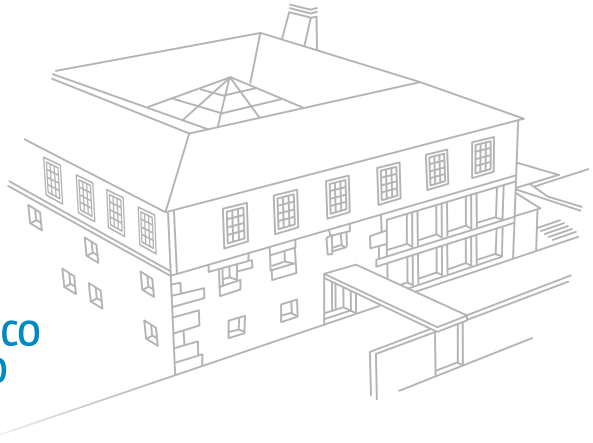

ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Algoritmos RAMP para o problema P-Median

DESIGNAÇÃO DO MESTRADO Mestrado em Engenharia Informática

AUTOR José Carlos Sousa Veloso

ORIENTADOR(ES) Doutora Dorabela Regina Chiote Ferreira Gamboa

ANO 2014

Dedico este trabalho a todos aqueles que lutam pela concretização dos seus projetos. Que tenham a força e a persistência necessária para nunca se darem por vencidos...

Agradecimentos

Não podia deixar de agradecer a todos aqueles que me ajudaram de alguma forma e tornaram possível a concretização deste projeto. Gostaria, em particular, de agradecer aos meus pais que com o esforço do seu trabalho foram capazes de proporcionar todas as condições necessárias à minha formação, à minha irmã e avós pois sempre me ajudaram a ter uma visão positiva do futuro, à minha namorada que sempre me acompanhou e apoiou, aos meus colegas do Centro de Inovação e Investigação em Ciências Empresariais e Sistemas de Informação, sempre disponíveis para me auxiliar, e a todos os meus professores da Escola Superior de Tecnologia e Gestão de Felgueiras, por todo o conhecimento que me transmitiram.

Por fim, um agradecimento muito especial à minha orientadora, Professora Doutora Dorabela Gamboa, que para além de me guiar ao longo de todo este projeto me motivou continuamente a melhorar o meu trabalho.

Resumo

A vasta aplicabilidade dos Problemas de Localização de Instalações em variados cenários do mundo real (como a escolha da localização de um hospital ou armazém) associada à complexidade de resolução caraterística destes problemas, atrai grande atenção por parte da comunidade científica, que procura continuamente novos métodos de resolução, mais eficazes e eficientes.

O presente trabalho incide naquele que é considerado um dos Problemas de Localização de Instalações mais estudado: o problema *P-Median*. Este problema tem por finalidade a escolha de um conjunto de *p* instalações (medianas) de entre um conjunto de instalações candidatas, de modo a minimizar o somatório da distância de cada cliente à respetiva mediana mais próxima. Uma vez que a resolução deste problema através de métodos exatos implica recursos computacionais elevados, com alguma naturalidade surgem abordagens heurísticas, que garantem boas soluções com recursos computacionais reduzidos.

Neste estudo são apresentados dois novos algoritmos para a resolução do problema *P-Median* baseados na metaheurística RAMP (*Relaxation Adaptive Memory Programming*), caraterizada pela sua eficiente exploração dos espaços primal e dual de um problema.

Os novos algoritmos propostos, Dual-RAMP e PD-RAMP, produzem resultados de qualidade que demonstram o sucesso da abordagem RAMP na resolução do problema *P-Median*.

Palavras-chave: Problemas de localização, RAMP, problema *P-Median*, Metaheurísticas.

Abstract

The wide applicability of Facility Location Problems in various real-world scenarios (such as determining the location of an hospital or a warehouse) associated with the well-known resolution complexity, attracts great attention from the scientific community, that persistently seeks more capable and efficient methods.

This work focuses on what is considered one of the most studied Facilitiy Location Problem: the P-Median problem. The goal is to choose the set of p facilities (medians) from a set of candidate facilities, that minimizes the sum of the distance of each customer to the closest respective median. Solving this problem using exact methods implies high computational resources, hence heuristic approaches arise naturally, since they ensure good solutions with reduced computational resources.

In this study we present two new algorithms for solving the P-Median problem, based on the metaheuristic RAMP (Relaxation Adaptive Memory Programming), which allows an efficient exploitation of primal and dual spaces of the problem.

The new proposed algorithms, Dual-RAMP and PD-RAMP, yield high quality results that demonstrate the success of the RAMP approach for solving the P-Median problem.

Keywords: Location problems, RAMP, P-Median problem, Metaheuristics.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Os problemas de localização de instalações são problemas de otimização combinatória que têm por objetivo escolher a localização de um conjunto de instalações de modo a satisfazer as necessidades de um conjunto de clientes, respeitando possíveis restrições inerentes, como o número de instalações pelas quais um cliente poderá ser servido. Estes problemas têm uma vasta aplicabilidade em diversas áreas do mundo real, como a escolha da localização de um banco, hospital ou planta industrial [1-2].

Desde a sua introdução por Alfred Weber em 1909 [3], que considerou localizar um único armazém de modo a minimizar a distância total deste armazém a vários clientes, o estudo dos problemas de localização atrai grande atenção por parte da comunidade científica (justificada pela aplicabilidade que estes problemas apresentam em contextos reais), que procura constantemente modelar, formular e solucionar esta classe de problemas, através de métodos cada vez mais eficientes.

A complexidade caraterística dos problemas de localização de instalações leva a que estes sejam classificados problemas NP-Difíceis [4], o que significa que não é conhecido qualquer algoritmo capaz de garantir a solução ótima de todas as instâncias de um problema deste tipo em tempo polinomial. Esta caraterística torna desaconselhado o uso de métodos exatos para a resolução destes problemas pois, embora garantam a obtenção da solução ótima do problema, implicam grandes recursos computacionais. Deste modo, são sugeridas abordagens baseadas em métodos heurísticos que, embora não garantam a obtenção da solução ótima do problema, são capazes de garantir resultados de boa qualidade com recursos computacionais mais reduzidos, comparativamente aos requeridos pelos métodos exatos.

O estudo contínuo de métodos de resolução heurísticos levou ao aperfeiçoamento dos métodos já existentes, bem como ao surgimento de novos métodos, onde se destacam as metaheurísticas (métodos gerais de resolução de problemas de otimização combinatória que aliam ao método de pesquisa local estratégias que evitam o aprisionamento em ótimos

locais), que estão na génese de algoritmos cada vez mais capazes de garantir soluções de elevada qualidade.

RAMP (*Relaxation Adaptive Memory Programming*) é uma metaheurística proposta por Rego [5], caraterizada por permitir combinar de forma iterativa a informação obtida no espaço primal (problema original com todas as suas restrições) e dual (problema obtido através da remoção de pelo menos uma das restrições do problema original) de um problema de otimização combinatória, possibilitando uma exploração eficiente da relação existente entre os dois espaços de soluções de um problema. A abordagem RAMP possuí níveis distintos de sofisticação, sendo que na sua versão mais simples, designada por Dual RAMP, é explorado com maior intensidade o espaço dual do problema, enquanto que a versão mais sofisticada, o PD-RAMP, acrescenta ao Dual RAMP um método evolutivo que permite fortalecer a relação existente entre o espaço primal e dual de um problema.

