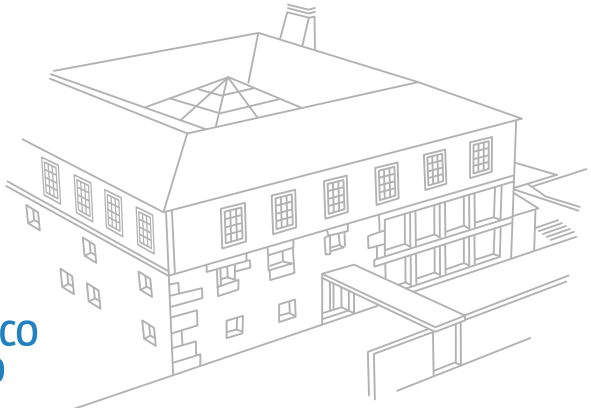

ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

# RAMP para o Problema de Localização de Instalações com Restrições de Capacidade

DESIGNAÇÃO DO MESTRADO
Mestrado em Engenharia Informática

AUTOR
Telmo Manuel Sampaio Pinto de Matos

ORIENTADOR(ES) Dorabela Regina Chiote Ferreira Gamboa

ANO
2011

www.estgf.ipp.pt

# Agradecimentos

Gostaria de agradecer a todos os que me ajudaram na realização deste Mestrado, em particular:

. À Professora Doutora Dorabela Gamboa pelo apoio, orientação e permanente disponibilidade demonstrada durante a elaboração da tese de Mestrado.

. Ao Centro de Inovação e Investigação em Ciências Empresariais e Sistemas de Informação (CIICESI) pelas instalações e disponibilização de recursos para a realização deste trabalho.

. Aos meus colegas de Mestrado, da ESTGF, em particular do CIICESI que, através do companheirismo, tornaram mais fácil a elaboração e conclusão da tese de Mestrado.

. A toda a minha família pela paciência, apoio e motivação para todo o trabalho desenvolvido.

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

Este trabalho sem ti não teria sido possível.

Obrigado

# Resumo

Os problemas de Localização de Instalações fazem parte do conjunto de problemas complexos de otimização combinatória em que o objetivo é a determinação de um conjunto de localizações onde colocar instalações, de forma a satisfazer a procura de um determinado número de clientes com custo mínimo. Tratando-se de problemas NP-difíceis, a utilização de métodos exatos na resolução de problemas de grande dimensão pode ser seriamente comprometida pelos tempos computacionais elevados para a obtenção da solução ótima. Para ultrapassar esta dificuldade, um número significativo de algoritmos heurísticos de vários tipos têm sido propostos com o objetivo de encontrarem soluções de boa qualidade em tempos tão reduzidos quanto possível.

Neste trabalho é explorada a aplicação da abordagem RAMP (Relaxation Adaptive Memory Programming) a dois problemas de localização de instalações: o problema de Localização de Instalações sem Restrições de Capacidade (Uncapacitated Facility Location Problem – UFLP) e o problema de Localização de Instalações com Restrições de Capacidade (Capacitated Facility Location Problem – CFLP). O sucesso obtido com a versão mais simples da abordagem RAMP ao UFLP, tornou interessante a exploração de uma nova abordagem RAMP, com um nível de sofisticação mais elevado, que produziu resultados ainda mais competitivos, dos que os conseguidos com a versão inicial.

Como a aplicação da abordagem RAMP ao UFLP produziu muito bons resultados, foi proposta uma nova aplicação, neste caso ao CFLP. O algoritmo RAMP desenvolvido obteve resultados muito competitivos com os melhores da literatura, evidenciando, novamente, o potencial desta abordagem para outras extensões e variantes dos problemas de localização de instalações.

**Palavras-chave:** Localização de Instalações, Primal-Dual, RAMP, CFLP, UFLP

# Abstract

Facility Location Problems (FLP) are complex combinatorial optimization problems which general goal is to locate a set of facilities that serve a particular set of customers with minimum cost. Being NP-Hard problems, using exact methods to solve large instances of these problems can be seriously compromised by 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 good quality solutions in reasonable computational times.

We propose RAMP (Relaxation Adaptive Memory Programming) approaches for two FLP problems: the Uncapacitated Facility Location Problem (UFLP) and the Capacitated Facility Location Problem (CFLP). The success obtained with the simplest level of the RAMP framework applied to the UFLP, suggested that an application of a more sophisticated level of RAMP to the same problem should improve the results. This expectation was confirmed by the excellent results obtained by this new RAMP application to the UFLP. Guided by the success of the RAMP applications to the UFLP we proposed a new approach for the CFLP. The RAMP algorithm for the CFLP produced very good results in comparison with the best algorithms in the literature, demonstrating the potential of RAMP for new applications to other extensions and variations of Facility Location Problems.



**Keywords:** Facility Location, Primal-Dual, RAMP, CFLP, UFLP

# Índice

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

Os problemas de localização de instalações são problemas muito estudados na literatura e com várias aplicações práticas que vão desde as telecomunicações aos transportes, até às redes de distribuição de água, sendo considerados problemas de difícil resolução.
A obtenção de uma solução ótima em problemas de localização de instalações, nem sempre é compensatória quando esta solução envolve grandes tempos computacionais. Assim, cada vez mais é necessário a conceção de algoritmos metaheurísticos capazes de resolver eficazmente problemas de difícil resolução em tempos reduzidos, contrariando assim a resolução destes problemas pelos métodos exatos.

Optou-se por abraçar este tema pela importância que os problemas de localização de instalações têm na comunidade científica e também pelo desafio que estes problemas oferecem, dada a sua reconhecida dificuldade. Foram implementados algoritmos para duas variantes dos problemas de localização de instalações: a versão mais simples sem restrições de capacidade, denominada UFLP, e uma versão com restrições de capacidade, denominada CFLP.

Diferentes abordagens RAMP foram usadas para a resolução destes problemas, produzindo resultados competitivos com os melhores conhecidos na literatura. Para o UFLP, já foi proposto por Gamboa e Rego[2] uma algoritmo RAMP na versão menos sofisticada, obtendo resultados muito competitivos. Como esta abordagem demonstrou ser competitiva com as melhores metaheurísticas da literatura, explorou-se uma aplicação da abordagem RAMP na versão mais sofisticada, obtendo resultados que ultrapassam os obtidos com a versão menos sofisticada. Visto o sucesso obtido com a abordagem RAMP, decidiu-se aplicar esta metaheurística a um problema mais complicado, o CFLP, tendo também sido obtidos resultados competitivos com os melhores da literatura. Espera-se que o trabalho realizado sirva de alavanca para trabalhos futuros no âmbito de extensões e variantes dos problemas de localização de instalações.

O trabalho aqui descrito, está esquematizado em seis capítulos. O capítulo 1 introduz o trabalho proposto, enquadrando o leitor no âmbito da dissertação que está a ser apresentada. O capítulo 2 introduz conceitos chave e algumas definições para o seguimento deste trabalho. No capítulo 3, são introduzidos conceitos específicos do tema da tese, nomeadamente qual a abordagem metaheurística utilizada e a sua formulação. Nos capítulos 4 e 5 são apresentados os problemas a tratar, os métodos utilizados e os resultados obtidos. Finalmente, o capítulo 6 apresenta as conclusões do trabalho efetuado e o trabalho futuro.

# Capítulo 2

# Enquadramento: Definições e Conceitos Gerais

Neste capítulo, serão abordados termos e notações importantes para a fácil compreensão do trabalho descrito nos capítulos seguintes.

**Otimização Combinatória**: ramo da investigação operacional extensivamente estudado, em que o objetivo é obtenção da melhor solução possível para um dado problema. Em termos matemáticos, um problema de otimização combinatória possui uma função objetivo e um conjunto de restrições associadas a variáveis de decisão. Os valores dessas variáveis de decisão são impostos pelas restrições sobre essas variáveis e o problema pode ser visto como um problema de minimização ou de maximização. Os **Problemas de Localização de Instalações** fazem parte do conjunto de problemas de otimização combinatória de difícil resolução (NP-difíceis[3]), que requerem grande tempo computacional para a obtenção de uma solução ótima ou quase ótima.

A figura 2.1 ilustra o problema de localização de instalações clássico em que $S_m, m = 1, ..., M$ são os clientes, $D_n, n = 1, ..., N$ as instalações e $c_{ij}$ os custos de alocação. O objetivo é a minimização dos custos de servir completamente todos os clientes sabendo o custo de abertura de cada instalação. Neste problema as instalações não possuem restrições nas capacidades, denominando-se **Problema de Localização de Instalações sem Restrições de Capacidade**. Quando o problema admite restrições nas capacidades das instalações denomina-se **Problema de Localização de Instalações com Restrições de Capacidade**.

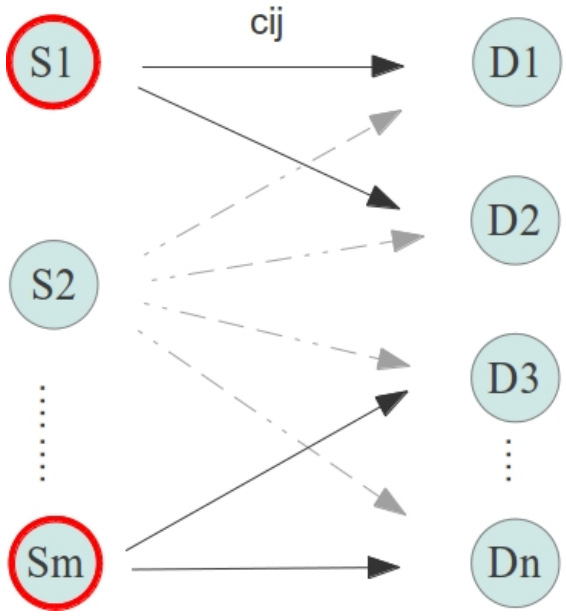


Figura 2.1: Problema de localização de instalações clássico

**Espaço de Soluções Admissíveis**: é o conjunto composto por todas as soluções válidas (que satisfazem todas as restrições do problema) para um dado problema.

**Estrutura de Vizinhança**: seja $S$ o espaço de soluções admissíveis para um dado problema $P$. Sendo $s$ uma solução de $P$, chama-se vizinhança de $s$ $V(s)$, ao conjunto de soluções $V(s) \in S$ que se podem alcançar a partir de $s$, através de uma operação denominada movimento. A cada elemento de $V(s)$ chama-se vizinho de $s$.

**Ótimo Local**: é a melhor solução que um algoritmo de melhoramento consegue obter com uma determinada estrutura de vizinhança.

**Ótimo Global**: é a solução ótima para um dado problema.

**Problema Dual**: para cada problema Primal (problema original) existe um problema Dual. Se o problema primal é de maximização o dual é de minimização. A cada restrição do primal em que se verifique uma desigualdade (maior ou menor), no dual é o inverso (menor ou maior). Cada variável no primal equivale a uma restrição no dual e vice-versa. O dual do dual é o primal. Os problemas são resolvidos em espaços de soluções distintas. De seguida é apresentado um exemplo da obtenção do dual para um problema de programação linear básico.

O problema primal é definido da seguinte forma:

$$MaxL = \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

s.a

$$\sum a_{ij} x_j, \leq b_i \quad , i = 1, ..., m \tag{2.2}$$

$$x_j \geq 0 \quad , \forall j \tag{2.3}$$

O respetivo dual é obtido da seguinte forma:

$$MinL = \sum_{i=1}^{m} b_i y_i \tag{2.4}$$

s.a

$$\sum a_{ij} y_i, \geq c_{ij} \quad , j = 1, ..., n \tag{2.5}$$

$$y_i \geq 0 \quad , \forall i \tag{2.6}$$

**Técnicas de Relaxação**: diferentes técnicas de relaxação têm sido largamente usadas no ramo da otimização combinatória dando limites inferiores e superiores para o problema a tratar. Muitas heurísticas têm sido implementadas com o recurso a técnicas de relaxação produzindo resultados computacionais competitivos. A relaxação tem como base a exploração do espaço de soluções do problema relaxado que é obtido através do problema original, atuando sobre as restrições de um problema tornando-as mais simples de resolver. Exemplos de técnicas de relaxação são a relaxação por restrições substitutas, a relaxação lagrangeana e o otimização por subgradiente. Podemos também obter com a combinação da relaxação por restrições substitutas e da relaxação lagrangeana a relaxação paramétrica cruzada[1]. O uso do tipo de relaxação depende do problema a analisar e da sua dificuldade.

**Algoritmos Exatos**: os algoritmos usados para a resolução de problemas de otimização combinatória podem ser divididos em duas classes: os algoritmos exatos e os algoritmos heurísticos. Os algoritmos exatos, como o próprio nome indica, garantem a obtenção da solução ótima, sendo que o tempo computacional necessário para a obtenção dessa solução ótima é proporcional à dimensão do problema. Assim, os algoritmos exatos apresentam

grandes limitações para a obtenção em tempo útil de soluções para problemas complexos (de grande dimensão). Alguns exemplos de algoritmos exactos mais conhecidos são o método Simplex[4] e o "*Branch and Bound*"[5].

Para colmatar o problema do tempo computacional elevado na obtenção de uma solução ótima, surgem os algoritmos heurísticos que, apesar de não garantirem a solução ótima, encontram uma boa solução num tempo computacional aceitável.

