0,001	0,265
25T9	284 852	0	0	0	0,273
	Média	0,921	0,125	-0,001	0,299

Tabela 5.6: UMAHLP: Instâncias de teste de problemas CAB

Capítulo 6

Conclusões e Trabalho Futuro

Os Problemas de Localização de Instalações (FLP) e os Problemas de Localização de Hubs (HLP), têm sido extremamente estudados pela comunidade científica devido à sua complexidade e às inúmeras aplicações no mundo real, tais como, redes de computadores, localização de hospitais, localização de escolas, postos de correio, rede elétrica, rede de distribuição de água, entre muitas outras. Ambos os problemas são considerados na literatura como NP-difíceis, o que tem motivado muitos autores a apresentarem algoritmos de estado de arte para a resolução destes problemas.

Inicialmente os método exatos eram suficientes para a resolução deste problemas. Com o passar dos anos, e especialmente com o crescimento da dimensão dos problemas, os métodos exatos começaram a comprometer, devido à necessidade de muitos recursos e de elevados tempos computacionais. Neste contexto, os algoritmos heurísticos começaram a afirmar-se como alternativa. Desde então, vários métodos heurísticos têm sido propostos para a resolução destes problemas, nomeadamente metaheurísticas, tais como, Pesquisa Tabu, Pesquisa por Dispersão, Algoritmos Genéticos, entre outras.
No início quando começaram a aparecer as metaheurísticas, os algoritmos propostos tinham por base apenas uma metaheurística. Com o decorrer dos anos, e com a afirmação das metaheurísticas como uma das melhores formas de resolução dos problemas de otimização, vários algoritmos têm sido propostos combinando vários elementos de várias metaheurísticas, sendo estas conhecidas na literatura como métodos híbridos.

Em 2005, Rego [1] propõem uma nova metaheurística, designada de RAMP, com o objetivo de explorar a relação primal-dual do problema. A metaheurística é bastante flexível, ajustando-se às necessidades do problema, isto é, a *framework* tem presente vários níveis de sofisticação, que permite ajustar em função dos objetivos ou dos resultados que se pretendem atingir. Inicialmente deve-se começar por versões mais simples do método, e avançar para formas mais complexas, apenas se existir necessidade. A versão mais simples

do RAMP, tem como objetivo explorar intensivamente o lado dual do problema, sendo o lado primal explorado duma forma simples, sendo designada por Dual-RAMP, ou apenas RAMP. Já numa versão mais sofisticada da mesma, ambos os dois lados, primal e dual, são explorados intensivamente, sendo a versão conhecida por PD-RAMP. Ambas as versões referidas anteriormente já foram aplicadas com sucesso a vários problemas de otimização.

Nesta dissertação foram aplicadas ambas as versões, a versão mais simples ao UMAHLP e a versão mais sofisticada ao UFLP.
No UFLP, foram propostos dois métodos, o primeiro tendo por base a metaheurística Pesquisa por Dispersão (SS), utilizando como método de melhoramento a abordagem Pesquisa Tabu, e a versão mais sofisticada do RAMP (PD-RAMP).
O PD-RAMP é constituído pelos métodos, Pesquisa por Dispersão, sendo uma versão mais simples da descrita anteriormente, que explora o lado primal do problema, o lado dual é explorado pelo método *dual-ascent* proposto por Erlenkotter [18], em 1978, ao UFLP. Para o UMAHLP, a abordagem RAMP proposta é constituída pelo método de Pesquisa Tabu que explora o lado primal do problema e pelo método *dual-ascent* que explora intensivamente o lado dual.

Ambos os algoritmos propostos nesta dissertação mostram ser competitivos com os melhores existentes na literatura.
Para o UFLP, ambas as abordagens propostas conseguem obter excelentes resultados. O método SS proposto, apesar de não conseguir encontrar tantas soluções ótimas como o PD-RAMP, consegue encontrar excelentes resultados em tempos muito reduzidos. O PD-RAMP consegue encontrar todas as melhores soluções conhecidas, à exceção do grupo de instâncias GHOSH, para as quais atingido um desvio muito próximo de zero, revelando-se muito eficaz na resolução do problema.
O algoritmo RAMP proposto para o UMAHLP superou todas as expetativas, conseguindo obter resultados excelentes em tempos reduzidos. Para os 104 problemas de teste, o algoritmo consegue encontrar 100 resultados ótimos e duas soluções melhores do que as conhecidas atualmente para a grupo de testes CAB.