A criação da metaheurística RAMP inovou a forma como eram explorados os espaços de soluções de um problema no sentido em que, até então, as metaheurísticas focavam-se em apenas um dos espaços. Esta abordagem provou ser capaz de produzir resultados de elevada qualidade em diversos problemas de otimização combinatória, como o Problema de Ordenação Linear (*Linear Ordering Problem*) [6], o Problema de Localização de Instalações sem Restrições de Capacidade (*Uncapacitated Facility Location Problem*) [6], o Problema da Árvore de Expansão Mínima com Restrições de Capacidade (*Capacitated Minimum Spanning Tree*) [7], o Problema de Localização de Hubs com Afetação Múltipla e sem Restrições (*Uncapacitated Multiple Allocation Hub Location Problem*) [8], o Problema de Localização de Instalações com e sem restrições de capacidade (*Capacitated/Uncapacitated Facility Location Problem*) [9] e o Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor (*Single Source Capacitated Facility Location Problem*) [10].

No presente estudo é aplicada a metaheurística RAMP ao problema *P-Median*. Este problema é considerado um dos problemas mais representativo dos problemas de localização de instalações, bem como um dos problemas mais estudados. Introduzido por Hakimi em 1964 [11], o problema *P-Median* é um problema NP-Difícil [12] que consiste em escolher *p* instalações (medianas) de entre um conjunto de instalações candidatas, de modo a minimizar o somatório da distância de cada cliente à sua mediana mais próxima.

O objetivo deste estudo passa por desenvolver algoritmos baseados na abordagem RAMP, que permitam avaliar se a aplicação desta metaheurística ao problema *P-Median* permite obter resultados de elevada qualidade à semelhança do que acontece com outros problemas de otimização combinatória já enunciados.

A dissertação encontra-se estruturada como a seguir se descreve:

- No capítulo dois serão descritos métodos para a resolução de problemas de otimização combinatória. O capítulo inicia com uma breve exposição de conceitos básicos que facilitarão a leitura e compreensão do restante documento. De seguida serão abordados métodos exatos e métodos aproximados, seguindo-se a descrição de algumas metaheurísticas, nomeadamente, o GRASP (*Greedy Randomized Adaptive Search Procedure*), a Pesquisa *Tabu*, a Pesquisa por Dispersão e RAMP. Por fim serão analisadas diferentes técnicas de relaxação matemática.
- No capítulo três são descritos os problemas de localização de instalações e é apresentado o estado da arte do problema *P-Median*.
- No capítulo quatro são descritos os algoritmos RAMP propostos para a resolução do problema *P-Median*.
- Por último, no capítulo cinco, são apresentadas as conclusões deste estudo e as propostas de trabalho futuro.

.

2. Métodos para a Resolução de Problemas de Otimização Combinatória

Os problemas de optimização são problemas cujo objetivo é encontrar a melhor solução possível de entre aquelas consideradas admissíveis para o problema em questão. Estes problemas podem ser divididos em diferentes categorias, de acordo com o tipo de variáveis: discretas (conjunto de valores admissível limitado, como a quantidade de casas de uma cidade) ou contínuas (a variável pode assumir qualquer valor de um intervalo, sendo por isso admitidos valores reais, como por exemplo, o peso de uma pessoa).

Chama-se problema de otimização combinatória a um problema de otimização cujas variáveis são discretas. Os problemas de otimização combinatória procuram alocar de forma eficiente recursos limitados considerando um determinado objetivo, como a minimização do somatório total das distâncias dos clientes de uma loja à respetiva loja mais próxima.

O conjunto de alternativas possíveis no âmbito de um problema de otimização é limitado não só pelos valores que as variáveis de decisão podem assumir (apenas valores de conjuntos limitados e discretos), como também por restrições adicionais, como por exemplo, o número máximo de clientes que uma determinada loja poderá atender.

O processo de resolução de um problema de otimização envolve geralmente vários passos, nomeadamente, a identificação do problema, a definição do problema, a modelação do problema, a resolução do modelo do problema, a validação das soluções obtidas e, por fim, a implementação de uma das soluções [13].

No quotidiano, quer indivíduos quer organizações são constantemente confrontados com problemas de otimização, sendo que a alternativa escolhida para solucionar esse problema terá sempre um impacto no individuo/organização, impacto esse medido segundo uma escala de avaliação dessa entidade. É por isso de grande interesse o estudo desta área, na tentativa de melhorar cada vez mais as ferramentas existentes para cada um dos passos da resolução de um problema de otimização combinatória.

São apresentados de seguida alguns conceitos considerados pertinentes para uma melhor leitura e compreensão deste estudo, seguindo-se uma descrição de métodos de resolução de problemas de otimização combinatória. O capítulo termina com a exposição de diferentes técnicas de relaxação matemática.

2.1. Definições e Conceitos Fundamentais

Esta secção é destinada à descrição de alguns conceitos fundamentais para este estudo e que deverão ser bem entendidos para uma melhor compreensão do presente trabalho.

Espaço de soluções admissíveis

Conjunto de todas as soluções para um problema de otimização que respeitam todas as restrições que definem esse problema.

Estrutura de vizinhança

Considere que S representa o espaço de soluções admissíveis de um problema P e que uma dada solução $s \in S$. Denomina-se vizinhança de s, $V(s)$, o conjunto de todas as soluções $V(s) \subseteq s$ possíveis de ser alcançadas desde s através de um movimento, específico para cada estrutura de vizinhança.

Grafo

Diagrama composto por um conjunto de pontos e linhas que unem alguns pares desses pontos. Em termos matemáticos, um grafo é um tríplice ordenado $(V(G),E(G)),\Psi_G)$ em que $V(G)$ representa um conjunto não vazio de vértices, $E(G)$ representa um conjunto de arestas (ou arcos) e Ψ_G uma função de incidência que associa cada aresta pertencente a E a um par não ordenado de vértices pertencentes a G [14].

A figura seguinte representa um grafo composto por cinco vértices e seis arestas:

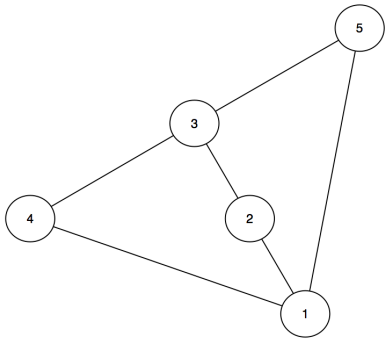

Figura 1 – Grafo.

Um grafo será considerado simples se não possuir ciclos nem arestas paralelas ou cujo vértice de partida ou destino seja o mesmo (lacete). O grafo é designado pesado caso cada aresta do mesmo tenha um custo ou peso associado (num problema do mundo real esse peso poderá ser a distância entre dois pontos), e completo se cada um dos pares de vértices pertencentes a V se encontrar ligado por uma aresta. Por fim, chama-se grafo orientado a um grafo cuja direção que define o vértice de partida e chegada é explicita para cada uma das arestas.

Instância de um problema

Uma instância de um problema pode ser vista como uma concretização desse problema. Uma instância é formada por um par (S, f) em que S representa o espaço de soluções admissíveis e f uma função de avaliação que atribui um valor real a cada uma das soluções $s \in S$ [13].

Ótimo local

Melhor solução obtida por um algoritmo com uma determinada estrutura de vizinhança.

Ótimo global

Solução ótima de um problema.

Variável de decisão

Variáveis de um problema cujo valor deverá ser determinado. Considere-se como exemplo uma variável que representa o estado de um armazém (aberto – 1; fechado – 0). Para que o problema seja resolvido é necessário que seja determinado qual o valor dessa variável, que neste caso poderá ser 0 ou 1.

2.2. Métodos exatos e métodos aproximados

Um problema de otimização combinatória é composto por um vasto número de alternativas possíveis que podem ser exploradas na tentativa de encontrar a melhor existente. A abrangência da exploração do domínio de alternativas determina a existência de dois métodos distintos de resolução de um problema de otimização: métodos exatos e métodos aproximados.

Um método exato avalia a totalidade das possíveis alternativas existentes para solucionar um problema, garantindo deste modo que será encontrada a solução ótima do problema. Contudo, a avaliação de todo o espaço de alternativas implica a existência de elevados

recursos computacionais, o que nem sempre se verifica. Um algoritmo bem representativo dos métodos exatos é o *Branch and Bound* introduzido por Lang e Doid [15].