**Algoritmos Heurísticos**: os algoritmos heurísticos podem ser divididos em três diferentes grupos: construtivos, de melhoramento ou pesquisa local e metaheurísticos.

Os **algoritmos construtivos** obtêm uma solução para o problema de forma incremental, ou seja, em cada iteração do algoritmo é escolhido um componente a inserir na solução que se está a construir até criar uma solução admissível.

Nos **algoritmos de melhoramento** ou de **pesquisa local**, a solução corrente é perturbada para gerar uma nova solução de melhor qualidade com recurso a uma estrutura de vizinhança. Tomando como exemplo os problemas de localização de instalações, podemos partir de uma solução admissível e depois ir somando ou subtraindo um armazém, ou mudando a sua localização e assim avançando consecutivamente para a melhor solução vizinha. O algoritmo prossegue até não ser possível melhorar a solução com a estrutura de vizinhança escolhida. Nesta situação o algoritmo termina retornando a melhor solução encontrada (ótimo local).

Na figura 2.2 é apresentada a estrutura genérica do algoritmo de melhoramento para um problema de maximização com função objetivo $f$ e onde $V(S_c)$ representa a vizinhança da solução corrente.

***Bloco I: Método de Melhoramento ou de Pesquisa Local***

**Passo 1.** Obter uma solução admissível, $S_0$.

**Passo 2.** Inicializar a 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 $S_v = S_c$. Senão $V(S_v) = V(S_v) - S_v$.

**Passo 5.** Se $V(S_v) \neq \oslash$ voltar ao **Passo 3**. Caso contrário terminar.

Figura 2.2: Método de Melhoramento ou de Pesquisa Local

**Algoritmos Metaheurísticos**: para colmatar a facto de um método de melhoramento parar num ótimo local, surgem os algoritmos metaheurísticos, que conseguem escapar da vizinhança desse ótimo local mesmo que piore a solução corrente, com o objetivo de passar para outra parte do espaço de soluções admissíveis e ir ao encontro desse ótimo global.

A figura 2.3 ilustra a ideia base de uma metaheurística na tentativa de escapar do ótimo local para um dado problema de minimização. Tendo obtido uma solução S, ótimo local, consegue através de uma perturbação, pesquisar na vizinhança de S', que é novamente um ótimo local e finalmente pesquisa na vizinhança de S" que é ótimo global.

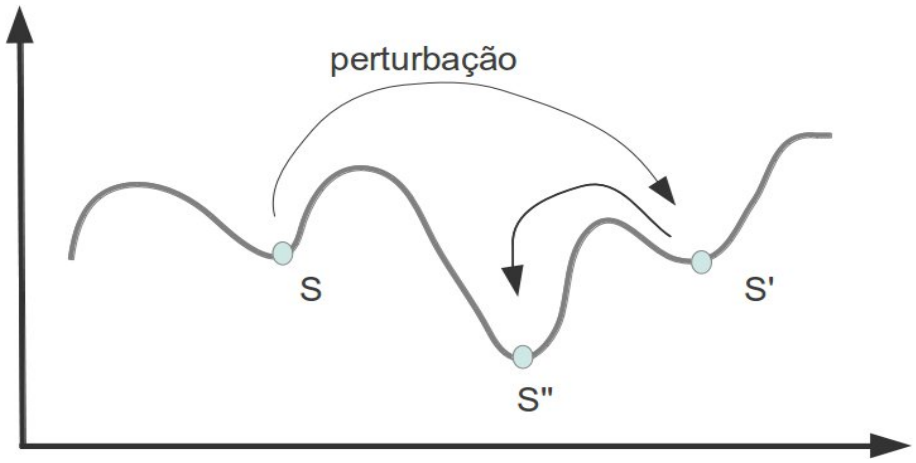


Figura 2.3: Perturbação da solução para tentar escapar ao ótimo local

Para além da pesquisa numa vizinhança do ótimo local, os algoritmos metaheurísticos permitem piorar a solução de forma a conseguir explorar o espaço de soluções ainda não visitadas, esperando encontrar soluções de melhor qualidade, nomeadamente o ótimo global. Os algoritmos construtivos embora sejam capazes de obter soluções muito rapidamente, a qualidade dessas soluções é inferior aos fornecidos pelos algoritmos de melhoramento e em especial aos algoritmos metaheurísticos que, frequentemente, produzem soluções ótimas ou muito próximas da solução ótima, em tempos computacionais aceitáveis.

Um exemplo clássico de um algoritmo metaheurístico é a pesquisa tabu (ou *Tabu Search*[6][7]). Este método, que inclui um procedimento de pesquisa local, tenta fugir aos ótimos locais para tentar encontrar o ótimo global, mesmo que piore a solução corrente, guardando movimentos proibidos para que o movimento não leve a uma solução previamente visitada. Este método pode possuir memória flexível que pode ter vários níveis (memória de curto, médio e longo prazo).

Na sua estrutura mais simples, este método inclui os seguintes componentes:

**Procedimento de Pesquisa Local:** este procedimento parte de uma solução inicial e, através de um movimento definido pela estrutura de vizinhança, explora o espaço de soluções próximas (vizinhas) da solução corrente, passando para a melhor solução vizinha (solução que melhora a solução corrente, ou que menos piora a solução corrente).

**Lista Tabu:** a lista tabu tem como objetivo registar movimento proibidos para que no procedimento de pesquisa local não se corra o risco de voltar sempre a cair no mesmo ótimo local. Desta forma, o algoritmo só pode considerar movimentos que não sejam tabu, a não ser que exista um Critério de Aspiração (exemplo: o movimento tabu melhora a melhor solução encontrada até ao momento).

**Critério de Paragem:** como o algoritmo não pára quando obtém um ótimo local, é necessário definir quando parar o processo de pesquisa. Alguns exemplos de critérios de paragem são o número fixo de iterações, tempo fixo de utilização do CPU, o número fixo de iterações sem melhoramento da solução encontrada, etc.

Para além da estrutura básica acima descrita, a pesquisa tabu possui também duas fases muito importantes: a **Intensificação** e a **Diversificação**. A Intensificação permite a exploração com maior intensidade numa região do espaço de soluções que tenha sido identificada como promissora. Isto envolve utilizar memória de longo prazo para guardar características das melhores soluções encontradas. O termo Diversificação permite a exploração no espaço de soluções que ainda não foram exploradas.

O procedimento genérico da pesquisa tabu pode ser visto na figura 2.4, onde $f$ é a função objetivo, $V(S_c)$ a vizinhança da solução corrente, $T$ a lista com os movimentos tabu e $S(T)$ as soluções da vizinhança $V(S_c)$ que são proibidas (pois os movimentos necessários para as obter pertencem a $T$).

***Bloco II: Procedimento Pesquisa tabu na versão mais simples***

**Passo 1.** Obter uma solução inicial admissível, $S_0$.

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

**Passo 3.** Iniciar o contador $k = 0$.

**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 2.4: Procedimento Pesquisa tabu na versão mais simples

# Capítulo 3

# Relaxation Adaptive Memory Programming

O método RAMP (Relaxation Adaptative Memory Programing)[1] é uma abordagem metaheurística baseada na exploração da relação entre o primal e o dual na resolução de problemas de otimização, sendo que a pesquisa tem por base o princípio de memória adaptativa. O termo primal diz respeito ao espaço de soluções do problema original enquanto que o termo dual refere-se ao problema dual de uma relaxação do problema original.

O método permite diferentes níveis de sofisticação, dependendo da intensificação na exploração dos lados primal e dual, e das técnicas usadas para combinar da melhor forma as soluções obtidas, através de ambas as componentes (primal e dual). Na sua forma mais simples, este método explora mais predominantemente o lado dual, sendo a interação com o primal, obtida projetando as soluções duais para o espaço de soluções primais e aplicando-lhes um método de melhoramento. Uma sofisticação com nível mais elevado permite a exploração mais intensiva do lado primal, incorporando estruturas de memória mais complexas. São apresentados de seguida os diferentes níveis de sofisticação em que Dual RAMP, ou simplesmente RAMP, designa a versão mais simples e PD-RAMP a versão mais elaborada.

Podemos definir em termos gerais o método RAMP como a combinação de conceitos de memória adaptativa (*Adaptive Memory Programming*) [8] e técnicas de pesquisa metaheurísticas com princípios de relaxação matemática, abrangendo o espaço de soluções duais e primais, com o objetivo de obter informação crucial na pesquisa a desenvolver e agrupando-a com informação obtida pelos lados dual e primal. Originalmente este método sugere a combinações dos conceitos mencionadas anteriormente envolvendo diferentes níveis de sofisticação na sua implementação. A utilização da relaxação Lagrangeana, re-

laxação por restrições substitutas entre outras técnicas de relaxação (relaxação paramétrica cruzada proposta no contexto do RAMP[1]) são exemplos de estratégias duais. Como estratégias para a componente primal é sugerido a pesquisa por dispersão (ou *Scatter Search*[9][10]) ou a pesquisa por ligação de caminhos (ou *Path Relinking* [9]) como exemplos de métodos adequados à utilização de memória adaptativa. Em termos gerais o método RAMP pode ser descrito como um processo incremental, começando na sua forma menos sofisticada e passando sucessivamente para formas mais complexas.

A figura 3.1 representa o modelo genérico do método PD-RAMP.

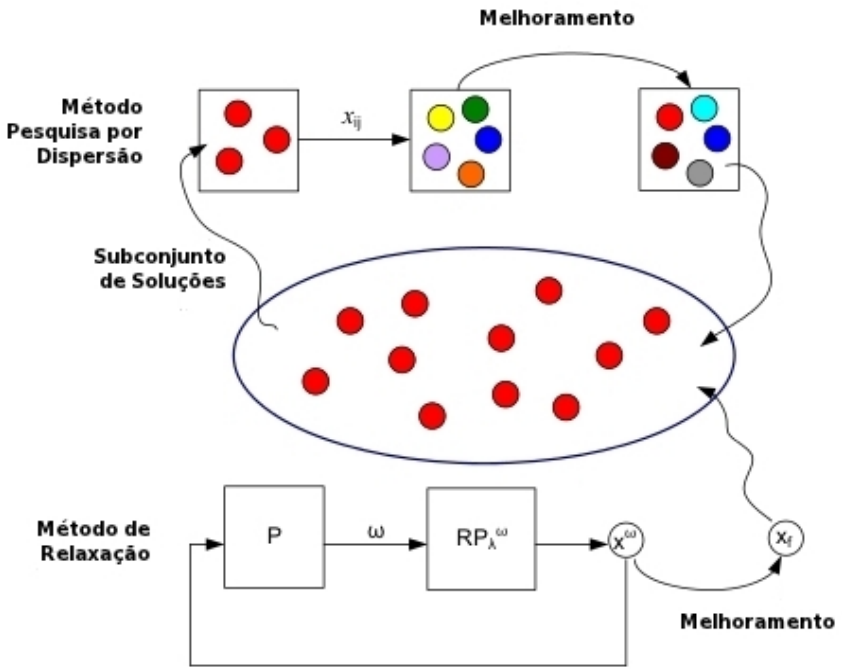


Figura 3.1: Modelo genérico RAMP[1]

Para este modelo é usada uma técnica de relaxação para a obtenção do soluções duais e um método evolutivo para a exploração do lado primal. Escolhendo uma técnica apropriada para o modelo RAMP, o nível de sofisticação a usar depende muito dos resultados obtidos e/ou dos resultados a obter. Usando o nível de sofisticação mais simples e obtendo os resultados desejados, não faz sentido implementar um nível de sofisticação mais elevado, pois o tempo computacional irá certamente aumentar.

Gamboa e Rego[2] aplicaram o método RAMP ao UFLP na sua versão mais simples, produzindo muito bons resultados. Nesta implementação foi usada o *Dual Ascent* como técnica de relaxação para obtenção de soluções duais baseado em Erlenkotter[11] e o método de melhoramento utilizado foi a pesquisa tabu proposta por Michel and Hentenryck[12]. Os resultados computacionais superaram os apresentados na literatura, o que demonstra a sua aplicabilidade neste e noutros tipos de problemas de localização de instalações.

# 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, também conhecido por *Uncapacitated Facility Location Problem*(UFLP) ou *Uncapacitated Plant Location Problem* ou ainda *Uncapacitated Warehouse Location Problem* (de agora em diante será utilizado o acrónimo **UFLP** para identificar este problema), com m instalações (armazéns, por exemplo) e n clientes pode ser definido da seguinte forma. A cada instalação $i$ está associado um custo fixo de abertura $F_i$ e um custo de transporte $c_{ij}$ para servir totalmente o cliente $j$ a partir da instalação $i$. O objetivo é encontrar um conjunto de localizações onde colocar uma instalação de forma a satisfazer a procura de todos os clientes com custo mínimo.