Apesar dos excelentes resultados em ambos os algoritmos, muito trabalho futuro pode ser elaborado, e acrescentado de forma a tornar ainda mais eficazes e eficientes os algoritmos. De seguida, são descritas algumas melhorias que se podem aplicar, as partes onde é mais provável existir sucesso na aplicação das melhorarias e outros tipos de problemas semelhantes ao UFLP, onde se possa aplicar com sucesso as abordagens propostas. Posteriormente são descritas as consideradas para o UMAHLP.

No algoritmo SS, como apresentado anteriormente, ele é constituído por 5 métodos. Em ambos os métodos podem ser efetuadas melhorias, por vezes mínimas, mas que podem aumentar a qualidade dos resultados. Um dos métodos que pode influenciar mais a qualidade dos resultados é a combinação de soluções, pois podem ser implementadas técnicas mais sofisticadas, que permitam extrair o que de melhor tem cada uma das soluções.
Outro dos métodos com fortes possibilidades é na geração das soluções diversificadas. Uma vez que as soluções são geradas por um algoritmo pseudo-aleatório, a possibilidade de utilizar informação acerca dos armazéns a escolher, é uma possibilidade que pode ter resultados positivos.
Fazendo o algoritmo uso de 5 métodos, cada um com a sua própria função no algoritmo global, quanto mais robusto estiver cada um dos métodos mais eficaz e eficiente se torna o algoritmo. Logo, ainda que, as melhorias sejam mínimas em cada um, ou em alguns dos métodos, no global podem notar-se melhorias na qualidade dos resultados.

Sendo o PD-RAMP constituído pelo SS, ainda que uma versão mais simples do algoritmo aqui proposto ao UFLP, podemos desde logo verificar que as alterações propostas anteriormente ao SS, podem muito bem ser também implementadas no PD-RAMP.
Uma vez que, o método que explora o lado primal no PD-RAMP é uma versão mais simples da abordagem SS, aqui proposta, seria interessante substituir essa versão mais simplificada pela abordagem proposta nesta dissertação. Neste caso, os tempos computacionais podem aumentar consideravelmente e deve-se ter em consideração esse facto importante, podendo por vezes trazer mais fraquezas que benefícios ao PD-RAMP. Outra melhoria que pode trazer algo positivo ao algoritmo é a forma como alternar entre o primal e o dual, isto é, quando é feita a passagem do lado primal para o lado dual. Neste caso, podem ser consideradas informações mais específicas que contribuam para a obtenção mais rápida de boas soluções.

Descritas as possíveis melhorias que se podem considerar a aplicar aos algoritmos do UFLP, e verificando-se que os algoritmos já demonstram excelentes resultados, todas as alterações que os algoritmos sofram só vêm melhorar ainda mais os resultados. Posto isto, e uma vez que analisando a literatura encontramos outros problemas com formulações idênticas à do UFLP, considera-se aplicar a abordagem RAMP a esses problemas, nomeadamente o CFLP (problema de localização de instalações com restrições de capacidade) e o SSCFLP (problema de localização de instalações com restrições de capacidade e um único servidor).
Ambos estes problemas apresentam formulações idênticas ao problema estudado, e uma vez que se verificam que as abordagens propostas são competitivas com as existentes na literatura, conjetura-se que a aplicação da abordagem RAMP aos problemas referidos

também atinjam resultados similares.
As diferenças significativas desses problemas para o UFLP são, existirem capacidades nas instalações, sendo o SSCFLP, uma versão mas simplificada do CFLP. No SSCFLP cada cliente apenas pode ser servido por uma única instalação, enquanto que o CFLP pode ser servido por várias, até que o cliente esteja completamente servido com a quantidade de procura que deseja.

Para o segundo problema tratado, o UMAHLP, também existem outros problemas com formulações idênticas, nomeadamente o problema localização de hubs com restrições de capacidade e alocação múltipla (*Capacitated Multiple Allocation Hub Location Problem* – CMAHLP) e o problema localização de hubs sem restrições de capacidade e alocação múltipla com o número de hubs definido à priori (*Uncapacitated Multiple Allocation p-Hub Median Problem* – UMApHMP).

Podemos considerar o UMApHMP como um problema mais simples em relação ao UMAHLP, uma vez que no problema tratado é necessário descobrir o número de hubs a definir, e posteriormente fazer a alocação dos restantes nós aos hubs. Neste problema essa tarefa já é simplificada, uma vez que já é conhecido o número de hubs a definir à priori, sendo apenas necessário descobrir quais os hubs que devem ser definidos.
Para o CMAHLP, pode-se definir como sendo a versão mais complexa do UMAHLP. A única diferença entre estes dois problemas resume-se ao facto de o CMAHLP existir capacidades nos hubs. De realçar, que esta alteração faz toda a diferença, e acrescenta maior dificuldade na resolução do problema. A escolha dos hubs a serem definidos tem que ser conjugada com a alocação dos nós aos hubs, de forma a existir capacidade suficiente e a minimizar o valor da função objetivo.