A natureza dos métodos exatos torna-os mais apropriados para a resolução de problemas cuja dificuldade de resolução cresce de modo polinomial de acordo com o tamanho do problema, situação que não se verifica em problemas classificados como NP-Difíceis. Nestes, o esforço de resolução é exponencial e mesmo para instâncias de pequena e média dimensão será impraticável recorrer a métodos exatos [13], sugerindo-se por isso o uso de métodos aproximados.

Um método aproximado é capaz de obter, em tempo polinomial, uma solução admissível cujo valor da função objetivo se encontra num intervalo de qualidade aceitável, ou seja, não dista muito do valor da solução ótima [16]. Uma vez que, ao contrário dos métodos exatos, um método aproximado apenas explora uma porção do domínio global de alternativas existentes, os recursos exigidos são consideravelmente inferiores.

A concepção de algoritmos, quer exatos, quer aproximados, implica descodificar a estrutura do problema, de forma a aplicar técnicas algorítmicas que explorem eficientemente essa estrutura. Contudo, a estrutura utilizada para a implementação de métodos aproximados deverá ser mais sofisticada, de modo a determinar eficazmente quais as alternativas que deverão ser exploradas [16].

2.3. Heurísticas

As heurísticas são técnicas de resolução de problemas frequentemente aplicadas a problemas NP-Difíceis. Permitem resolver de forma rápida um dado problema, tendo por base aspectos empíricos retirados da resolução de problemas semelhantes. Quando comparadas com as soluções obtidas através de métodos aproximados, a qualidade das soluções originadas por métodos heurísticos poderá ser inferior, já que estes não garantem que a solução se encontra num intervalo próximo à solução ótima, mas como contrapartida, os recursos exigidos são menores.

Podemos classificar as heurísticas em três categorias: construtivas, de melhoramento (ou pesquisa local) e metaheurísticas. As heurísticas construtivas são caraterizadas por construírem, de forma iterativa, uma solução admissível para um problema, sendo por isso muitas vezes o ponto de partida para os métodos subsequentes de um algoritmo para a resolução de um problema de otimização combinatória. A construção da solução é considerada iterativa uma vez que, em cada iteração do método, é adicionado um novo elemento à solução. Por sua vez, as heurísticas de melhoramento são algoritmos que

procuram melhorar uma solução inicial (que poderá ser construída com recurso a uma heurística construtiva) com o objetivo de encontrar uma solução de melhor qualidade. Em cada passo de um algoritmo de pesquisa local ou melhoramento, são exploradas as soluções vizinhas da solução corrente. A estrutura de vizinhança utilizada irá definir o espaço de soluções vizinhas e, em última instância, a qualidade da solução encontrada pela heurística de melhoramento. Caso na iteração atual da heurística de melhoramento for encontrada uma solução de qualidade superior àquela da solução corrente será efetuada uma nova iteração, caso contrário o procedimento termina e considera-se que foi encontrado um ótimo local (que poderá, em alguns casos, coincidir com o ótimo global).

Na Figura 2, é apresentado um algoritmo de pesquisa local que reflete o comportamento anteriormente descrito, onde f representa a função objetivo de um problema de minimização (problema que tem por objetivo encontrar a solução com menor valor da função objetivo), que $f(S)$ representa o valor da função objetivo de uma solução S e que $V(S)$ representa o espaço de soluções vizinhas da solução S.

1. Obter uma solução inicial admissível, S_0.
2. Igualar a solução corrente S_c à solução inicial S_0.
3. Considerar uma solução $S_v \in V(S_c)$.
4. Se $f(S_v) < f(S_c)$ então
5. $S_c = S_v$
6. Senão
7. $V(S_c) -= S_v$
8. Se $V(S_c) \neq \emptyset$ então
9. Ir para **3.**
10. Senão
11. Terminar.

Figura 2 - Algoritmo de melhoramento.

As heurísticas de melhoramento tendem a ficar “presas” em ótimos locais, fazendo com que o procedimento termine antes que seja encontrado o ótimo global. As metaheurísticas tentam contornar esta problemática acrescentando aos métodos heurísticos estratégias que visam escapar aos ótimos locais, nomeadamente o facto de permitirem piorar a solução corrente.

De seguida será feita uma descrição mais aprofundada das metaheurísticas.

2.4. Metaheurísticas

Metaheurística é a designação dada a métodos gerais de resolução que aliam procedimentos de pesquisa local a estratégias capazes de escapar de ótimos locais, efetuando uma pesquisa mais robusta do espaço de soluções que permita encontrar soluções de boa qualidade [17].

Uma vez que são métodos aproximados e não determinísticos [18], as metaheurísticas não garantem a obtenção da solução ótima. Outra das características dos métodos metaheurísticos é que não são específicos ou exclusivos para a resolução de um problema em particular, devendo ser entendidos como estratégias gerais que guiam o processo de resolução.

A capacidade dos métodos metaheurísticos escaparem de ótimos locais é obtida através do uso de estratégias de diversificação, que permitem explorar outras zonas do espaço de soluções. Para além das técnicas de diversificação, as metaheurísticas empregam também métodos de intensificação que exploram com maior intensidade zonas do espaço de soluções nas quais se poderão encontrar soluções promissoras.

As metaheuríticas têm outras características para além daquelas já descritas e que definem o modo como sugerem a abordagem para a resolução de um problema. A Tabela 1 [19] apresenta essas características de acordo com as classificações propostas por Stützle [20] e Talbi [21].

Os métodos metaheurísticos inspirados na natureza têm por base elementos e comportamentos do mundo natural. Um exemplo deste tipo de metaheurística são os Algoritmos Genéticos [22] inspirados pela biologia evolutiva. Por outro lado, a Pesquisa Tabu [23] é o exemplo de uma metaheurística não inspirada pela natureza. Os Algoritmos Genéticos são também exemplos de métodos populacionais, ou seja, que recorrem a uma população de soluções, ao invés de focarem a resolução numa única solução.

Autor	Característica
Stützle [20]	• Métodos de trajetória vs. Métodos descontínuos • Métodos com uso de memória vs. Métodos sem uso de memória • Métodos populacionais vs. Métodos de pesquisa em pontos únicos • Uso de uma vs. uso de várias estruturas de vizinhança • Função objetiva Dinâmica vs. Estática • Métodos inspirados na natureza vs. Métodos não inspirados na natureza
Talbi [21]	• Métodos iterativos vs Métodos gulosos • Métodos determinísticos vs. Métodos estocásticos • Métodos com uso de memória vs. Métodos sem uso de memória • Métodos populacionais vs. Métodos de pesquisa em pontos únicos • Métodos inspirados na natureza vs Métodos não inspirados na natureza

Tabela 1 – Classificação das metaheurísticas.

Tal como visto na Tabela 1, Stützle categoriza as metaheurísticas de acordo com o uso de uma função objetivo estática ou dinâmica. Enquanto que muitos procedimentos metaheurísticos mantêm constante a função objetivo utilizada, outros, como a Pesquisa Local Guiada [24] modificam a função objetivo durante a pesquisa de modo a diversificar o espaço de soluções [18]. O uso de várias estruturas de vizinhança como efetuado pela Pesquisa de Vizinhança Variável [25] é outra das formas de se conseguir diversificar a pesquisa no espaço de soluções. Também o uso de memória de curta ou longa duração contribui para que o procedimento de pesquisa explore novas soluções.

Importa também fazer a distinção entre procedimentos determinísticos e estocásticos. Enquanto que os primeiros, partindo de uma mesma solução inicial encontram sempre a mesma solução final, os segundos poderão dar origem a soluções finais distintas em cada execução do algoritmo [19].