O problema tem a seguinte formulação inteira:

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

s.a

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

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

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

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

Nesta formulação, $C = \{1, ..., n\}$ e $A = \{1, ..., m\}$ representam, respetivamente, os conjuntos de clientes e de instalações. As variáveis $x_{ij}$ indicam se o cliente $j$ é ou não servido pela instalação $I$, e $y_i$ especifica se a localização $I$ contém uma instalação aberta ou fechada. Note-se que, devido ao facto de se admitir que as instalações possuem capacidade ilimitada, uma vez selecionado um conjunto de localizações com instalação aberta, a afetação ótima dos clientes é imediata, consistindo em servir cada cliente através da instalação aberta com menor custo. Deste modo, pode-se definir o problema exclusivamente com variáveis binárias ($x_{ij}$ passa a ser 1 ou 0) e assim uma solução UFLP fica completamente definida pelo conjunto de instalações abertas (ou seja, uma solução pode ser representada por um vetor $Y = (y_1, ..., y_i, ..., y_m)$ onde $y_i = 1$ se a instalação $i$ estiver aberta e 0 caso contrário.

## 4.2 Algoritmos Heurísticos para o UFLP

Muito progresso tem sido feito no campo das metaheurísticas no que diz respeito à resolução do UFLP. São bons exemplos a pesquisa tabu[12][13][14], os algoritmos genéticos[15] e os algoritmos baseados em redes neuronais artificiais[16][17]. As abordagens baseadas em relaxação também têm contribuido com algoritmos eficazes para a resolução deste problema [11][18]. Existem algoritmos que, por agruparem mais do que um tipo de abordagem, são denominados híbridos. São estes híbridos que estão a conseguir resolver eficazmente instâncias de dificuldade reconhecida na localização de instalações, encontrando o seu valor óptimo e, em alguns casos, atingindo esse valor óptimo em tempos computacionais muito reduzidos. De entre os algoritmos para a resolução do UFLP importa referir os que produzem os resultados mais competitivos. São eles:

O algoritmo híbrido dos autores Resende e Werneck (H-RW)[19] que consiste essencialmente em duas fases: uma fase essencialmente de intensificação e uma fase de otimização. Na primeira fase é criada uma solução aleatória por cada iteração e aplicado um algoritmo de pesquisa local. A solução resultante é combinada através de um processo de ligação de caminhos, com outras soluções do conjunto de soluções elite. A solução resultante e a solução que deu origem à nova solução desta combinação é posteriormente inserida, seguindo determinados critérios, no conjunto de soluções elite. Na segunda fase do processo são novamente combinadas soluções do conjunto de soluções elite com outras soluções do processo iterativo na esperança de que com a geração de novas soluções, resulte numa solução melhor que a previamente encontrada. A estrutura de pesquisa local utilizada é a passagem de uma instalação aberta para fechada e vice versa (também denominada de "*Swap*"). Este método diz-se híbrido pois combina elementos de várias metaheurísticas tais como pesquisa por dispersão, pesquisa tabu e algoritmos genéticos.

O algoritmo Dual RAMP (Relaxation Adaptive Memory Programming) apresentado por Rego e Gamboa (RAMP)[2], como já foi visto no capítulo 3, consiste na utilização de uma relaxação para a resolução do problema dual, seguido do método de projeção para transformar as soluções obtidas pelo método dual para o espaço de soluções primais, seguido de um método de melhoramento baseado em pesquisa tabu proposto por Michel e Hentenryck (TS-MH)[12]. O movimento utilizado consiste em passar o estado de uma instalação de aberta para fechada ou vice versa ("*Flip*").

O algoritmo proposto por Sun (TS-S)[13] tem por base a pesquisa tabu com utilização de memória flexível de curta, média e longa duração, para guardar o estado da instalação e também o uso do critério de aspiração. A estrutura de vizinhança da pesquisa local é baseado em movimentos "*ADD*"e "*DROP*". O movimento *ADD* começa com todas as instalações fechadas e abre a instalação que obtém o mínimo custo em abrir. Esta fase termina quando já não existem instalações para abrir que minimizam o custo total. O método *DROP* consiste em alterar o estado de uma instalação do conjunto de instalações abertas para fechada, em que o custo de fechar essa instalação seja mínimo.

O algoritmo proposto pelos autores Michel e Hentenryck (TS-MH)[12] também é baseado em pesquisa tabu em que a estrutura de pesquisa local consiste a estrutura de vizinhança em trocar o estado de uma instalação de aberta para fechada ou vice versa ("*Flip*").

Por fim o algoritmo PBS ("*Population Based Hybrid*") proposto por Wayne Pullan[20] consiste no uso dos algoritmos genéticos para gerar soluções (ou populações) e assim gerar novos conjuntos para entrar no procedimento de pesquisa local. A mais valia deste meta-heurística híbrida é a geração de soluções promissoras para o procedimento de pesquisa local e assim explorar mais regiões do espaço de soluções. Os movimentos usada são o *ADD* e o *DROP* que, iterativamente, alteram o estado das instalações.

## 4.3 Descrição do Algoritmo

Como a abordagem RAMP proposto por Gamboa e Rego[2] obteve resultados muito competitivos para o UFLP, resolveu-se implementar um algoritmo PD-RAMP com o objetivo de conseguir ainda melhores resultados.

O algoritmo básico RAMP aplicado ao UFLP proposto por Gamboa e Rego[2] pode ser descrito da seguinte forma: começa-se por encontrar soluções para o problema dual. As soluções encontradas são projetadas para o espaço de soluções do primal, as quais são posteriormente melhoradas com base num método de melhoramento. Este procedimento é repetido até ser atingido um determinado número de iterações consecutivas sem melhoramento da melhor solução encontrada até ao momento.

São apresentados de seguida as componentes Primal e Dual do método PD-RAMP proposto para a resolução do UFLP.

### 4.3.1 Método Dual

O método Dual do algoritmo proposto utiliza a relaxação e dualização usada por Erlenkotter[11] no algoritmo DUALOC para a resolução do UFLP. Neste estudo utilizou-se o procedimento *Dual Ascent*, obtendo um limite inferior para o problema que, projetando as soluções para o primal, produzem com frequência soluções próximas do ótimo global.

A formulação condensada do problema dual obtém-se da seguinte forma: começa-se por obter uma relaxação linear do problema original ((4.1) até (4.5)), substituindo as restrições de integralidade por restrições de não negatividade, obtendo a seguinte formulação dual do problema:

$$Max \qquad \sum_{j=1}^{n} v_j \tag{4.6}$$

s.a

$$\sum_{j=1}^{n} w_{ij} \leq F_i \qquad \forall i \in A, \tag{4.7}$$

$$v_j - w_{ij} \leq c_{ij} \qquad \forall i \in A, j \in C, \tag{4.8}$$

$$w_{ij} \geq 0 \qquad \forall i \in A, j \in C, \tag{4.9}$$

Mantendo a admissibilidade sem alteração da função objetivo, assume-se que $w_{ij} = max\{0, v_j - c_{ij}\}$. Substituindo $w_{ij}$ no primeiro conjunto de restrições, obtém-se a formulação condensada do dual apenas contemplando as variáveis $v_j$:

$$Max \qquad \sum_{j=1}^{n} v_j \tag{4.10}$$

s.a

$$\sum_{j=1}^{n} max\{0, v_j - c_{ij}\} \leq F_i \qquad \forall i \in A, \tag{4.11}$$

De seguida são apresentados os traços gerais do procedimento *Dual Ascent* que parte de uma solução inicial para o dual e tenta melhorar essa solução. Começando com uma solução admissível $\{v_j\}$ percorrendo repetida e sequencialmente todos os clientes $j$, tentando maximizar o valor de $v_j$ para o valor mais alto de $c_{ij}$. Neste procedimento, se houver uma restrição do dual no método condensado que bloqueie esse aumento, $v_j$ é aumentado até ao seu máximo permitido pela restrição. Quando $v_j$ não pode ser aumentado para nenhum valor de $j$, o procedimento termina.

A figura 4.1 apresenta o procedimento *Dual Ascent* tal como foi proposto por Erlenkotter[11]. Assuma-se $c_j^k (k = 1, ..., m)$ a reindexação de $c_{ij}$ para cada $j$ por ordem não decrescente (inclui-se um custo fictício $c_j^{m+1} = +\infty$). Assim é obtida uma solução inicial admissível para o problema dual fazendo $v_j = c_j^1 \forall j$. No **Passo 3** restringe-se a alteração do valor de $v_j \forall j \in C^+$. Esta restrição é apenas usada no procedimento *Dual Adjustment*, que é um dos procedimentos, juntamente com o *Dual Ascent*, usado por Erlenkotter. Para o

algoritmo proposto somente foi usado o *Dual Ascent*, pois os resultados obtidos foram de grande qualidade, não sendo necessário o uso do procedimento *Dual Adjustment*.

***Bloco III: Procedimento Dual Ascent para o UFLP***

**Passo 1.** Começar com uma solução dual admissível $\{v_j\}$, tal que $v_j \geq c_j^1$, para cada $j \in C$ e $S_i = F_i - \sum_{j=1}^{n} max\{0, v_j - c_{ij}\} \geq 0$, para cada $i \in A$. Para cada $j \in C$, definir $k(j) = min\{k : v_j \leq c_j^k\}$. Se $v_j = c_j^{k(j)}$, incrementar $k(j)$ uma unidade.

**Passo 2.** Inicializar $j = 1$ e $\delta = 0$.

**Passo 3.** Se $j \notin C^+$, passar para o **Passo 7**.

**Passo 4.** Fazer $\Delta_j = min_{i \in A}\{S_i : v_j - c_{ij} \geq 0\}$.

**Passo 5.** Se $\Delta_j > c_j^{k(j)} - vj$, fazer $\Delta_j = c_j^{k(j)} - vj$ e $\delta = 1$, e incrementar $k(j)$ uma unidade.

**Passo 6.** Decrementar $\Delta_j$ a $S_i$, para cada $i \in A$ com $v_j - c_{ij} \geq 0$; depois incrementar $\Delta_j$ a $v_j$ .

**Passo 7.** Se $j \neq n$, incrementar 1 a $j$ e voltar ao **Passo 3**.

**Passo 8.** Se $\delta = 1$, voltar ao **Passo 2**. Caso contrário terminar.

Figura 4.1: Procedimento *Dual Ascent* para o UFLP

### 4.3.2 Método Primal

Conhecendo a solução dualizada, essa solução é transformada numa solução para o problema primal, e posteriormente aplicado um método de melhoramento. A projeção das soluções duais para o espaço de soluções primais é feita da seguinte forma:

Seja o conjunto $A^+$ o conjunto de instalações satisfazendo a igualdade $\sum_{j=1}^{n} max\{0, v_j^+ - c_{ij}\} = F_i \quad \forall i \in A^+$. Considere-se a solução dual admissível $v_j^+$ com o conjunto correspondente de instalações $A^+$ e para cada $j, v_j^+ \geq c_{ij}$ para algum $i \in A^+$. Defina-se para o cliente $j$ uma instalação de custo mínimo $i^+(j) \in A^+$ com custo $c_j^+ = c_{i^+(j)j} = min_{i \in A^+} c_{ij}, j \in C$.

Deste modo, a solução primal pode ser obtida através das expressões:

$$y_i^+ = 1, i \in A^+ \tag{4.12}$$
$$y_i^+ = 0, i \notin A^+ \tag{4.13}$$
$$y_i^+ = 1, i = i^+(j), j \in C \tag{4.14}$$
$$y_i^+ = 0, i \neq i^+(j), j \in C \tag{4.15}$$

A solução obtida primal é ótima se satisfaz as seguintes condições de desvios complementares:

$$y_i^*[F_i - \sum_{j=1}^{n} max\{0, v_j^* - c_{ij}\}] = 0 \tag{4.16}$$
$$[y_i^* - x_{ij}^*]max\{o, v_j^* - c_{ij}\} = 0 \tag{4.17}$$

Após a projeção da solução dual para o espaço de soluções do problema primal, obtém-se o conjunto $A^+$ como referido anteriormente, ou seja, considerando como instalações abertas as instalações pertencentes a $A^+$, que correspondem às instalações para quais as variáveis desvio são nulas, ou seja, $s_i = F_i - \sum_{j=1}^{n} max\{0, v_j^+ - c_{ij}\} = 0$. Desta forma são fechadas as instalações em que se verifique $s_i > 0$ antes do algoritmo aplicar o método de melhoramento.

De seguida é aplicado um procedimento de Pesquisa por Dispersão (Scatter Search - SS). A Pesquisa por Dispersão[9][10] é um método evolutivo que utiliza técnicas de intensificação, diversificação e de combinação de soluções. Este método utiliza componentes, que conjugadas entre si, fazem desta abordagem um método robusto e com muitas aplicações[10], não só na localização de instalações mas também em muitos outros problemas de otimização combinatória[9] [10]. As componentes da Pesquisa por Dispersão para o UFLP são apresentadas na figura 4.2:

O procedimento de Pesquisa por Dispersão utilizado no algoritmo aqui apresentado foi proposto por Matos et al[21]. O algoritmo PD-RAMP alterna entre as componentes dual e primal até que não haja melhoramento da solução corrente. O método pode ser melhorado através da componente **Combinação de Soluções**, pois combinar soluções do conjunto de referência com soluções melhoradas pode vir a revelar soluções de boa qualidade. De seguida são descritas as componentes deste método à exceção do método de melhoramento que irá ser descrito num ponto mais abaixo.