Para o problema UMAHLP, ainda podíamos considerar a hipótese de futuramente também aplicar esta abordagem aos problemas descritos anteriormente, mas com a particularidade de ser considerada alocação singular, em vez de múltipla. Nestes problemas apesar de as formulações serem parecidas em parte, existem diferenças mais significativas do que as referidas anteriormente. Neste caso, a abordagem tem que ser bastante modificada, de forma a contemplar outras necessidades da alocação singular, sendo a probabilidade de se obter sucesso bastante mais reduzida.

Concluindo, todo este trabalho foi bem sucedido e com certeza vai ser útil no futuro, atendendo à área que está inserido. O trabalho pode ainda ser aplicado em várias áreas no mundo real, podendo ainda servir de base para futuros estudos na área. Todo este trabalho serviu também para consolidar mais conhecimentos, e futuramente trabalhar em outros tipos de problemas pertencentes ao FLP/HLP.

Bibliografia

[1] C. Rego, "Ramp: A new metaheuristic framework for combinatorial optimization," in *Metaheuristic Optimization via Memory and Evolution* (R. Sharda, S. Voß, R. Sharda, S. Voß, C. Rego, and B. Alidaee, eds.), vol. 30 of *Operations Research/Computer Science Interfaces Series*, pp. 441–460, Springer US, 2005. 10.1007/0-387-23667-8_20.

[2] L. Sagbansua, *Algorithms for very large scale multi-resource generalized assignment problems*. PhD thesis, University, MS, USA, 2004. AAI3167448.

[3] D. Gamboa, *Algoritmos de Memória Adaptativa para a Resolução de Problemas de Optimização Combinatória de Grande Dimensão*. PhD thesis, Instituto Superior Técnico, 2008.

[4] C. Riley, C. Rego, and H. Li, "A simple dual-ramp algorithm for resource constraint project scheduling," in *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, (New York, NY, USA), pp. 67:1–67:5, ACM, 2010.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[6] J. Lenstra, A. Kan, and E. university Rotterdam Econometric institute, *Complexity of packing, covering and partitioning problems*. Econometric Institute, 1979.

[7] M. E. O'kelly, "A quadratic integer program for the location of interacting hub facilities," *European Journal of Operational Research*, vol. 32, pp. 393–404, December 1987.

[8] M. E. O'Kelly, D. Bryan, D. Skorin-Kapov, and J. Skorin-Kapov, "Hub network design with single and multiple allocation: A computational study," *Location Science*, vol. 4, pp. 125–138, Oct. 1996.

[9] F. Glover, "Tabu Search — Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[10] F. Glover, "Tabu Search — Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[11] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decision Sciences*, vol. 8, no. 1, pp. 223–253, 1977.

[12] É. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin, "Adaptive memory programming: A unified view of metaheuristics," *European Journal of Operational Research*, vol. 135, no. 1, pp. 1–16, 2001.

[13] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Manage. Sci.*, vol. 50, pp. 1861–1871, December 2004.

[14] F. Glover, M. Laguna, and R. Marti, "Fundamentals of scatter search and path relinking," *Control And Cybernitics*, vol. 29, no. 3, pp. 653–684, 2000.

[15] M. Sun, "Solving the uncapacitated facility location problem using tabu search," *Comput. Oper. Res.*, vol. 33, pp. 2563–2589, September 2006.

[16] A. A. Kuehn and M. J. Hamburger, "A heuristic program for locating warehouses," *Management Science*, vol. 9, no. 4, pp. 643–665, 1963.

[17] M. Daskin, *Network And Discrete Location: Models, Algorithms, And Applications*. Wiley-Interscience, har/dis ed., May 1995.

[18] D. Erlenkotter, "Dual-based procedure for uncapacitated facility location," *Operations Research*, vol. 26, no. 6, pp. 992–1009, 1978.

[19] M. Korkel, "On the exact solution of large-scale simple plant location problems," *European Journal of Operational Research*, vol. 39, pp. 157–173, March 1989.

[20] K. S. Al-Sultan and M. A. Al-Fawzan, "A tabu search approach to the uncapacitated facility location problem," *Annals of Operations Research*, vol. 86, pp. 91–103, 1999.