No presente estudo quatro metaheurísticas assumem papel de destaque, nomeadamente, a Pesquisa Tabu, o GRASP (*Greedy Randomized Adaptive Search Procedure*), a Pesquisa por Dispersão e a metaheurística RAMP. Dada a importância destas metaheurísticas, apresenta-se de seguida, cada uma delas com maior detalhe.

2.4.1. Pesquisa Tabu

A Pesquisa Tabu (*Tabu Search*) é um procedimento adaptativo introduzido por Glover [23] [27] que atribui a um procedimento de pesquisa local a capacidade de escapar a ótimos locais. A Pesquisa Tabu pode, portanto, ser vista como uma extensão dos métodos clássicos de pesquisa local, combinando-os estes com estratégias de utilização de memória de curta ou longa duração [28].

Como acontece com outras metaheurísticas, o espaço de pesquisa (espaço de soluções que podem ser visitadas durante a pesquisa) tem influência direta na qualidade final da solução conseguida pela Pesquisa Tabu. Por sua vez, o espaço de pesquisa é definido pelo tipo ou tipos de movimentos efetuados pelo procedimento.

O termo "tabu" deriva da existência de mecanismos que permitem evitar que sejam efetuados movimentos cíclicos, ou seja, que o procedimento de pesquisa visite repetidamente as mesmas soluções. Estes mecanismos passam pelo uso de memória de curta duração responsável por armazenar uma pequena quantidade de informação acerca da pesquisa. Frequentemente, essa informação consiste nos movimentos efetuados pelo procedimento e é mantida com recurso a uma estrutura de dados como uma lista circular de tamanho fixo ou variável denominada lista tabu. Os movimentos presentes nessa lista são denominados tabu e não poderão ser repetidos (salvo quando permitidos pelo critério de aspiração que será posteriormente descrito) enquanto constarem dessa lista. Importa referir que geralmente os movimentos apenas constam na lista tabu durante um período limitado que poderá ser definido por um número máximo de iterações ou, se se verificar o caso de uma lista circular, até que seja necessário remover o elemento da lista para que seja inserido um elemento mais recente. No caso de o procedimento de pesquisa local recorrer a vários tipos de movimentos, poderá ser pertinente o uso de várias listas tabu, correspondendo cada lista a um tipo de movimento.

A necessidade do uso de listas tabu com tamanho variável emerge com o facto de uma lista de tamanho fixo ser insuficiente para evitar movimentos fixos aquando da resolução de alguns problemas, como podemos ver na abordagem apresentada por Taillard [29] ao Problema Quadrático de Alocação (*Quadratic Assignment Problem*). Uma lista com tamanho dinâmico poderá também ser uma das formas de acrescentar alguma diversidade à pesquisa.

O uso de diferentes estratégias de pesquisa permite à Pesquisa Tabu explorar diferentes espaços de soluções. Essas estratégias podem ser de intensificação, que conduzem o algoritmo a explorar espaços de soluções considerados promissores recorrendo geralmente

a memória de média duração; ou de diversificação que forçam o algoritmo a explorar espaços de soluções ainda não visitados.

Um dos principais componentes da Pesquisa Tabu, que já foi referido mas ainda não explicado, é o critério de aspiração, que permite ignorar um estado tabu. No caso da informação da lista tabu ser referente aos movimentos efetuados, o critério de aspiração permite que um movimento que conste nessa lista seja executado desde que respeite uma condição pré-definida que muitas vezes consiste em permitir que um movimento tabu seja efetuado caso dele resulte uma solução com um valor da função objetivo que melhora o melhor até então conhecido. A importância deste critério advém da sua capacidade de evitar que uma determinada solução de boa qualidade não seja atingida devido às restrições impostas pela lista tabu.

A pesquisa tabu parte de uma solução inicial que pode ser construída com recurso a uma heurística construtiva e é executada até que seja atingido um critério de paragem. Os critérios de paragem mais comuns são:

- Número máximo de iterações estabelecido;
- Tempo máximo de execução do algoritmo estabelecido;
- Número máximo de iterações sem melhorar a melhor solução até então obtida;
- Valor de função objetivo estabelecido.

A Figura 3 apresenta um algoritmo básico de Pesquisa Tabu, onde S representa uma solução, $V(S)$ representa a vizinhança da solução S, T representa a lista com os movimentos tabu e $S(T)$ represente as soluções de $V(S)$ que não podem ser visitadas já que os movimentos que levariam a elas pertencem a T.

1. Construir uma solução inicial admissível S_0.
2. Inicializar a solução corrente S_c e a melhor solução encontrada S^*, fazendo $S_c = S^* = S_0$.
3. Inicializar o contador de iterações, fazendo $k = 0$.
4. Se $V(S_c) - S(T) \neq \emptyset$ então
5. Incrementar k.
6. Escolher a melhor solução $S_v \in V(S_c) - S(T)$.
7. Se $f(S_v) < f(S^*)$ então
8. $S^* = S_v$.
9. Senão
10. $V(S_c) = v(S_C) - S_v$.
11. Caso ainda não tenha sido satisfeito um critério de paragem, ir para o passo **4**.
12. Terminar.

Figura 3 – Algoritmo básico da Pesquisa Tabu.

2.4.2. GRASP

A metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) é um procedimento iterativo proposto por Feo e Resende [26].

Cada uma das iterações do GRASP é composta por duas fases principais. Na primeira fase é construída uma solução admissível para o problema recorrendo a uma função gulosa adaptativa aleatória. Esta função constrói a solução adicionando elementos à mesma de forma iterativa. A escolha do elemento a adicionar à solução é feita escolhendo aleatoriamente um dos elementos da RCL (*Restricted Candidate List*) composta pelos *n* melhores candidatos, em que *n* é um tamanho definido, que poderá ser modificado ao longo das iterações do GRASP. A avaliação dos melhores candidatos é feita considerando o impacto que a sua inserção na solução tem no valor da função objetivo, sendo escolhidos aqueles cujo impacto é mais favorável. A escolha aleatória de um destes candidatos implica que não é garantido que o elemento selecionado seja o melhor candidato presente na RCL, o que confere ao GRASP um traço probabilístico que lhe permite gerar soluções mais diversificadas. É, portanto, possível concluir que o tamanho da RCL terá impacto na diversidade das soluções geradas. Na Figura 4 [26], é apresentado o algoritmo da fase de construção.

```
1. Solução → ∅.
2. Enquanto a solução não estiver totalmente construída
   fazer:
3.      Construir RCL.
4.      Escolher aleatoriamente um elemento s da RCL.
5.      Solução = Solução U {s}.
6. Terminar.
```

Figura 4 - Algoritmo da fase de construção do GRASP.

A segunda fase do GRASP é responsável por tentar melhorar a solução obtida no passo anterior recorrendo a um algoritmo de pesquisa local. Uma vez que ao longo das iterações do GRASP serão geradas soluções distintas, o algoritmo de pesquisa local terá a oportunidade de explorar diferentes espaços da estrutura de vizinhança, alcançando diferentes ótimos locais.

O procedimento GRASP é executado tantas iterações quantas as necessárias para alcançar um critério de paragem estabelecido (como um número máximo de iterações) e no final é devolvida a melhor das soluções encontradas.

Na figura 5 [26], encontra-se um algoritmo genérico do GRASP, onde $bestS$ representa a melhor solução s encontrada.

```
1. Enquanto o critério de paragem não for satisfeito fazer:
2.      Construir uma solução s.
3.      Aplicar um algoritmo de pesquisa local a s.
4.      Se s for melhor que bestS então
5.           bestS = s.
6. Terminar e devolver bestS.
```

Figura 5 – Algoritmo genérico do procedimento GRASP.

2.4.3. Pesquisa por Dispersão

A Pesquisa por Dispersão (*Scatter Search*) é uma metaheurística evolutiva (baseada em populações de soluções) proposta por Glover [30] que permite a criação de métodos de resolução capazes de derivar novas soluções a partir da combinação de duas ou mais soluções já conhecidas para um problema de otimização. Foi desenhada para operar em

conjuntos de pontos, denominados pontos de referência, obtidos em esforços anteriores do processo de pesquisa [31].