***Bloco IV: Componentes da Pesquisa por Dispersão***

1. Método de Geração de Soluções Diversificadas
   Gera soluções diversificadas a partir de soluções iniciais arbitrárias usadas para iniciar o método.
2. Método de Melhoramento
   Fase de melhoramento das soluções em **1**.
3. Método de Atualização do Conjunto de Referência
   Responsável por atualizar, gerir e guardar as melhores soluções, garantindo a diversificação e ao mesmo tempo as soluções de elite.
4. Método de Geração de Subconjuntos
   Gera subconjuntos de soluções para serem usados em **5**.
5. Método de Combinação de Soluções
   Transforma cada subconjunto obtido em **4**. em soluções combinadas através da utilização de combinações lineares.

Figura 4.2: Componentes da Pesquisa por Dispersão para o UFLP

Após terminar a componente **Gerar Soluções Diversificadas** descrita na figura 4.3 a fase de melhoramento tenta melhorar as soluções obtidas na geração de soluções diversificadas através da aplicação do método de melhoramento. Após terminar o método de melhoramento é atualizado o conjunto de referência descrito na figura 4.4.

Se ainda não foi atingido o nível pretendido de qualidade e diversidade de soluções, volta a criar soluções diversificadas. Terminada esta fase inicial do método, vem a fase da pesquisa por dispersão propriamente dita (figuras 4.5 e 4.6):

Nesta fase (figura 4.5) são obtidos os subconjuntos do conjunto de referência (Refset) onde são escolhidas soluções que irão posteriormente ser combinadas no método de **Combinação de Soluções**, ilustrado na figura 4.6.

Após terminar a fase de gerar a combinação de soluções (figura 4.6), tenta melhorar essas soluções através do método de melhoramento. De seguida atualiza o conjunto de referência e se esta fase não produziu soluções melhores ou se ainda não foi alcançado um número pré-definido de iterações, volta a gerar subconjuntos do conjunto de referência (Refset) através do **Método de Geração de Subconjuntos**.

***Bloco V: Gerar soluções diversificadas***

1. Inicializar vetor de soluções.
2. Inicializar semente para pseudo-aleatório.
3. Gerar armazém.
4. Se armazém ainda não existe na solução, adicionar à solução.
5. Se ainda não foi atingido o número de armazéns abertos por solução, ir para **Passo 3**.
6. Se ainda não atingiu o número total de soluções, ir para **Passo 2**.
7. Terminar.•

Figura 4.3: Componente para gerar soluções diversificadas para o UFLP

***Bloco VI: Criar e Actualizar o Conjunto de Referência***

1. Inicializar o conjunto de referência (Refset).
2. Se a solução ainda não existe no conjunto de referência, adiciona-a.
3. Se ainda não atingiu o número de soluções total, voltar para o **Passo 2**.
4. Terminar.•

Figura 4.4: Componente para Criar e Actualizar o Conjunto de Referência para o UFLP

O método de melhoramento usado no algoritmo proposto foi baseado na Pesquisa Tabu apresentada pelo autor Michel e Hentenryck[12]. Este método utiliza uma estrutura de vizinhança que consiste em trocar o estado de cada instalação (de aberta para fechada ou vice versa) e utiliza uma lista tabu que guarda as instalações que não podem ver o seu estado alterado durante um determinado número de iterações.

O método pode ser descrito da seguinte forma: assumindo as listas de armazéns fechados e abertos, $FechadasInstalacoes^+$ e $AbertasInstalacoes^+$, respetivamente, ordenados por ordem crescente de custo e as função Ganho($S^+$), $S^+ \in FechadasInstalacoes^+$ e Ganho($S^-$), $S^- \in AbertasInstalacoes^+$, que 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 $AbertasInstalacoes^+$ existe algum movimento que melhora a melhor solução encontrada até ao momento. A função Melhor($S^+$) e Melhor($S^-$) retorna o menor custo de cada uma das listas $FechadasInstalacoes^+$ e $AbertasInstalacoes^+$,

***Bloco VII: Geração dos subconjuntos***

1. Inicializar o subconjunto.
2. Verificar as soluções melhores/piores a escolher.
3. Se a solução ainda não existe no subconjunto, adiciona-a.
4. Passar à solução seguinte a adicionar.
5. Se ainda não atingiu o número de soluções, voltar para o **Passo 3**.
6. Terminar.•

Figura 4.5: Componente para Geração dos subconjuntos para o UFLP

***Bloco VIII: Combinar as Soluções***

1. Inicializar conjunto que guardará as soluções combinadas.
2. Obter soluções a combinar.
3. Calcular rácios para cada solução.
3. Combinar soluções e adicionar ao conjunto de soluções combinadas.
5. Se ainda não atingiu o número de soluções total a combinar, voltar para o **Passo 2**.
6. Terminar.•

Figura 4.6: Componente para Combinar as Soluções para o UFLP

respetivamente. Na figura 4.7, é apresentado o modelo utilizado para o método de melhoramento:

## 4.4 Estruturas de Dados

Na figura 4.8 são apresentadas as estruturas de dados para o UFLP. As estruturas consistem numa matriz de custos para armazenar o custo de servir completamente um dado cliente por uma instalação, um vetor contendo os custos fixos de cada instalação e um vetor onde são guardadas as instalações abertas. As instalações que não se encontrem neste vetor assume-se que são instalações fechadas.

***Bloco IX: Método de Melhoramento UFLP***

**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 4.7: Método de Melhoramento para o UFLP

Adicionalmente existe uma estrutura que guarda para cada cliente uma lista duplamente ligada com as instalações abertas e respetivo custo de alocar um cliente, ordenada por ordem crescente dos custos de serviço. Deste modo, pode-se verificar, em qualquer altura, a

cabeça de cada lista para saber que instalação (das instalações abertas) deve ser atribuída ao cliente associado a essa lista. Quando se opta por fechar uma instalação, facilmente se determina qual a nova instalação a alocar a cada cliente que estava atribuído à instalação que vai fechar.

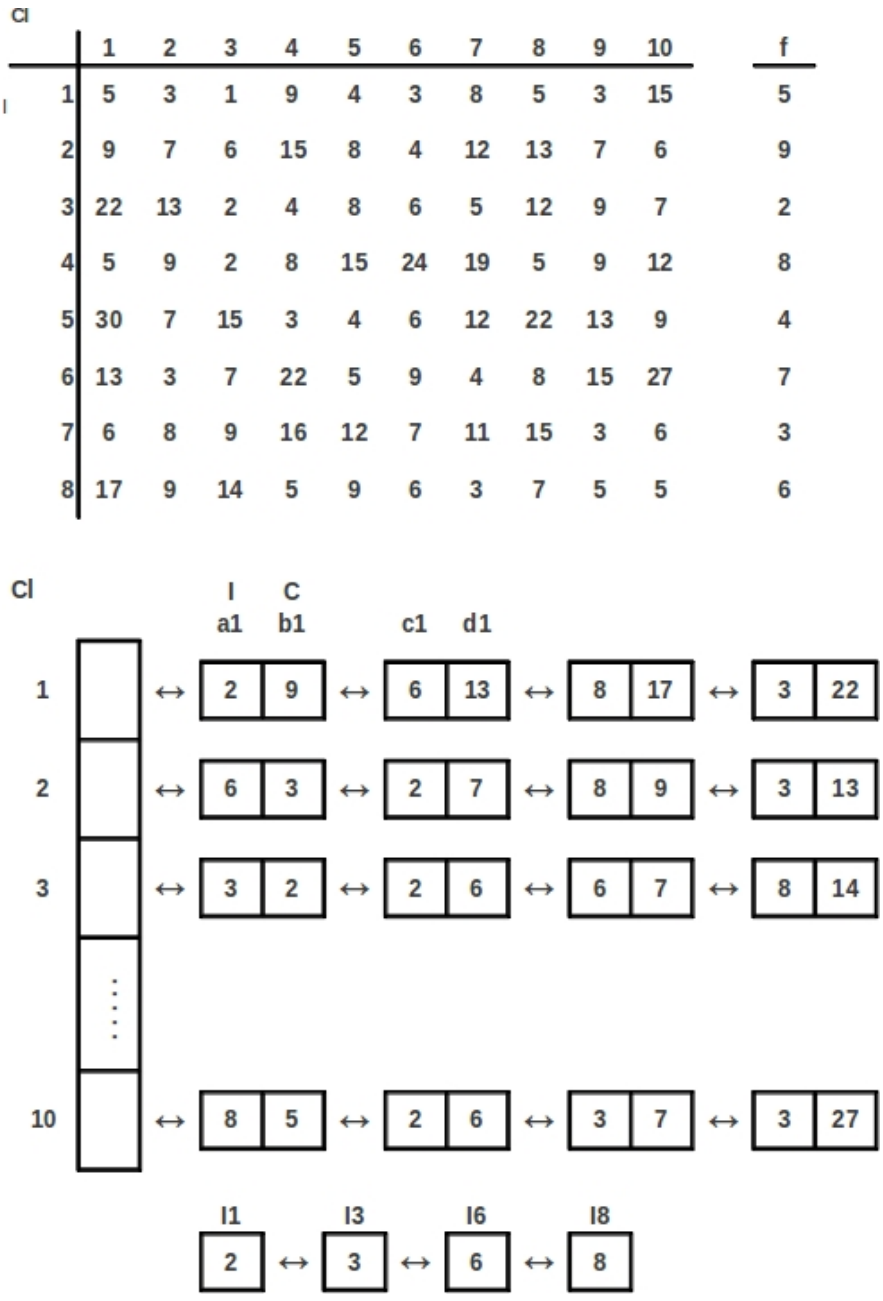


Figura 4.8: Estruturas de dados para o UFLP

## 4.5 Resultados Computacionais

Neste capítulo apresentamos uma comparação dos resultados obtidos com os reportados pelos algoritmos atualmente mais competitivos [13], [12],[19] e [2] e [20].

Para os resultados aqui apresentados, a bateria de testes consiste em 22 instâncias apresentados por Kratica, Filipovic e Ljubic[15] (M*), 60 instâncias pertencentes a dois grupos de problemas criados por Kochetov[22] (FPP), 90 instâncias apresentados por Diptesh Ghosh[23](GHOSH) e 15 instâncias apresentados por Beasley[24](ORLIB).

Os resultados para o algoritmo híbrido proposto por Resende e Werneck[19](H-RW) foram obtidos pelo autor Wayne Pullan[20](PBS) pois este testou o algoritmo numa máquina mais potente do que inicialmente fora testado, produzindo assim melhores resultados ao nível do tempo computacional e da qualidade das soluções.

Para a obtenção deste resultados, o algoritmo PD-RAMP foi testado num portátil, com um processador Intel core 2 Duo 2,5 GHz com 4GB de memória RAM e 7GB de memória *SWAP*. Foi testado também a performance do algoritmo isolado de Pesquisa por Dispersão (SS), produzindo resultados competitivos como podemos ver abaixo na tabela 4.1. Para este algoritmo foi usado um processador Intel I5 2,2GHz com 4 GB de memória RAM e 7GB de memória *SWAP*.

Ambos os algoritmos foram escritos em linguagem C e foi utilizado o compilador gcc (com recurso a otimização) do sistema operativo Ubuntu. A tabela 4.1 compara os desvios em relação aos ótimos e os tempos computacionais.

| | H-RW | | RAMP | | PBS | | **SS** | | **PD-RAMP** | |
|---|---|---|---|---|---|---|---|---|---|---|
| Classes | 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 | 0.59 | 0 | 17.3 | 0 | 0.16 | 0 | 4.83 | 0 | 37.93 |
| FPP | 0 | 94.43 | 0.037 | 2.92 | 0 | 12.26 | 0.015 | 1.67 | 0 | 8.94 |
| GHOSH | 0 | 26.28 | 0.002 | 53.115 | 0 | 15.34 | 0.001 | 9.19 | 0.001 | 41.26 |
| Média | 0 | 30.337 | 0.020 | 21.723 | - | - | 0.005 | 4.460 | $\approx 0$ | 22.450 |

[a] % desvio em relação ao óptimo.
[b] Tempo de processamento em segundos.

Tabela 4.1: Resultados comparativos dos melhores algoritmos conhecidos - UFLP

Podemos ver que que o algoritmo PD-RAMP apresenta resultados extremamente competitivos com os obtidos pelos melhores algoritmos da literatura, com tempos computacionais baixos e resultados de elevada qualidade. Apenas o algoritmo PBS atingiu os resultados ótimos para todas as instâncias, mas o PD-RAMP fica muito perto desses resultados. Para algumas instâncias GHOSH o algoritmo PD-RAMP não conseguiu atingir o ótimo. O uso do algoritmo SS traz grande vantagem a nível computacional pois atinge quase todos os ótimos em tempo computacional reduzido. No caso do algoritmo SS, apenas para as instâncias FPP e GHOSH não conseguiu atingir todos os ótimos. Ao nível do tempo computacional é muito robusto, superando todos os algoritmos aqui apresentados.

# Capítulo 5

# Problema de localização de Instalações com Restrição de Capacidade - CFLP

## 5.1 Introdução

O problema de localização de instalações com restrições de capacidade, também conhecido por *Capacitated Facility Location Problem* ou *Capacitated Plant Location Problem* ou ainda *Capacitated Warehouse Location Problem* (de agora em diante será utilizado o acrónimo **CFLP** para identificar este problema), com m instalações (armazéns, por exemplo) e n clientes pode ser definido da seguinte forma. A cada instalação $i$ está associado um custo fixo de abertura $F_i$, uma capacidade $S_i$ e um custo de transporte $C_{ij}^{tot} = c_{ij} * (x_{ij}/D_j)$ para servir totalmente o cliente $j$ a partir da instalação $i$. Cada cliente $j$ tem uma procura $D_j$. O objetivo é encontrar um conjunto de localizações onde colocar uma instalação de forma a satisfazer a procura de todos os clientes com custo mínimo.