[21] D. Ghosh, "Neighborhood search heuristics for the uncapacitated facility location problem," *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, vol. 150, pp. 150–162, OCT 1 2003.

[22] L. Michel and P. V. Hentenryck, "A simple tabu search for warehouse location," *European Journal of Operational Research*, vol. 157, no. 3, pp. 576–591, 2004.

[23] J. Kratica, D. Tosic, V. Filipovic, and I. Ljubic, "Solving the simple plant location problem by genetic algorithm," *RAIRO Operations Research*, vol. 35, pp. 127–142, 2001.

[24] M. G. C. Resende and R. F. Werneck, "A hybrid multistart heuristic for the uncapacitated facility location problem," *European Journal of Operational Research*, vol. 174, pp. 54–68, October 2006.

[25] W. Pullan, "A population based hybrid meta-heuristic for the uncapacitated facility location problem," in *World Summit On Genetic And Evolutionary Computation (GEC 09)*, (1515 Broadway, New York, NY 10036-9998 USA), pp. 475–482, ACM SIGEVO, ASSOC COMPUTING MACHINERY, 2009.

[26] C. Blum and A. Roli, "Hybrid metaheuristics: An introduction," in *Hybrid Metaheuristics* (C. Blum, M. Aguilera, A. Roli, and M. Sampels, eds.), vol. 114 of *Studies in Computational Intelligence*, pp. 1–30, Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-78295-7_1.

[27] M. Laguna and R. Martí, *Scatter Search: Methodology and Implementations in C*. Kluwer Academic Publishers, 2003.

[28] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[29] Y. Kochetov, "Benchmarks library at. http://www.math.nsc.ru/lbrt/k5/kochetov/bench.html."

[30] D. Ghosh, "Benchmarks library at. http://www.iimahd.ernet.in/ diptesh/."

[31] A. T. Ernst and M. Krishnamoorthy, "Efficient algorithms for the uncapacitated single allocation p-hub median problem," *Location Science*, vol. 4, no. 3, pp. 139–154, 1996.

[32] A. T. Ernst and M. Krishnamoorthy, "Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem," *European Journal of Operational Research*, vol. 104, pp. 100–112, January 1998.

[33] J. G. and Klincewicz, "A dual algorithm for the uncapacitated hub location problem," *Location Science*, vol. 4, no. 3, pp. 173–184, 1996.

[34] J. F. and Campbell, "Integer programming formulations of discrete hub location problems," *European Journal of Operational Research*, vol. 72, no. 2, pp. 387–405, 1994.

[35] G. Mayer and B. Wagner, "Hublocator: an exact solution method for the multiple allocation hub location problem.," *Computers & OR*, pp. 715–739, 2002.

[36] N. Boland, M. Krishnamoorthy, A. T. Ernst, and J. Ebery, "Preprocessing and cutting for multiple allocation hub location problems," *European Journal of Operational Research*, vol. 155, no. 3, pp. 638–653, 2004.

[37] H. W. Hamacher, M. Labbé, S. Nickel, and T. Sonneborn, "Adapting polyhedral properties from facility to hub location problems," *Discrete Applied Mathematics*, vol. 145, pp. 104–116, 2004.

[38] A. Marín, "Uncapacitated euclidean hub location: Strengthened formulation, new facets and a relax-and-cut algorithm," *Journal of Global Optimization*, vol. 33, pp. 393–422, 2005. 10.1007/s10898-004-6099-4.

[39] A. Marín, L. Cánovas, and M. Landete, "New formulations for the uncapacitated multiple allocation hub location problem," *European Journal of Operational Research*, vol. 172, no. 1, pp. 274–292, 2006.

[40] L. Cánovas, S. García, and A. Marín, "Solving the uncapacitated multiple allocation hub location problem by means of a dual-ascent technique," *European Journal of Operational Research*, vol. 179, no. 3, pp. 990–1007, 2007.

[41] R. S. de Camargo, J. G. Miranda, and H. P. Luna, "Benders decomposition for the uncapacitated multiple allocation hub location problem," *Comput. Oper. Res.*, vol. 35, pp. 1047–1064, April 2008.

[42] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Computational Management Science*, vol. 2, pp. 3–19, 2005. 10.1007/s10287-004-0020-y.

[43] A. Ernst and M. Krishnamoorthy, "Solution algorithms for the capacitated single allocation hub location problem," *Annals of Operations Research*, vol. 86, pp. 141–159, 1999. 10.1023/A:1018994432663.