A Pesquisa por Dispersão é composta por uma estrutura de dados que contém as soluções que serão combinadas, a que se dá o nome de conjunto de referência. Por esse motivo, este componente é unanimemente reconhecido como o mais importante desta metaheurística. As restantes soluções encontram-se geralmente armazenadas numa estrutura denominada *pool* de soluções.

Segundo Laguna e Martí [32], a Pesquisa por dispersão é composta por cinco métodos principais:

- **Método de geração de soluções diversificadas**, responsável por gerar uma coleção de soluções diversificadas que estarão na base dos próximos passos do algoritmo. Dadas as suas caraterísticas, o GRASP é frequentemente a abordagem escolhida neste passo da Pesquisa por Dispersão. É importante garantir que da população de soluções resultantes deste método constem soluções de qualidade diversificada.
- **Método de melhoramento**, que tem por objetivo transformar uma solução noutra de melhor qualidade. Este método é geralmente aplicado às soluções obtidas pelo método de geração de soluções e às soluções resultantes do método de combinação de soluções. A Pesquisa Tabu é um exemplo de uma metaheurística que poderá ser implementada neste passo.
- **Método de atualização do conjunto de referência**, cuja responsabilidade é construir e atualizar o conjunto de referência. Este método deverá garantir que constam no conjunto de referência as melhores soluções até então encontradas, sendo que as soluções são classificadas em função da sua qualidade ou diversidade. Se considerarmos que o tamanho b do conjunto de referência é 10 (normalmente b é inferior a 20 e o muitas vezes é definido fazendo $b = tamanho\ da\ pool\ de\ soluções/10$), 5 $(b/2)$ soluções deverão ser de qualidade, enquanto que as restantes 5 deverão ser soluções diversificadas. A diversidade de uma solução pode ser calculada utilizando uma medida de distância dessa solução em relação a todas aquelas que já se encontram no conjunto de referência. A presença de soluções diversificadas no conjunto de referência garante que no processo de combinação das soluções deste conjunto existe um equilíbrio entre a intensificação e diversificação da pesquisa do espaço de soluções.
- **Método de geração de subconjuntos**, que produz subconjuntos de soluções do conjunto de referência. O tamanho dos subconjuntos gerados varia de acordo com a

sofisticação do algoritmo. Numa abordagem mais simples, o tamanho de cada subconjunto será dois.

- **Método de combinação de soluções**, que transforma cada um dos subconjuntos gerados pelo método anterior numa ou mais novas soluções às quais se poderá aplicar o método de melhoramento. As soluções resultantes verão de seguida testada a sua admissibilidade ao conjunto de referência, através do método de atualização do conjunto de referência.

Dos cinco métodos descritos, apenas o método de melhoramento é de carácter opcional, contudo, a sua inclusão costuma levar à obtenção de soluções de melhor qualidade. A Figura 6 [32], apresenta uma representação esquemática do funcionamento da Pesquisa por Dispersão, onde P identifica a *pool* de soluções, $|P|$ o número de soluções em P e $PSize$ o tamanho da *pool* de soluções.

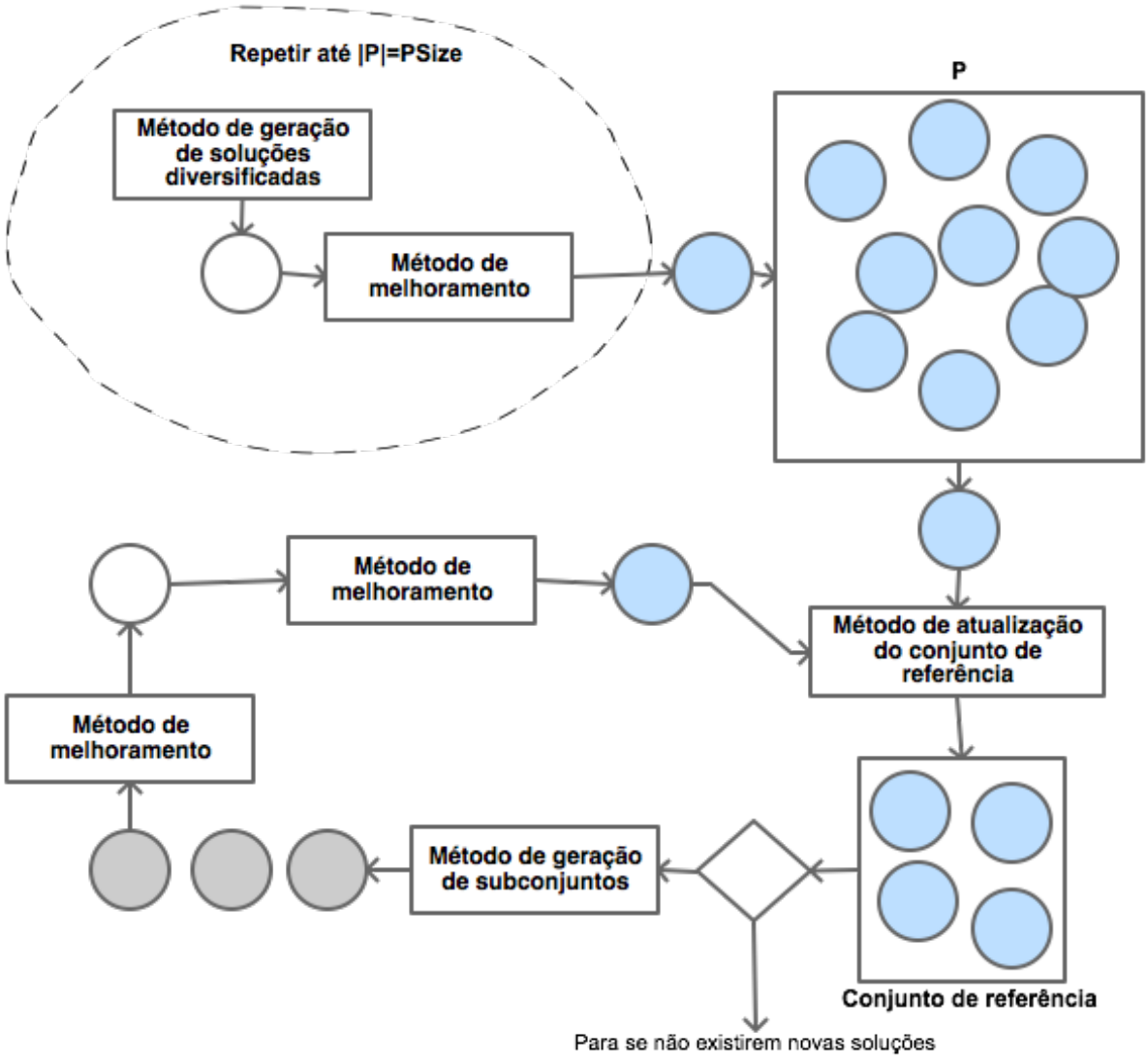

Figura 6 - Esquema do funcionamento da Pesquisa por Dispersão.

No decorrer da pesquisa, é importante garantir que o conjunto de referência não contém soluções duplicadas, pois isso levaria ao deterioramento do funcionamento do algoritmo. A mesma verificação deverá ser feita aquando da execução do método de geração de soluções diversificadas, que deverá popular a *pool* de soluções apenas com soluções únicas.