O problema tem a seguinte formulação inteira:

$$Min \qquad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} F_i y_i \tag{5.1}$$

s.a

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1, ..., n \tag{5.2}$$

$$\sum_{j=1}^{n} D_j x_{ij} \leq S_i y_i, \qquad i = 1, ..., m \tag{5.3}$$

$$x_{ij} \leq y_i, \qquad j = 1, ..., n, i = 1, ..., m \tag{5.4}$$

$$x_{ij} \geq 0, \qquad i = 1, ..., m; j = 1, ..., n \tag{5.5}$$

$$y_i \in \{0, 1\}, \qquad i = 1, ..., m \tag{5.6}$$

Nesta formulação, $C = \{1, ..., n\}$ e $A = \{1, ..., m\}$ representam, respetivamente, os conjuntos de clientes e de instalações, as variáveis $x_{ij}$ a quantidade distribuída do cliente $j$ para a instalação $i$, e $y_i$ especifica se a Localização $I$ contém uma instalação aberta ou fechada. As variáveis $D_j$ e $S_i$ representam a procura do cliente $j$ e a capacidade da instalação $i$ respetivamente.

Após a escolha das instalações a abrir é necessário que o problema seja resolvido do ponto de vista da alocação do cliente à instalação. Assim, dado um conjunto de instalações abertas ($y_i = 1, i = 1, ..., m$), o problema transforma-se num problema de transportes. O problema do transportes garante uma solução válida para o CFLP.

O problema de transportes clássico pode ser apresentado da seguinte forma:

$$Min \qquad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{5.7}$$

s.a

$$\sum_{i=1}^{m} x_{ij} = D_j, j = 1, ..., n \tag{5.8}$$

$$x_{ij} = S_i y_i, \qquad i = 1, ..., m \tag{5.9}$$

$$x_{ij} \geq 0, \qquad i = 1, ..., m; j = 1, ..., n \tag{5.10}$$

$$y_i \in \{0, 1\}, \qquad i = 1, ..., m \tag{5.11}$$

Onde $c_{ij}$ é o custo unitário de transporte do cliente $j$ para a instalação $i$, $x_{ij}$ é a quantidade distribuída do cliente $j$ para a instalação $i$, $S_i$ é a capacidade da instalação $i$ e $D_j$ é a procura do cliente $j$.
O objetivo é minimizar o custo total de transporte satisfazendo as restrições de procura do cliente $j$ e a capacidade da instalação $i$.
Para que o problema de transportes tenha solução deve estar equilibrado, ou seja, a procura total dos clientes deve ser igual ao total produzido pela instalações. Caso não esteja equilibrado é necessário verificar uma de duas condições:

**Oferta < Procura** : neste caso é necessário criar uma instalação fictícia com o valor da capacidade desta instalação igual ao valor que falta para igualar a procura e os custos de transporte desta instalação fictícia até aos clientes deverão ser definidos com um valor muito elevado.

**Oferta > Procura** : neste caso é necessário criar um cliente fictício com o valor da procura deste cliente igual ao valor que falta para igualar a oferta e os custos de transporte desta instalação fictícia até aos clientes deverão ser nulos.

## 5.2 Algoritmos Heurísticos para o CFLP

Muito progresso tem sido feito no campo das metaheurísticas no que diz respeito à resolução do CFLP. A primeira heurística para o CFLP foi inicialmente proposta por Jacobsen[25] e teve como base a heurística proposta por Kuehn e Hamburger[26] aplicada ao UFLP. Esta heurística é composta por duas fases: o método *ADD*, que começa com todas as instalações fechadas e abre a instalação que obtém o mínimo custo em abrir. Esta fase termina quando já não existem instalações para abrir que minimizam o custo total. A segunda fase consiste em minimizar o custo total de trocar o estado de uma instalação aberta para fechada e de uma instalação fechada para aberta.

Outro exemplo mais recente é o algoritmo baseado em pesquisa tabu proposto por Minghe Sun[27], em que é utilizada uma estrutura de memória flexível para efetuar pesquisas num espaço de regiões promissor. Este espaço de soluções é guardado em memória para que a direção da pesquisa seja posteriormente acedida e selecionada para efetuar uma pesquisa mais intensa. Este algoritmo aborda também o processo de diversificação e de intensificação como já foi referido no Capítulo 2. A estrutura de pesquisa local é efetuada com a troca do estado de uma instalação (aberta para fechada ou vice-versa). Depois de definido o vetor de instalações abertas que satisfaçam a procura dos clientes, o problema pode ser resolvido pelo problema dos transportes. É usado um algoritmo "*Network Flow*"com base em [28] adaptado pelo autor para a resolução do problema dos transportes. Grolimund e Ganascia[29] também aplicam a pesquisa tabu para a resolução do CFLP.

Um algoritmo *ADD/DROP* é proposto por Bornstein[30] em que para além do método *ADD* utiliza o método *DROP* que consiste em alterar o estado de uma instalação do conjunto de instalações abertas para fechada em que o custo de fechar essa instalação seja mínimo. O autor propõe também o uso de limites inferiores e superiores provenientes da relaxação do problema. É usado o recurso ao problema dos transportes com base em Finke[31].

Algoritmos com base em técnicas de relaxação são frequentemente utilizados para a resolução do CFLP. Cornuejols[32] propôs o uso de várias heurísticas com base em relaxações para a resolução de grandes problemas de localização de instalações com restrições capacidade. Demonstrou também que apenas sete tipos de relaxação obtém diferentes limites inferiores e superiores. Guignard e Spielberg[33] propuseram o método *Dual Ascent* inicialmente proposto por Bilde e Krarup[34] e por Erlenkotter[11], considerando as alterações necessárias para contemplar as capacidades das instalações e a procura dos clientes. É utilizado um algoritmo para a resolução do problema dos transportes baseado em [35][36][37]. Ling[38], Lorena[39] e Beasley[40] também propõe a resolução do CFLP com base em

ralaxações obtendo limites inferiores e superiores para o problema e deste modo um bom ponto de partida para a obtenção de excelentes aproximações para o CFLP. Sridharan[41] também propõe a relaxação do problema original recorrendo ao uso da decomposição cruzada (*Cross Decomposition*) inicialmente proposta por Van Roy[42] em que a ideia geral é a exploração da estrutura primal e dual do problema em simultâneo.

Bornstein[43] propôs o uso de testes reduzidos e critérios de dominância para definir prioridades para determinada instalação poder alterar o seu estado de fechada para aberta ou vice-versa. Utilizou também o Arrefecimento simulado (também conhecida como *Simulated Annealing*) como metaheurística, em que a ideia genérica baseia-se na analogia com o processo térmico para a obtenção do estado de baixa energia de um sólido, processo esse que consiste no aquecimento e posteriormente o arrefecimento lento desse sólido para que átomos se reagrupem uniformemente de modo a gerarem o mínimo de energia. Analogamente, este algoritmo procura substituir a solução atual por uma solução na sua vizinhança $V$ de acordo com uma variável $T$, que mede a temperatura. Quanto maior a temperatura, maior será a componente aleatória e consequentemente maior será a aleatoriedade na escolha dessa solução. Nas seguintes iterações do algoritmo, o valor $T$ é decrementado e o algoritmo tende a aproximar-se de um ótimo local na vizinhança $V$.

Outras metaheurísticas mais recentes e promissoras estão a produzir resultados computacionais interessantes. Colónia de Formigas[44] e Colónia de Abelhas[45] são um exemplo dessas metaheurísticas que se baseiam na observação da natureza para produzir algoritmos que podem não só ser usados para os problemas de localização de instalações, mas para outro tipo de problemas[46].

Ming-Che Lai[47] propôs a Decomposição Benders ou "*Benders Decomposition*"inicialmente proposto por[48]. Genericamente, este algoritmo consiste em dividir o problema em subproblemas mais simples de resolver. O uso do dual para relaxar o problema original, subdividindo as variáveis e restrições do problema dual, faz com que seja possível alternar entre os subproblemas do problema dual e o problema original. O autor utiliza os Algoritmos Genéticos unicamente para a resolução do problema original. Esta metaheurística pertencentes aos algoritmos evolutivos (algoritmos que envolvem a geração de populações de soluções) imitando a seleção natural da evolução das espécies. Com a geração de soluções (populações) elite é aplicado um processo de pesquisa local e posteriormente aplicados fatores de mutação e seleção para gerar novas populações e assim convergir para a solução ótima.

Existem algoritmos que por agruparem mais que uma metaheurísticas são denominados de híbridos, como é caso do algoritmo *GRASP* "híbrido"[49] que também produzem bons resultados em tempos computacionais reduzidos. Também são encontrados algoritmos exatos para a resolução do CFLP, baseados em técnicas de relaxação[50] e também em "*Branch & Bound*"[51]. O termo "*Branch & Bound*"significa divisão e conquista e pode ser descrito como a divisão do problema $P$ em subproblemas $SP_n$ de forma a que possam ser resolvidos os subproblemas mais facilmente para chegar à solução do problema $P$. Este algoritmo obtém limites inferiores e superiores para os subproblemas $SP_n$ e descarta os subproblemas que não estiverem contidos nesses limites. No final, são agrupados os subproblemas para dar lugar ao problema original $P$ obtendo assim a solução final.

## 5.3 Descrição do Algoritmo

O algoritmo proposto para a resolução do CFLP pode ser descrito da seguinte forma: são obtidos soluções pelo método *Dual Ascent* que através de um método de projeção são obtidas as soluções primais correspondentes. Obtendo o conjunto de armazéns abertos é feita a alocação da instalação ao cliente. Inicialmente foi pensado construir uma solução admissível com o *software* CPLEX, através da resolução do problema dos transportes, visto que, obtendo um vetor de instalações abertas, o problema reduz-se ao problema dos transportes. Posteriormente foi implementado um método de construção da solução inicial com base numa atribuição direta por ordem ascendente dos custos de alocação.
A comparação dos dois métodos pode ser vista na tabela 5.1 para as instâncias ORLIB.

Facilmente se pode observar que a resolução do problema dos transportes com o *software* CPLEX traduz-se em grande esforço computacional e a qualidade dos resultados está muito próxima dos resultados obtidos sem a ajuda deste programa. Desta forma optou-se por não utilizar o CPLEX como *software* para a resolução do problema dos transportes mas sim a alocação pelo método direto, que atribui o cliente à instalação pelo custo mínimo, obtendo assim, um ganho computacional muito elevado.
Este método de alocação pode ser descrito da seguinte forma (ver figura 5.1):

Após a solução inicial, segue-se a aplicação do método de melhoramento consistindo no método *ADD* e no método *DROP* para a inserção e remoção de uma instalação, e num método de melhoramento para a alocação cliente/instalação. Com a introdução ou remoção de uma instalação é efetuado o procedimento descrito na figura 5.1. Após um número predefinido de iterações sem melhoramento da função objetivo, é efetuado o melhoramento da alocação cliente/instalação.
De seguida são enunciados os passos do algoritmo, nomeadamente o procedimento *Dual*

***Bloco X: Método para gerar uma solução inicial para o CFLP***

1 Ordenar os clientes $i \in I$ por ordem decrescente da sua procura.

2 Para todos os clientes $i \in I$ fazer:

3 Enquanto o cliente $i$ não tiver sido satisfeito fazer:

4 Efetuar a alocação do cliente $i$ à instalação $j_n \in J$ através do custo de alocação mais baixo. Se a capacidade da instalação $j_n$ estiver nula, alocar o cliente à instalação $j_m$ em que o custo de alocação seja mais baixo.

5 Quando o cliente $i$ estiver satisfeito passar para o cliente seguinte.

6 Terminar.

Figura 5.1: Método para gerar uma solução inicial para o CFLP

*Ascent*, o método de melhoramento baseado no método *ADD* e *DROP* e o método de melhoramento para a alocação cliente/instalação.

### 5.3.1 Método Dual

O método Dual do algoritmo proposto utiliza a relaxação e dualização usada por Guignard et al[33] para a resolução do CFLP. Neste estudo utilizou-se o procedimento *Dual Ascent*, obtendo um limite superior para o problema que, projetando as soluções para o primal, produzem com frequência soluções próximas do ótimo global.
Guignard propôs uma dualização para o problema CFLP com uma particularidade: as capacidades das instalações podem ser ilimitadas ou não (também denominado por CFLP misto - *Mixed Plant Location Problem*). São apresentadas várias fases da dualização proposto por Guignard, mas como no algoritmo proposto somente foi utilizada uma das fases, será essa a ser enunciada. A fase referida é uma generalização da abordagem de Bilde-Krarup[34] aplicada ao CFLP misto. Como a aplicação da dualização foi proposta para o CFLP misto, a adaptação ao CFLP é trivial de se conseguir, bastando para isso não considerar instalações com capacidades ilimitadas.
Assumindo os indices $i = 1, ..., m$ e $j = 1, ..., n$ associados às instalações e aos clientes respetivamente, e considerando o vetor de instalações $y_i \in \{0, 1\}$, $x_{ij}$ a quantidade expedida da instalação $i$ para o cliente $j$, $a_i$ a capacidade da instalação $i$, $b_j$ a procura do cliente $j$,

$c_{ij}$ o custo de expedição da instalação $i$ para o cliente $j$ e por fim o $f_i$ o custo que ocorre quando a instalação $i$ está aberta.

O problema CFLP dado pelas expressões (5.1) até (5.6) pode ser reescrito da seguinte forma:

$$Min \qquad cx + fy \tag{5.12}$$

s.a

$$\sum_j x_{ij} - a_i y_i \leq 0 \tag{5.13}$$

$$-\sum_i x_{ij} \leq -b_j \tag{5.14}$$

$$x_{ij} - m_{ij} \leq 0 \tag{5.15}$$

$$y_i \leq 1 \tag{5.16}$$

$$\sum_i R_{ki} y_i \leq r_k \tag{5.17}$$

$$x_{ij} \geq 0 \tag{5.18}$$

$$y_i \geq 0 \tag{5.19}$$

A correspondente formulação dual é a seguinte:

$$Max \qquad \sum b_j v_j - \sum t_i - \sum r_k \rho_k = z_D \tag{5.20}$$

s.a

$$u_i \geq 0 \tag{5.21}$$
$$v_j \geq 0 \tag{5.22}$$
$$w_{ij} \geq 0 \tag{5.23}$$
$$t_i \geq 0 \tag{5.24}$$
$$\rho_k \geq 0 \tag{5.25}$$
$$e_{ij} = c_{ij} + u_i + w_{ij} - v_j \geq 0 \tag{5.26}$$
$$g_i = f_i + t_i - \sum_j m_{ij} w_{ij} - a_i u_i + \sum_k R_{ij} \rho_k \geq 0 \tag{5.27}$$

De notar que a equação $\sum_i R_{ki} y_i \leq r_k$ impõe restrições quanto ao número de instalações abertas, o que não será necessário usar para o problema em questão. Como o problema trata-se é o CFLP e não o problema misto CFLP, assume-se que $m_{ij} = min\{a_i, b_j\}$, ao invés de $m_{ij} = min\{a_i, \mu_{ij}, b_j\}$, pois $\mu_{ij}$ é a capacidade no arco $\{i, j\}$ e no problema a analisar, temos de ter em consideração a capacidade de todas as instalações.
Assumindo $v_j = min_i c_{ij}$ e $w_{ij} = max\{0, v_j - c_{ij}\}$, $c^*_{ij}$ a reordenação dos custos $c_{ij}$ por ordem crescente ($c^1_j \leq c^2_j \leq c^3_j \leq ... \leq c^m_j$) e $\tau$ uma pequena tolerância e $L$ um número muito grande (representação de infinito). Seja $J$ um vetor contendo $\{1, 2, ..., n\}$ e $J_h$ o $h$-ésimo índice e $H$ a sua cardinalidade.
O procedimento *Dual Ascent* pode ser visto na figura 5.2.

Pela formulação dada em (5.20) podemos ver que a equação (5.27) diz respeito às condições de complementaridade, ou seja, quando $g_i = 0$ para algum $i \in I$ significa que, quando se passa para o espaço de soluções primais, a instalação é aberta. Quando o $g_i > 0$ para algum $i \in I$ significa que, quando se passa para o espaço de soluções primais, a instalação é fechada. No caso do procedimento anterior se verificar que a projeção das soluções para o espaço de soluções primais dá lugar a uma solução inadmissível (caso da oferta ser inferior à procura), deve ser feito uma adição de instalações por ordem crescente da sua capacidade. Poder-se-ia considerar a solução inadmissível, considerando uma limite inferior para o dado problema, mas optou-se por considerar que, para soluções inadmissíveis, são adicionadas as instalações necessárias para tornar a respetiva solução

***Bloco XI: Procedimento Dual Ascent para o CFLP***

1.1 Inicializar os indicadores $\psi_h = 0, h = 1, 2, ..., H$.

1.2. Fazer $h = 0, p = 0$.

2. Fazer $h = h + 1, j = J_h$. Se $\psi_h = 1$, ir para **Passo 6**. Senão fazer $d_j = L$.

3. Se $v_j < c_j^1 < L$ então $v_j = c_j^1$.

4. Determinar $k^*$ como o maior valor $k$ tal que $v_j \geq c_j^k$. Seja:

$I_j = \{i \mid m_{ij} > 0 \wedge v_j - c_{ij} \geq 0\}$

Se $\mid I_j \mid = 0$, ir para **Passo 5.1**. Senão fazer:

$d_j = min_{I_j} g_i / m_{ij}$

5.1. Se $k^* = m$, ou se $c_j^{k^*+1} \geq L$ fazer $\psi_h = 1$ (bloquear o valor de $j$). Senão fazer:

$d_j = min\{d_j, c_j^{k^*+1} - v_j\}$.

5.2. Se $d_j < \rho$, ir para o **Passo 6**. Senão fazer:

$g_i = g_i - m_{ij} d_j, \qquad \forall i : c_{ij}^* - v_j \leq \rho.$

Fazer $v_j = v_j + d_j$ e $p = 1$.

6. Se $h < H$ ir para **Passo 2**.

Se $p = 1$ ir para **Passo 1.2**. Senão Terminar.

Fazer $w_{ij} = w_{ij} + max\{0, v_j - c_{ij}^*\}, \forall i, j$

Figura 5.2: Procedimento *Dual Ascent* para o CFLP

admissível.

### 5.3.2 Método Primal

Para concluir a projeção da solução obtida pelo dual no espaço de soluções do primal é necessário, após obtermos o conjunto de instalações abertas pelo método *Dual Ascent*, proceder à alocação das instalações aos clientes. Para esta alocação foi usado o procedimento descrito na figura 5.1. De seguida é aplicado o método de melhoramento consistindo em melhorar o vetor de instalações abertas obtidas pelo método *Dual Ascent* e melhorar a alocação cliente/instalação dada por 5.1.

Obtendo o vetor de instalações abertas e determinando também a alocação pelo método de alocação direta já referido em 5.1 obtém-se assim um limite superior para o problema e uma solução admissível. Numa primeira fase é efetuado um método de melhoramento durante um número fixo de iterações sem melhoramento de função objetivo, com base na inserção de uma instalação do conjunto de instalações fechadas e remoção de uma instalação do conjunto de instalações abertas. Seja $G_{IN}(s)$ e $G_{OUT}(s)$ o ganho de inserir uma instalação fechada na lista de instalações fechadas e o ganho de remover uma instalação fechada da lista de instalações abertas, e $vs$ o valor da função objetivo.
O método pode ser descrito da seguinte forma (ver figura 5.3):
Numa segunda fase, é efetuado o método de melhoramento para a alocação cliente/instalação. As linhas gerais para este procedimento iterativo consistem em percorrer todos os cliente e efetuar trocas onde existir maior ganho de modo a minimizar consecutivamente a função objetivo e assim atingir um estado em que não exista mais ganho. A figura 5.4 ilustra o tipo de trocas possíveis de obter.

Seja $In \in I$ a instalação que atualmente possui a alocação para o cliente $Ca \in C$ e seja $Im \in I$ a instalação que poderá melhorar ou piorar o custo de alocação para o cliente $Ca \in C$. Como a a instalação $Im \in I$ pode já não ter capacidade para alocar a quantidade a passar da instalação $In \in I$ para $Im \in I$ para o cliente $Ca \in C$, será preciso encontrar uma troca outro cliente $Cb \in C$ para equilibrar as capacidades das instalações e desta forma poder efetuar a troca. Ora, para haver ganho na troca, o ganho $G_Ca(s)$ na passagem do cliente $Ca \in C$ das instalações $In \in I$ para $Im \in I$, ou o ganho $G_Cb(s)$ na passagem do cliente $Cb \in C$ das instalações $In \in I$ para $Im \in I$ terá que ser positivo e maximizado.

Seja $Xan$ e $Xam$ definido como a alocação da instalação $In \in I$ para o cliente $Ca \in C$ e da instalação $Im \in I$ para o cliente $Ca \in C$ e $Ybn$ e $Ybm$ definido como a alocação da instalação $In \in I$ para o cliente $Cb \in C$ e da instalação $Im \in I$ para o cliente $Cb \in C$, respetivamente. $QtDn$ e $QtDm$ será a capacidade restante das instalações $In \in I$ e $Im \in I$ respetivamente.

***Bloco XII: Método de Melhoramento baseado em ADD/DROP***

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

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

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

**Passo 4.** Se $noImprove \geq maxIter$, ir para **Passo 10**.

**Passo 5.** Se $G_{IN}(s) > G_{OUT}(s)$ então abre a instalação definida pela melhor ganho de $G_{IN}(s)$ Senão fecha a instalação definida pela melhor ganho de $G_{OUT}(s)$.

**Passo 6.** Obtém $vs$ com a solução obtida pela alocação em 5.1.

**Passo 7.** Se $vs \leq vs'$ fazer $vs' = vs$, $S' = S$ e $noImprove = 0$.

**Passo 8.** $noImprove = noImprove + 1$.

**Passo 9.** Voltar ao passo **Passo 4**.

**Passo 10.** Terminar.

Figura 5.3: Método de Melhoramento baseado em $ADD/DROP$ para o CFLP

As trocas possíveis serão:

**Caso 1.** Se $G_Ca(s) > 0$ e se existe capacidade da instalação $Im$, ou seja $QtDm > 0$, será o caso mais simples em que há troca direta com melhoramento da função objetivo.

**Caso 2.** Se $G_Ca(s) > 0$ e $QtDm = 0$. Neste caso temos de ver se $G_Ca(s) > G_Cb(s)$ ou $G_Ca(s) < G_Cb(s)$ e ver se $QtDn > 0$. Se houver ganho em qualquer um dos casos pode ser feito o movimento de acordo com o maior ganho. Se não, o movimento não é efetuado, pois levaria a um aumento do valor da função objetivo.

**Caso 3.** Se $G_Ca(s) < 0$ e se não existe capacidade da instalação $Im$, ou seja $QtDm = 0$, é necessário encontrar um cliente $Cb \in C$ que maximize o ganho total de troca de alocação para que, mesmo efetuando a troca em que $G_Ca(s) < 0$, haja um ganho em $G_Cb(s) > 0$ e consequentemente $G_Cb(s) > G_Ca(s)$.

**Caso 4.** Se $G_Ca(s) < 0$ e se existe capacidade da instalação $Im$, ou seja $QtDm > 0$, é necessário encontrar um cliente $Cb \in C$ que maximize o ganho total de troca de alocação para que, mesmo efetuando a troca em que $G_Ca(s) < 0$ haja um ganho em $G_Cb(s) > 0$ e consequentemente $G_Cb(s) > G_Ca(s)$.

Em todos os casos vistos anteriormente, é preciso considerar a situação em que a quantidade $Xan$ alocada da instalação $In \in I$ para o cliente $Ca \in C$ é maior do que a quantidade $Xam$ alocada da instalação $Im \in I$ para o cliente $Ca \in C$ e vice-versa e também a quantidade $Ybn$ alocada da instalação $In \in I$ para o cliente $Cb \in C$ é maior do que a quantidade $Ybm$ alocada da instalação $Im \in I$ para o cliente $Cb \in C$ e vice-versa. Em todos estes casos, todas as trocas possíveis são simétricas, ou seja, verificam-se os ganhos $G_Ca(s)$ e $G_Cb(s)$ com base nas quantidades $Xan > Xam$ ou $Xan < Xam$ e $Ybn > Ybm$ ou $Ybn < Ybm$.

Este método de melhoramento é muito robusto e diminui muito rapidamente a função objetivo pois efetua uma pesquisa alargada na vizinhança do espaço de soluções.

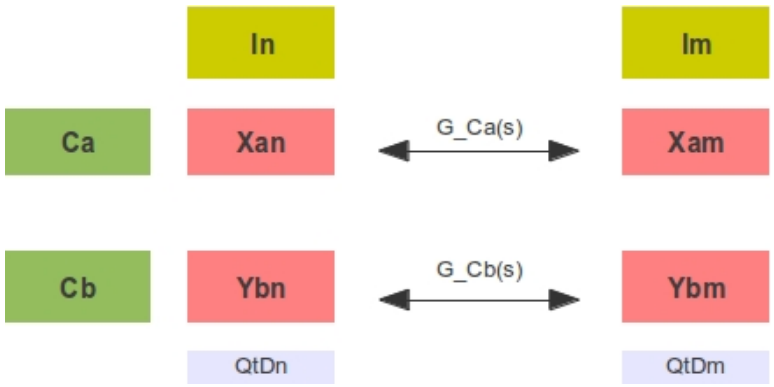


Figura 5.4: CFLP: método de melhoramento para a alocação cliente/instalação

## 5.4 Estruturas de Dados

Na figura 5.5 são apresentadas as estruturas de dados para o CFLP. As estruturas foram baseadas nas estruturas apresentada na figura 4.8, consistindo numa matriz de custos para armazenar o custo de servir completamente um dado cliente por uma instalação, um vetor contendo os custos fixos de cada instalação e um vetor onde são guardados as instalações abertas. As instalações que não se encontrem neste vetor assume-se que são instalações fechadas.