Abordagens mais sofisticadas da Pesquisa por Dispersão poderão incluir o uso de diferentes métodos de combinação. Também o método de atualização do conjunto de referência poderá ser estático ou dinâmico. Enquanto uma atualização estática apenas tenta adicionar as soluções resultantes do método de combinação ao conjunto de referência após todos os subconjuntos terem sido combinados (o que implica a passagem dessas soluções pela *pool*), a atualização dinâmica tenta adicionar imediatamente a solução gerada ao conjunto de referência. A ideia por detrás deste comportamento passa por tentar combinar o mais rapidamente possível novas soluções com as soluções já existentes no conjunto de referência, contudo esta abordagem poderá impossibilitar algumas combinações.

O algoritmo de Pesquisa por Dispersão é executado até que seja satisfeito um determinado critério de paragem. Tradicionalmente um dos critérios é a execução da Pesquisa por Dispersão até que numa iteração não sejam adicionadas novas soluções ao conjunto de referência.

Na Figura 7 [32], encontra-se um algoritmo básico da Pesquisa por Dispersão, onde:

- P representa a *pool* de soluções;
- $|P|$ representa o número de soluções na *pool*;
- $PSize$ o tamanho da *pool*;
- s representa uma solução;
- $RefSet$ representa o conjunto de referência.

```
1.  Iniciar com P = ∅. Utilizar o método de geração de soluções
    diversificadas para construir uma solução s e aplicar de
    seguida o método de melhoramento. Se s ∉ P fazer P = P ∪ s.
    Repetir este passo até |P| = PSize.
2.  Utilizar o método de atualização do conjunto de referência
    para construir o RefSet. Fazer NewSolutions = TRUE.
3.  Enquanto NewSolutions = TRUE fazer
4.      Gerar NewSubsets com o método de geração de
    subconjuntos. Fazer NewSolutions = FALSE.
5.      Enquanto NewSubsets ≠ ∅ fazer
6.          Selecionar o próximo subconjunto ss.
7.          Invocar o método de combinação de soluções.
8.          Aplicar o método de melhoramento às soluções
    geradas.
9.          Invocar o método de atualização do conjunto de
    referência.
10.         Se o RefSet mudou então
11.             NewSolutions = TRUE.
12.         Remover ss de NewSubsets.
13. Terminar.
```

Figura 7 - Algoritmo básico da Pesquisa por Dispersão.

2.4.4. RAMP

A metaheurística RAMP (*Relaxation Adaptive Memory Programming*), introduzida por Rego [5], tem por base o uso de estruturas de memória que lhe permitem explorar eficientemente a informação obtida no espaço primal (problema original) e no espaço dual (correspondente ao problema obtido através da eliminação de uma ou mais restrições do problema original) de um problema de otimização. Os conceitos de memória adaptativa presentes no RAMP são os mesmos que estão na génese de outras metaheurísticas, das quais se destaca a Pesquisa Tabu.

O RAMP tem por objetivo suprimir uma lacuna nos métodos heurísticos relacionada com a exploração exclusiva de um dos espaços de soluções de um problema, através da criação

de estruturas de memória capazes de compreender informação que não pode ser obtida através da exploração de apenas um destes espaços [5].

Embora o RAMP seja considerado por si só uma metaheurística, pode integrar outras metaheurísticas na resolução do espaço primal e dual do problema. Na resolução do problema primal pode-se recorrer a metaheurísticas como a Pesquisa Tabu ou a Pesquisa por Dispersão, enquanto que na resolução do problema dual se recorre a técnicas de relaxação matemática como a relaxação lagrangeana ou a relaxação por restrições substitutas.

A metaheurística RAMP possuí diferentes níveis de sofisticação: o Dual RAMP e o PD-RAMP. O Dual RAMP consiste na versão mais simples do RAMP e explora com mais intensidade o espaço dual de um problema. O PD-RAMP, versão mais sofisticada do RAMP, surge como uma extensão do Dual-RAMP capaz de intensificar a exploração da relação existente entre o espaço dual e primal. Isto é conseguido através da adição de um componente primal (baseado num método evolutivo) capaz de utilizar mais intensamente a memória adaptativa. O uso de métodos evolutivos está na base da existência de um conjunto de referência composto por soluções que resultam quer do espaço dual, quer do espaço primal do problema.

A Figura 8 [5], apresenta um esquema do funcionamento do PD-RAMP.

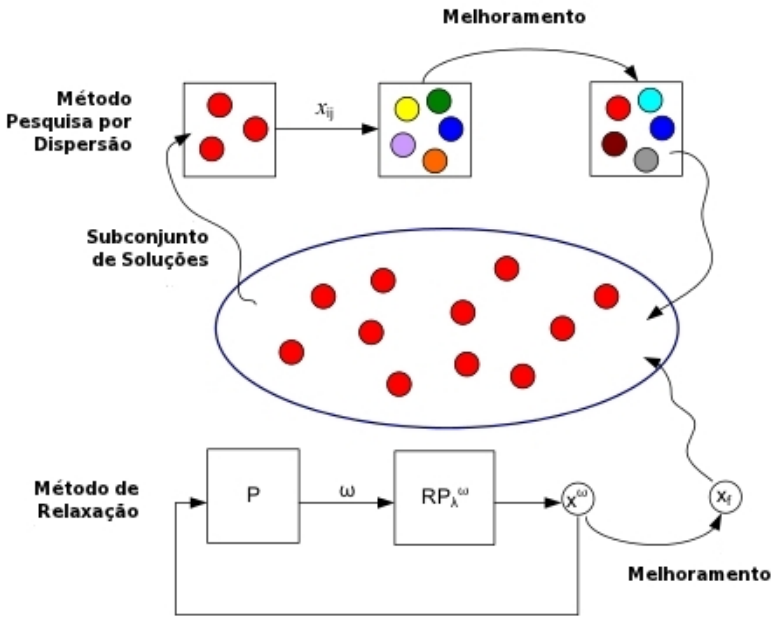

Figura 8 - Esquema do funcionamento do PD-RAMP.

Tal como descrito por Rego [5], o PD-RAMP inicia-se com a construção do conjunto de referência com soluções oriundas da aplicação de uma técnica de relaxação matemática, dotando o conjunto de referência de informação disponibilizada pelo espaço dual do

problema. Estas soluções serão de seguida combinadas seguindo as orientações de um método evolutivo como a Pesquisa por Dispersão, seguindo-se a atualização do conjunto de referência que permitirá incluir informação obtida no espaço primal do problema. A estas soluções poderá ser aplicado um algoritmo de melhoramento na tentativa de encontrar soluções de melhor qualidade. O conjunto de referência deverá ser mantido de forma a conter apenas soluções únicas, sendo que parte delas deverá ser de qualidade e as restantes serão incluídas em função do seu grau de diversificação. O PD-RAMP continua a sua execução, alternando entre o espaço dual e primal do problema, até que seja satisfeito um dos critérios de paragem estabelecidos, como o número máximo de iterações.

Apesar de ser considerada uma abordagem recente, o RAMP já demonstrou que é capaz de obter soluções de boa qualidade em vários problemas de otimização combinatória, entre os quais o Problema de Ordenação Linear (Linear Ordering Problem) [6], o Problema de Localização de Instalações sem Restrições de Capacidade (Uncapacitated Facility Location Problem) [6], o Problema da Árvore de Expansão Mínima com Restrições de Capacidade (Capacitated Minimum Spanning Tree) [7], o Problema de Localização de Hubs com Afetação Múltipla e sem Restrições (*Uncapacitated Multiple Allocation Hub Location Problem*) [8], o Problema de Localização de Instalações com e sem restrições de capacidade (*Capacitated/Uncapacitated Facility Location Problem*) [9] e o Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor (*Single Source Capacitated Facility Location Problem*) [10].

2.5. Técnicas de Relaxação Matemática

A aplicação de técnicas de relaxação matemática tem por objetivo simplificar a resolução de um determinado problema através da eliminação de uma ou mais restrições. O problema resultante da aplicação de uma técnica de relaxação matemática ao problema original recebe a denominação de problema dual e a sua resolução permite obter um limite superior ou inferior para o valor da função objetivo do problema original, dependendo de se tratar de um problema de minimização ou maximização, respetivamente.

De seguida serão descritas algumas técnicas de relaxação matemática, tendo por base os trabalhos de Rego [5] e Gamboa [6]. De modo a facilitar a compreensão de cada uma das técnicas será utilizado como exemplo um problema genérico de programação linear P definido do seguinte modo:

$$P = \text{Min}\, cx \qquad (1)$$

$s.a$:

$$Ax \leq b \quad (2)$$

$$Dx \leq e \quad (3)$$

$$x \in \{0,1\} \quad (4)$$

Em que se considera que é conjunto de restrições (2) que torna o problema de difícil resolução.

2.5.1. Relaxação Lagrangeana

A relaxação lagrangeana do problema P é obtida através da eliminação (dualização) do conjunto de restrições (2) identificadas como aquelas que dificultam a resolução do problema. O problema LP^{λ} que resulta deste processo é definido da seguinte forma:

$$LP^{\lambda} = Min\ cx + \lambda(Ax - b) \quad (5)$$

$s.a$:

$$Dx \leq e \quad (6)$$

$$x \in \{0,1\} \quad (7)$$

Na formulação apresentada, λ identifica um vetor de multiplicadores não negativos que penalizam a dualização do conjunto de restrições (2). A resolução do problema dual tem por objetivo encontrar um vetor λ que maximize o valor de $v(LP^{\lambda})$. Esse valor corresponde a um limite inferior de $v(P)$ devendo por isso ser igual ou inferior ao valor da função objetivo da solução ótima de P. Note-se que caso P fosse um problema de maximização o valor obtido através da resolução da sua forma dual seria um limite superior para o problema.

2.5.2. Relaxação por Restrições Substitutas

A relaxação por restrições substitutas do problema P é conseguida tornando o conjunto de restrições (2) numa única restrição multiplicada por um vetor de pesos w, o que origina o problema SP^{w} definido do seguinte modo:

$$SP^{w} = Min\ cx \quad (8)$$

$s.a$:

$$w(Ax - b) \leq 0 \quad (9)$$

$$Dx \leq e \quad (10)$$

$$x \in \{0,1\} \quad (11)$$

Originalmente introduzida por Glover [33], a relaxação por restrições substitutas permite, pelo menos em termos teóricos, a obtenção de resultados de qualidade superior àqueles concebidos pela aplicação de uma relaxação lagrangeana.

Tal como acontece com a relaxação lagrangeana, o valor $v(SP^w)$ que se obtém através da resolução do problema SP^w não pode exceder o valor da solução ótima do problema primal.

Uma vez que as restrições substitutas incorporam as restrições do problema original, uma solução ótima para o problema SP^w é considerada admissível para P sendo, portanto, automaticamente considerada ótima também para o problema primal.

2.5.3. Otimização por subgradiente

A otimização por subgradiente é uma técnica de resolução do problema dual lagrangeano que já demonstrou ser igualmente capaz de calcular com eficácia os pesos aplicados às restrições substitutas [6].

Assumindo-se a resolução do dual lagrangeano, a otimização por subgradiente atualiza em cada iteração os multiplicadores λ. Esse cálculo pode ser efetuado através da seguinte expressão:

$$\lambda^{k+1} = \lambda^k + \theta^k(Ax^k - b) \qquad (12)$$

Onde:

- k identifica a iteração corrente;
- $v(D^\lambda)$ é um limite superior do valor ótimo do problema dual;
- π^k é um parâmetro cujo valor se situa entre 0 e 2, a que se dá o nome de *agility*;
- Δ^ké um escalar, conhecido por passo, que pode ser calculado recorrendo à seguinte fórmula:

$$\Delta^k = \frac{\pi^k[v(D^\lambda) - v(LP^\lambda)]}{||Ax^k - b||^2} \qquad (13)$$

Para melhor compreensão do funcionamento da otimização por subgradiente sugere-se a leitura dos trabalhos de Beasley [34] e Fisher [35].

2.5.4. Relaxação Paramétrica Cruzada

A relaxação paramétrica cruzada [5] pode ser entendida como uma combinação entre a relaxação lagrangeana e a relaxação por restrições substitutas. Esta técnica de relaxação matemática é uma generalização daquela proposta por Narciso e Lorena [36] e foi desenhada para a obtenção eficiente de boas soluções para um problema dual. O problema $L_\lambda SP^w$, obtido pela relaxação paramétrica cruzada do problema P é definido da seguinte forma:

$$L_\lambda SP^w = Min\ cx + \lambda w(Ax - b) \quad (14)$$

$s.a$:

$$Dx \leq e \quad (15)$$

$$x \in \{0,1\} \quad (16)$$

Tal como descrito por Rego [5], o vetor w (vetor de pesos associados às restrições substitutas) é calculado recorrendo ao método do subgradiente, sendo posteriormente utilizado como parâmetro na aplicação da otimização por subgradiente levada a cabo na relaxação lagrangeana aplicada ao problema resultante da relaxação por restrições substitutas.

A solução obtida através da resolução do problema dual dará origem a um limite inferior enquanto que a sua projeção (reposição de admissibilidade) para o espaço de soluções do problema primal resultará num novo limite superior para o problema.

3. Algoritmos para a resolução do problema *P-Median*

A expressão “problema de localização” diz respeito à modelação, formulação e resolução de um conjunto de problemas em que se procura definir a localização de algo num dado espaço. Os problemas de localização de instalações têm por objetivo escolher a localização de uma ou mais instalações de modo a minimizar o custo de satisfação das necessidades de um conjunto de clientes, respeitando possíveis restrições existentes [4], como a quantidade máxima de recursos que um armazém poderá fornecer aos clientes.

A origem do primeiro problema de localização conhecido remonta ao século XVII. Este problema consistia em escolher um ponto entre três pontos no plano (formando um triângulo) de modo a que fosse minimizado o somatório da distância dos restantes pontos ao ponto escolhido [37]. Acredita-se que o problema tenha sido introduzido por Pierre Fermat, matemático francês, contudo não existe consenso nesta creditação [38]. Melzak [39] acredita que o problema apresentado foi proposto e resolvido pelo matemático italiano Battista Cavalieri, proposto novamente por Fermat e mais tarde resolvido pelo cientista italiano Evaristo Torricelli. Já Zacharias [40] acredita que foi este último o responsável pela proposta e resolução do problema.

Desde a introdução do primeiro problema de localização que a comunidade científica se dedicou ao estudo destes problemas, já que viam neles vasta aplicabilidade em contextos reais. Já no século XX, Alfred Weber encontrou aplicação prática para o problema já descrito no ramo industrial e desenvolveu o problema com a adição de pesos, simulando a procura de um cliente que deveria ser satisfeita [11]. O objetivo deste problema era escolher um desses três pontos ((x^*, y^*)) para colocar uma instalação. A sua formulação é a seguinte [38]:

$$min_{x,y}\left\{W(x,y) = \sum_{i=1}^{n} w_i d_i(x,y)\right\} \tag{17}$$

Em que:

- w_i representa o peso associado ao ponto i;
- $d_i(x, y)$ representa a distância euclidiana entre os pontos (x, y) e (a_i, b_i) que pode ser calculada com recurso à seguinte expressão:

$$\sqrt{(x - a_i)^2 + (y - b_i)^2} \tag{18}$$

O trabalho de Weber chamou ainda mais a atenção dos investigadores, que fizeram com que o problema se desenvolve-se para versões mais sofisticadas onde se verifica a existência de mais do que três pontos e três instalações.

De um modo geral, os problemas de localização são compostos por quatro componentes principais: clientes (que se assume já estarem instalados); instalações para as quais se pretende definir a localização; possíveis localizações para as instalações; e uma métrica que permita medir a distância ou tempo entre os clientes e as instalações [4]. Com base nestes componentes, os modelos matemáticos para os problemas de localização de instalações são formulados tendo em conta a resposta às seguintes questões [41]:

- Quantas instalações devem ser consideradas?
- Onde deve ser colocada cada instalação?
- Qual o tamanho/capacidade/abrangência de cada instalação?
- De que forma deverão os recursos de cada instalação serem alocados aos clientes?