Adicionalmente existe uma estrutura que guarda para cada cliente uma lista duplamente ligada com as instalações abertas e respetivo custo de servir o cliente, ordenada por ordem crescente dos custos de alocação. Existe também uma lista similar à lista referida anteriormente para guardar a alocação do cliente ao armazém e o valor dessa alocação ordenada por ordem ascendente dos respetivos custos de alocação. Uma lista simples contendo a capacidade restante de cada armazém é mantida e atualizada sempre que a lista de alocações é alterada.

## 5.5 Resultados Computacionais

Foram utilizadas várias instâncias standard para testar o desempenho deste algoritmo e efetuar a comparação com os melhores algoritmos conhecidos para o CFLP. As instâncias comuns aos algoritmos comparados foram propostas por Beasley[52] e são denominadas ORLIB. Estas instâncias são de dificuldade reconhecida, sendo constituídas no total por 49 ficheiros de testes, divididos por 4 ficheiros em cada grupo (o asterísco -*- em cada grupo de instâncias significa que agrupa 4 ficheiros), à exceção do ficheiro cap51.

Para a obtenção dos resultados foi utilizado um portátil com um processador Intel core I5 2,67 GHz com 4GB de memória RAM e 7GB de memória *SWAP*. O algoritmo foi escrito em linguagem C e foi utilizado o compilador gcc (com recurso a otimização) do sistema operativo Ubuntu. Como já foi referido, foram obtidas soluções iniciais utilizando dois métodos distintos (ver figura 5.1) mas somente foi utilizado o método de alocação direta.

O algoritmo foi comparado com os melhores resultados da literatura sendo eles produzidos pelo algoritmo de pesquisa tabu do Minghe Sun[27] (TS-S), o algoritmo de Lorena[39]

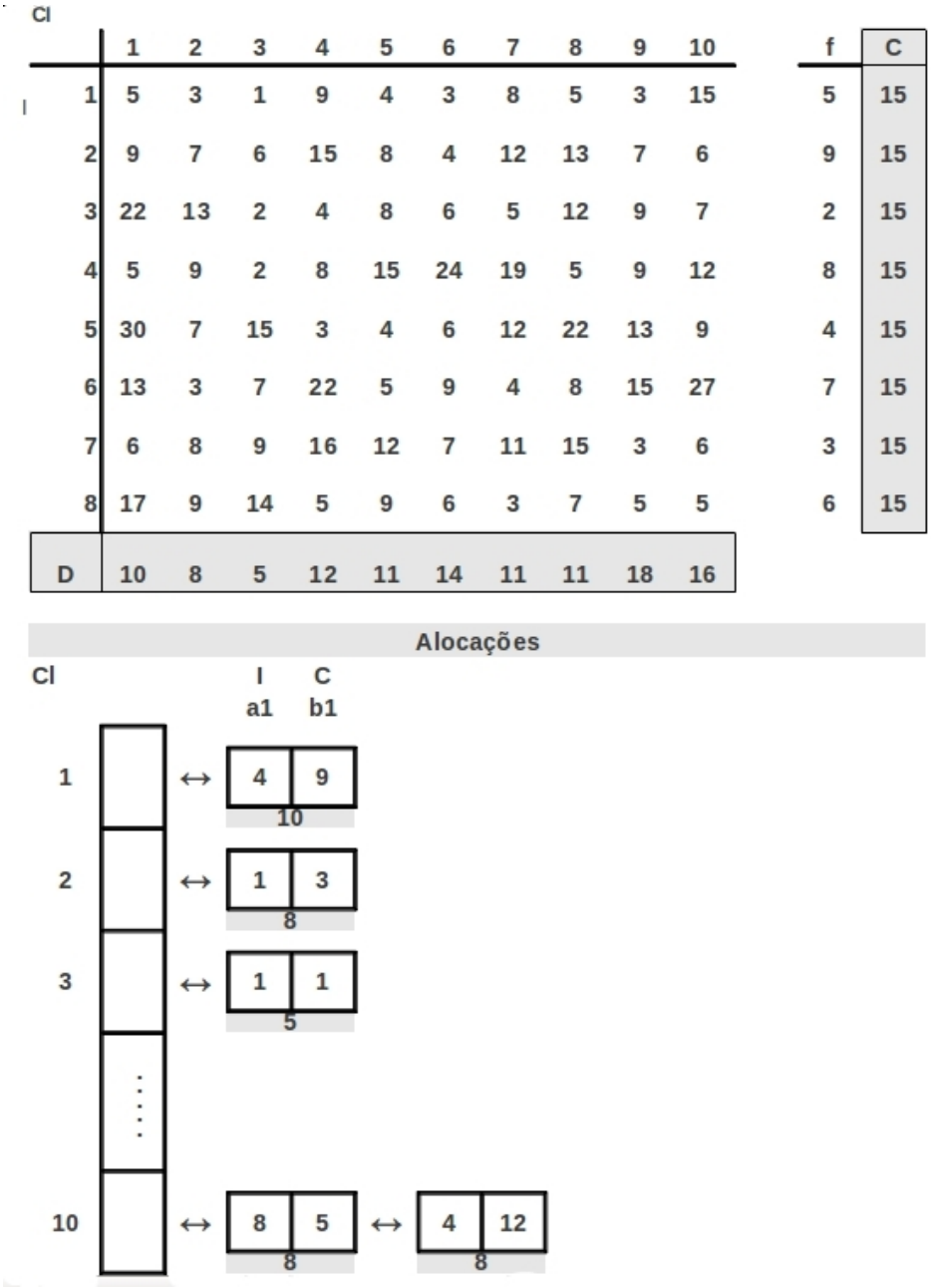


Figura 5.5: Estruturas de dados para o CFLP

(resultados retirados do autor Minghe Sun[27]) e o algoritmo de Beasley[40].

Podemos ver na tabela 5.2 que foram obtidos resultados muito competitivos com os melhores existentes na literatura. Somente com o nível mais baixo de sofisticação RAMP, foi possível superar o autor Lorena[39] quer ao nível do tempo computacional quer ao nível da qualidade das soluções. O algoritmo aqui proposto apresenta resultados a nível de tempos computacionais muito reduzidos, com tempos melhores ou iguais aos restantes algoritmos, para as instâncias cap4*, cap51, cap6*, cap7*, cap8*, cap9*, cap10*, cap11*, cap12* e cap13*, demonstrando que, para problemas de dimensão reduzida, é extremamente rápido, atingindo valores muito próximos do valor ótimo. Já para as instâncias

capa, capb e capc, de grande dimensão, demonstra ser mais lento e mais afastado do valor ótimo do que os algoritmos dos autores Beasley e Minghe Sun.

Sem dúvida que o algoritmo TS-S apresenta os melhores resultados para a resolução do CFLP mas prevê-se que com um nível de sofisticação mais elevado do método RAMP, sejam atingidos resultados ainda melhores do que os aqui apresentados.

| | Sem CPLEX | | | Com CPLEX | | |
|---|---|---|---|---|---|---|
| Classes | Melhor Encontrado | CPU(s) | Dev.[a] | Melhor Encontrado | CPU(s) | Dev.[a] |
| cap41 | 1359121,45 | 0,014 | 30,629 | 1283352,025 | 0,214 | 23,347 |
| cap42 | 1203364,9375 | 0,004 | 9,596 | 1149572,775 | 0,101 | 4,697 |
| cap43 | 1260807 | 0,013 | 9,350 | 1215014,2 | 0,239 | 5,378 |
| cap44 | 1378364,9375 | 0,014 | 11,563 | 1324572,775 | 0,138 | 7,209 |
| cap51 | 1111442,9875 | 0,004 | 8,411 | 1109715,275 | 0,069 | 8,243 |
| cap61 | 1064624,8625 | 0,004 | 14,155 | 1064624,8625 | 0,072 | 14,155 |
| cap62 | 1020000,0625 | 0,004 | 4,316 | 1020000,0625 | 0,058 | 4,316 |
| cap63 | 1100470,1875 | 0,004 | 8,521 | 1100470,1875 | 0,057 | 8,521 |
| cap64 | 1195000,0625 | 0,004 | 14,283 | 1195000,0625 | 0,055 | 14,283 |
| cap71 | 1064624,8625 | 0,003 | 14,155 | 1064624,8625 | 0,027 | 14,155 |
| cap72 | 1020000,0625 | 0,014 | 4,316 | 1020000,0625 | 0,078 | 4,316 |
| cap73 | 1100470,1875 | 0,009 | 8,888 | 1100470,1875 | 0,033 | 8,888 |
| cap74 | 1195000,0625 | 0,003 | 15,462 | 1195000,0625 | 0,042 | 15,462 |
| cap81 | 911190,1 | 0,017 | 8,669 | 911190,1 | 0,216 | 8,669 |
| cap82 | 992077,075 | 0,016 | 8,913 | 992077,075 | 0,267 | 8,913 |
| cap83 | 1117547,2 | 0,006 | 14,516 | 1117547,2 | 0,117 | 14,516 |
| cap84 | 1279577,075 | 0,006 | 19,657 | 1279577,075 | 0,117 | 19,657 |
| cap91 | 876804,55 | 0,004 | 10,062 | 876804,55 | 0,068 | 10,062 |
| cap92 | 946821,025 | 0,004 | 10,644 | 946821,025 | 0,083 | 10,644 |
| cap93 | 1072291,15 | 0,012 | 19,593 | 1072291,15 | 0,094 | 19,593 |
| cap94 | 1234321,025 | 0,005 | 30,471 | 1234321,025 | 0,069 | 30,471 |
| cap101 | 872847,05 | 0,011 | 9,565 | 872847,05 | 0,067 | 9,565 |
| cap102 | 946821,025 | 0,004 | 10,778 | 946821,025 | 0,066 | 10,778 |
| cap103 | 1072291,15 | 0,012 | 19,972 | 1072291,15 | 0,081 | 19,972 |
| cap104 | 1234321,025 | 0,006 | 32,874 | 1234321,025 | 0,041 | 32,874 |
| cap111 | 1006395,6375 | 0,007 | 21,821 | 1006395,6375 | 0,189 | 21,821 |
| cap112 | 1201066,6625 | 0,008 | 33,248 | 1201066,6625 | 0,158 | 33,248 |
| cap113 | 1436911,8625 | 0,009 | 48,049 | 1436911,8625 | 0,146 | 48,049 |
| cap114 | 1726660,8875 | 0,032 | 62,378 | 1726660,8875 | 0,187 | 62,378 |
| cap121 | 991797,3375 | 0,007 | 25,000 | 991797,3375 | 0,113 | 25,000 |
| cap122 | 1176236,7875 | 0,013 | 37,971 | 1176236,7875 | 0,126 | 37,971 |
| cap123 | 1412081,9875 | 0,011 | 57,721 | 1412081,9875 | 0,1 | 57,721 |
| cap124 | 1701831,0125 | 0,014 | 79,888 | 1701831,0125 | 0,27 | 79,888 |
| cap131 | 988591,7625 | 0,014 | 24,596 | 988591,7625 | 0,102 | 24,596 |
| cap132 | 1176236,7875 | 0,014 | 38,138 | 1176236,7875 | 0,118 | 38,138 |
| cap133 | 1412081,9875 | 0,015 | 58,114 | 1412081,9875 | 0,106 | 58,114 |
| cap134 | 1701831,0125 | 0,007 | 83,201 | 1701831,0125 | 0,084 | 83,201 |
| Capa:8 | 79563406,66843 | 0,156 | 313,514 | 79563406,66843 | 26,881 | 313,514 |
| Capa:10 | 79563406,66843 | 0,155 | 331,518 | 79563406,66843 | 21,719 | 331,518 |
| Capa:12 | 79563406,66843 | 0,146 | 347,861 | 79563406,66843 | 20,99 | 347,861 |
| Capa:14 | 79563406,66843 | 0,145 | 363,644 | 79563406,66843 | 19,507 | 363,644 |
| Capb:5 | 43970494,2297 | 0,18 | 221,978 | 43970494,2297 | 76,59 | 221,978 |
| Capb:6 | 43970494,2297 | 0,168 | 229,073 | 43970494,2297 | 38,91 | 229,073 |
| Capb:7 | 43970494,2297 | 0,162 | 233,146 | 43970494,2297 | 32,751 | 233,146 |
| Capb:8 | 43970494,2297 | 0,162 | 236,101 | 43970494,2297 | 28,694 | 236,101 |
| Capc:5 | 37375125,40151 | 0,177 | 220,910 | 37375125,40151 | 52,254 | 220,910 |
| Capc:5.75 | 37375125,40151 | 0,171 | 223,025 | 37375125,40151 | 34,89 | 223,025 |
| Capc:6.5 | 37375125,40151 | 0,156 | 224,472 | 37375125,40151 | 33,565 | 224,472 |
| Capc:7.25 | 37375125,40151 | 0,22 | 224,838 | 37375125,40151 | 41,679 | 224,838 |
| Média | - | 0,048 | 83,460 | - | 8.829 | 83.039 |

[a] % desvio em relação ao óptimo.

Tabela 5.1: Comparação de dois métodos para gerar soluções iniciais - CFLP

| | | TS-$S^1$ | | Lorena$^2$ | | Beasley$^3$ | | **RAMP** | |
|---|---|---|---|---|---|---|---|---|---|
| Classes | Dimensão$^0$ | Dev.$^a$ | CPU$^b$ | Dev.$^a$ | CPU$^b$ | Dev.$^a$ | CPU$^b$ | Dev.$^a$ | CPU$^b$ |
| cap4* | 16x50 | 0 | 0.03 | 5.00 | 0.29 | 0 | 0.68 | 0.69 | 0.04 |
| cap51 | 16x50 | 0 | 0.10 | 0.67 | 0.21 | 0 | 0.80 | 0.58 | 0.04 |
| cap6* | 16x50 | 0 | 0.08 | 0.19 | 0.11 | 0 | 0.49 | 0.18 | 0.05 |
| cap7* | 16x50 | 0 | 0.07 | 0 | 0.05 | - | - | 0 | 0.04 |
| cap8* | 25x50 | 0 | 0.27 | 1.77 | 1.89 | 0 | 2.19 | 1.00 | 0.11 |
| cap9* | 25x50 | 0 | 0.18 | 0.14 | 0.32 | 0.02 | 1.19 | 0.10 | 0.10 |
| cap10* | 25x50 | 0 | 0.18 | 0 | 0.13 | - | - | 0 | 0.09 |
| cap11* | 50x50 | 0 | 0.62 | 0.24 | 33647.75 | 0.11 | 3.20 | 0.22 | 0.47 |
| cap12* | 50x50 | 0 | 0.44 | 0.06 | 0.97 | 0.04 | 1.87 | 0.04 | 0.42 |
| cap13* | 50x50 | 0 | 0.41 | 0 | 0.29 | - | - | 0.05 | 0.41 |
| capa* | 100x1000 | 0.04 | 41.44 | 0.80 | 613.35 | - | - | 1.35 | 54.48 |
| capb* | 100x1000 | 0.17 | 54.18 | 4.19 | 14184.22 | - | - | 1.98 | 110.59 |
| capc* | 100x1000 | 0 | 50.27 | 0.78 | 480.90 | - | - | 3.22 | 137,71 |
| Média | | 0.02 | 11.41 | 1.06 | 3763.88 | - | - | 0.72 | 23.43 |

$^a$ % desvio em relação ao óptimo.

$^b$ Tempo de processamento em segundos.

$*$ Valores médios para as instâncias respetivas.

$^0$ Dimensão n x m, n=instalações e m=clientes.

$^1$ Algoritmo pesquisa tabu do Minghe Sun[27].

$^2$ Algoritmo de Lorena[39] com base em técnicas de relaxação.

$^3$ Algoritmo de Beasley[40] com base em técnicas de relaxação.

Tabela 5.2: CFLP: comparação de resultados com os melhores algoritmos conhecidos

# Capítulo 6

# Conclusões e Trabalho Futuro

Os Problemas de Localização de Instalações são muito estudados na literatura, tendo sido propostos uma grande variedade de algoritmos exatos e heurísticos para resolver eficaz e eficientemente estes problemas. Exemplos práticos de problemas de localização de instalações são a localização ideal de aeroportos, localização de escolas, localização de postos de correios entre outros.

O ramo das metaheurísticas têm cada vez mais um papel importante na obtenção de soluções boas em tempo útil e os resultados apresentados neste trabalho são bom exemplo disso. Sendo os problemas de localização de instalações problemas de difícil de resolução (problemas NP-Difícies), as metaheurísticas têm um papel extremamente importante na resolução destes problemas.

No decorrer desta dissertação foi proposto um algoritmo PD-RAMP para o UFLP e um algoritmo Dual-RAMP para o CFLP, com resultados muito competitivos com os melhores existentes na literatura. Para ambos os problemas foi apresentado um algoritmo que combina o método (*Duas Ascent*) com um método de melhoramento. Face aos resultados apresentados, nomeadamente para o UFLP, o PD-RAMP superou a versão mais simples RAMP e conseguiu resultados melhores ao nível da qualidade das soluções. O algoritmo PD-RAMP apresenta um método de melhoramento, baseado em pesquisa por dispersão, em que utiliza estratégias de combinação de soluções e geração de subconjuntos para produzir novas soluções, o que faz com que este método evolutivo efetue a exploração num espaço de soluções mais abrangente e talvez promissor, levando a que sejam obtidas informações acerca das soluções que levem ao encontro de soluções ótimas, ou muito perto das ótimas, tornando assim este método de melhoramento muito robusto e eficiente. Vimos também que o algoritmo SS (Pesquisa por Dispersão) produziu resultados interessantes em tempo computacional reduzido. Adicionalmente, os resultados obtidos com a aplicação do PD-RAMP ao UFLP (versão mais simples do problema de localização

de instalações em que não se consideram as capacidades das instalações nem a procura dos clientes), foi possível determinar a melhor abordagem para o CFLP (versão do problema de localização de instalações em que se considera as capacidades das instalações e a procura dos clientes).

Para o CFLP foi proposto um algoritmo baseado na versão mais simples do método RAMP, em que somente o lado dual é explorado mais intensamente, produzindo resultados competitivos com os melhores algoritmos. Para além do método RAMP na versão mais simples, foi também utilizado uma heurística de melhoramento que, por vezes, não consegue escapar do ótimo local. Conjetura-se que, com a utilização de um método de melhoramento baseado numa metaheurística, consiga-se escapar do ótimo local, levando a resultados ainda melhores do que os aqui apresentados. Podemos também afirmar que uma possível implementação da versão mais sofisticada do método RAMP, o PD-RAMP, irá permitir uma exploração de ambos os lados primal e dual, podendo levar a resultados ainda mais competitivos. Não se efectuou essa implementação pois os resultados obtidos foram suficientes para a discussão de resultados com outros algoritmos da literatura. A aplicação do método RAMP quer para o UFLP quer para o CFLP constitui um bom ponto de partida para o estudo do Problema de Localização de Instalações com Restrições de Capacidade com um Único Servidor (*Single-Source Capacitated Facility Location Problem* - SSCFLP) permitindo a identificação das restrições a relaxar, o tipo de ralaxação e o método de melhoramento mais promissor a utilizar no problema.

Podemos afirmar que a abordagem RAMP pode competir com qualquer abordagem existente até ao momento para os problema UFLP e CFLP. A sua vantagem reside na exploração dos lados primal e dual que originam boas soluções em tempos computacionais aceitáveis. O sucesso desta abordagem sugere que possa ser aplicada com o mesmo nível de êxito a outras extensões dos problemas de localização de instalações

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

[3] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal, *Numerical Optimization – Theoretical and Practical Aspects.* Universitext, Springer Verlag, Berlin, 2006.

[4] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, (New York, NY, USA), pp. 151–158, ACM, 1971.

[5] E. L. Lawler and D. E. Wood, "Branch-And-Bound Methods: A Survey," *Operations Research*, vol. 14, no. 4, pp. 699–719, 1966.

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

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

[8] E. 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, pp. 1–16, November 2001.

[9] F. Glover, M. Laguna, and R. Marti, "Fundamentals of scatter search and path relinking," *CONTROL AND CYBERNETICS*, vol. 29, no. 3, pp. 653–684, 2000.

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

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

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

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

[14] 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. Combinatorial Optimisation 96 Conference (CO96), London, England, 1996.

[15] J. Kratica, D. Tosic, V. Filipovic, and I. Ljubic, "Solving the simple plant location problem by genetic algorithm," *RAIRO-RECHERCHE OPERATIONNELLE-OPERATIONS RESEARCH*, vol. 35, pp. 127–142, JAN-MAR 2001.

[16] M. Gen, Y. Tsujimura, and S. Ishizaki, "Optimal design of a Star-LAN using neural networks," *COMPUTERS & INDUSTRIAL ENGINEERING*, vol. 31, pp. 855–859, DEC 1996. 18th International Conference on Computers and Industrial Engineering (ICC&IE 95), SHANGHAI, PEOPLES R CHINA, OCT 25-27, 1995.

[17] S. Vaithyanathan, L. Burke, and M. Magent, "Massively parallel analog tabu search using neural networks applied to simple plant location problems," *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, vol. 93, pp. 317–330, SEP 6 1996.

[18] M. Guignard, "A lagrangean dual ascent algorithm for simple plant location problems," *European Journal of Operational Research*, vol. 35, no. 2, pp. 193–200, 1988.

[19] 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, OCT 1 2006.

[20] 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. World Summit on Genetic and Evolutionary Computation (GEC 09), Shanghai, PEOPLES R CHINA, JUN 12-14, 2009.

[21] F. Maia, D. Gamboa, and T. Matos, "Pesquisa por dispersão para problemas de localização de instalações," tech. rep., Escola Superior de Gestão e Tecnologia de Felgueiras, 2011.

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

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

[24] 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.

[25] J. S. K., "Heuristics for the capacitated plant location model," *European Journal of Operational Research*, vol. 12, pp. 253–261, 1983.

[26] A. A. Kuehn and M. J. Hamburger, "A heuristic program for locating warehouses," *Management Science*, vol. 9, pp. 643–666, July 1963.

[27] M. Sun, "A tabu search heuristic procedure for the capacitated facility location problem," *Journal of Heuristics*, pp. 1–28, 2011. 10.1007/s10732-011-9157-3.

[28] J. L. Kennington and R. V. Helgason, *Algorithms for Network Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1980.

[29] S. Grolimund and J.-G. Ganascia, "Driving tabu search with case-based reasoning," *European Journal of Operational Research*, vol. 103, no. 2, pp. 326–338, 1997.

[30] C. Thomas and C. M. Bornstein, "An ADD/DROP procedure for the capacitated plant location problem," *Pesquisa Operacional*, vol. 24, pp. 151–162, 04 2004.

[31] J. H. Ahrens and G. Finke, "Primal transportation and transshipment algorithms," *Mathematical Methods of Operations Research*, vol. 24, pp. 1–32, 1980. 10.1007/BF01920269.

[32] G. Cornuejols, R. Sridharan, and J. M. Thizy, "A comparison of heuristics and relaxations for the capacitated plant location problem," *European Journal of Operational Research*, vol. 50, no. 3, pp. 280–297, 1991.

[33] M. Guignard and K. Spielberg, "A direct dual method for the mixed plant location problem with some side constraints," *Mathematical Programming*, vol. 17, pp. 198–228, 1979. 10.1007/BF01588244.

[34] O. Bilde and J. Krarup, "Sharp lower bounds and efficient algorithms for the simple plant location problem," in *Studies in Integer Programming* (E. J. B. K. P.L. Hammer and G. Nemhauser, eds.), vol. 1 of *Annals of Discrete Mathematics*, pp. 79–97, Elsevier, 1977.

[35] L. R. Ford and D. R. Fulkerson, "Solving the transportation problem," *Management Science*, vol. 3, no. 1, pp. 24–32, 1956.

[36] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 1962.

[37] D. R. Fulkerson, "An out-of-kilter method for minimal-cost flow problems," *Journal of the Societyfor Industrial and Applied Mathematics*, vol. 9, pp. 18–27, Mar. 1961.

[38] L.-Y. Wu, X.-S. Zhang, and J.-L. Zhang, "Capacitated facility location problem with general setup cost," *Computers & OR*, vol. 33, pp. 1226–1241, 2006.

[39] L. A. Lorena and E. L. F. Senne, "Improving traditional subgradient scheme for lagrangean relaxation: an application to location problems," *International Journal of Mathematical Algorithms*, vol. 1, pp. 133–151, 1999.

[40] J.E. and Beasley, "Lagrangean heuristics for location problems," *European Journal of Operational Research*, vol. 65, no. 3, pp. 383–399, 1993.

[41] R. Sridharan, "The capacitated plant location problem," *European Journal of Operational Research*, vol. 87, no. 2, pp. 203–213, 1995.

[42] T. J. Van Roy, "A cross decomposition algorithm for capacitated facility location," *Oper. Res.*, vol. 34, pp. 145–163, January 1986.

[43] C. T. Bornstein and H. B. Azlan, "The use of reduction tests and simulated annealing for the capacitated plant location problem," *Location Science*, vol. 6, no. 1-4, pp. 67–81, 1998.

[44] H. Venables and A. Moscardini, "An adaptive search heuristic for the capacitated fixed charge location problem," in *Ant Colony Optimization and Swarm Intelligence* (M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, eds.), vol. 4150 of *Lecture Notes in Computer Science*, pp. 348–355, Springer Berlin / Heidelberg, 2006. 10.1007/11839088_32.

[45] L. T. V. and T. E. A., "Development of a bee colony optimization algorithm for the capacitated plant location problem," in *International conference: Optimization and applications*, II, 2011.

[46] M. Dorigo and T. Stützle, "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances," in *Handbook of Metaheuristics* (F. Glover and G. Kochenberger, eds.), vol. 57 of *International Series in Operations Research & Management Science*, ch. 9, pp. 250–285–285, New York: Springer, 2003.

[47] M.-C. Lai, H. suk Sohn, T.-L. B. Tseng, and C. Chiang, "A hybrid algorithm for capacitated plant location problem," *Expert Systems with Applications*, vol. 37, no. 12, pp. 8599–8605, 2010.

[48] 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.

[49] R. Silva, "Um algoritmo grasp híbrido para o problema de localização capacitada de custo fixo," Master's thesis, Universidade Federal do Rio de Janeiro, COPPE, Feb. 2007.

[50] J. E. Beasley, "An algorithm for solving large capacitated warehouse location problems," *European Journal of Operational Research*, vol. 33, no. 3, pp. 314–325, 1988.

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

[52] 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.