A resposta a estas questões depende do contexto do problema e dos seus objetivos e restrições.

Os problemas de localização de instalações são considerados NP-Difíceis [4], o que desaconselha o uso de métodos exatos. A complexidade da sua resolução advém das restrições associadas a estes problemas que variam de acordo com o problema em questão e com as suas caraterísticas. Daskin [41] apresenta uma classificação dos problemas de localização de acordo com as suas caraterísticas, como pode ser observado na tabela seguinte:

Característica	Descrição
Problemas planares, discretos ou em rede	Num problema planar (ou contínuo) é possível colocar as instalações em qualquer ponto do plano, enquanto que um problema em rede é representado por um grafo, sendo apenas possível localizar as instalações nos nós desse grafo. Um problema discreto acrescenta aos problemas em rede distâncias arbitrárias entre os nós.

Problemas baseados em árvores ou grafos gerais	Se o grafo que representa o problema consentir a existência de um caminho a partir de cada um dos nós para todos os restantes, o problema é considerado baseado em árvore, caso contrário diz-se que o problema é baseado num grafo geral.
Métrica do problema	A métrica de um problema identifica o modo como é medida a distância entre dois pares de pontos no plano do problema. Dois exemplos de métricas recorrentes são a distância euclidiana ou a métrica de *Manhattan*.
Número de instalações	Total de instalações para as quais se pretende definir uma localização.
Problemas de localização estática ou dinâmica	Nos problemas de localização estáticos, os parâmetros do problema não variam ao longo do tempo (por exemplo, a quantidade de recursos que um cliente necessita de ver satisfeita não é alterada), situação inversa ao que se verifica nos problemas de localização dinâmica.
Problemas determinísticos ou probabilísticos	Se os parâmetros ou *inputs* do problema forem um dado adquirido o problema será considerado determinístico. Caso existam *inputs* incertos será considerado probabilístico. Por exemplo, um cliente poderá ou não necessitar de ser alocado a uma instalação.
Problemas com um único produto ou com vários produtos	A procura de um cliente poderá incidir sobre um ou vários produtos.
Problemas enquadrados no sector público ou privado	Enquanto que nos problemas enquadrados no sector público é possível medir os custos (incluindo investimentos e benefícios) usando uma unidade monetária, nos problemas do sector privado existe outro tipo de componentes que não pode ser medida segundo essa métrica, como por exemplo, custos relacionados com o impacto ambiental.
Problemas com um ou vários objetivos	Um problema poderá ter vários objetivos. Por exemplo, um problema poderá ter apenas como objetivo alocar cada cliente à instalação mais próxima, enquanto que um outro problema, para além deste objetivo, poderá também

	procurar que todas as instalações têm pelo menos um cliente.
Problemas com necessidades elásticas ou estáticas	Na maioria dos problemas de localização a procura por parte de um cliente não é elástica, contudo existem situações em que isso não se verifica já que, no mundo real, a procura de um cliente varia muitas vezes em função do nível de serviço e das alternativas à sua disposição.
Capacidade das instalações	As instalações poderão ou não ter uma capacidade ilimitada. A título de exemplo, num dado problemas as instalações poderão satisfazer as necessidades de tantos clientes quantos existirem, enquanto que noutro problema, uma instalação apenas poderá satisfazer a procura de um cliente.
Tipo de alocação dos clientes	Um cliente poderá ser afeto a apenas uma ou a várias instalações, de acordo com o objetivo do problema. Por exemplo, um problema pode ter como objetivo alocar cada cliente à instalação mais próxima enquanto que outro procura alocar cada cliente a tantas instalações quantas as necessárias para satisfazer as necessidades desse cliente.
Problemas hierárquicos ou de nível único	Em alguns problemas existe uma hierarquia de instalações enquanto que noutros um cliente poderá ver as suas necessidades diretamente satisfeitas por qualquer instalação.
Alocação de instalações desejadas ou não desejadas	Se na maioria dos casos o objetivo de um problema de localização é colocar instalações o mais próximo possível dos clientes, existem situações em que o inverso se verifica (como a localização de um centro de lixo tóxico).

Tabela 2 - Caraterização dos problemas de localização de instalações.

Como já referido, os problemas de localização de instalações encontram aplicabilidade em vários contextos, como são disso exemplos: a localização de armazéns de uma cadeia de abastecimento de modo a minimizar o tempo médio de chegada do produto ao mercado; a localização de depósitos de material nocivo de modo a que fique o mais longe possível da população; ou a localização de terminais multibanco [4].

O presente estudo tem como foco o problema *P-Median*. Assim sendo será de seguida apresentada a definição e formulação deste problema, assim como os melhores algoritmos conhecidos para a sua resolução.

O *P-Median* é um problema de otimização combinatória que se enquadra no grupo dos problemas de localização de instalações, sendo mesmo considerado o problema mais representativo e estudado deste grupo [4]. Introduzido por Hakimi [11], é um problema considerado NP-Difícil que pode ser definido do seguinte modo [12]. Considere-se um grafo completo, pesado e não direcionado $G = (V, A)$, em que V representa um conjunto de vértices e A um conjunto de arestas. Considere-se também que $c_{ij} \in \Re, \forall (i,j) \in A$ simboliza o custo de atribuir a procura de um cliente $j \in V$ a uma mediana (instalação) i. Dado um inteiro positivo $0 < p \leq |V|$, o objetivo do problema *P-Median* passa por selecionar p medianas do conjunto de vértices V de modo a que o custo total de associar cada cliente à respetiva mediana mais próxima seja minimizado [42]. Trata-se, portanto, de um problema de minimização.

O problema *P-Median* poderá ser formulado da seguinte forma [65]:

$$Min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{19}$$

$s.a$:

$$\sum_{i:(i,j)\in A} x_{ij} + x_{jj} = 1 \; \forall j \in V \tag{20}$$

$$\sum_{i\in V} x_{ii} = p \tag{21}$$

$$x_{ij} \leq x_{ii} \; \forall (i,j) \in A \tag{22}$$

$$x_{ii} \in \{0,1\} \forall i \in V \tag{23}$$

$$x_{ij} \geq 0 \forall (i,j) \in A \tag{24}$$

Onde x_{ii} são variáveis binárias associadas a cada vértice $i \in V$ que identificam se i é uma mediana, e x_{ij} são variáveis binárias associadas a cada aresta $(i,j) \in A$ que identificam se o vértice i se encontra conectado ao vértice j.

A função objetivo (19) minimiza o custo de atribuir cada nó j a uma mediana i. O conjunto de restrições (20) especifica que j ou se encontra alocado a uma mediana ou é uma mediana e a restrição (21) garante que p vértices são escolhidos como medianas. O conjunto de restrições (22) assegura que um vértice apenas poderá estar ligado a outro se este último for uma mediana. Por fim, os conjuntos de restrições (23) e (24) garantem a

integralidade das variáveis x_{ii} e x_{ij}. Uma solução para uma instância do problema *P-Median* apenas será considerada admissível se respeitar todas estas restrições, o que implica que todos os vértices deverão ser alocados a uma e só uma mediana, ou serem eles próprios medianas.

A Figura 9, apresenta uma esquematização dos passos para a resolução de uma instância de um problema *P-Median*. Em (a) encontra-se uma representação de uma micro instância com 10 vértices. O objetivo desta instância é escolher a localização de duas instalações (medianas), situação que se encontra ilustrada em (b), onde foram escolhidos para medianas os vértices 3 e 6. De seguida, cada um dos restantes vértices é alocado à mediana que se encontra mais próxima, como pode ser observado em (c).

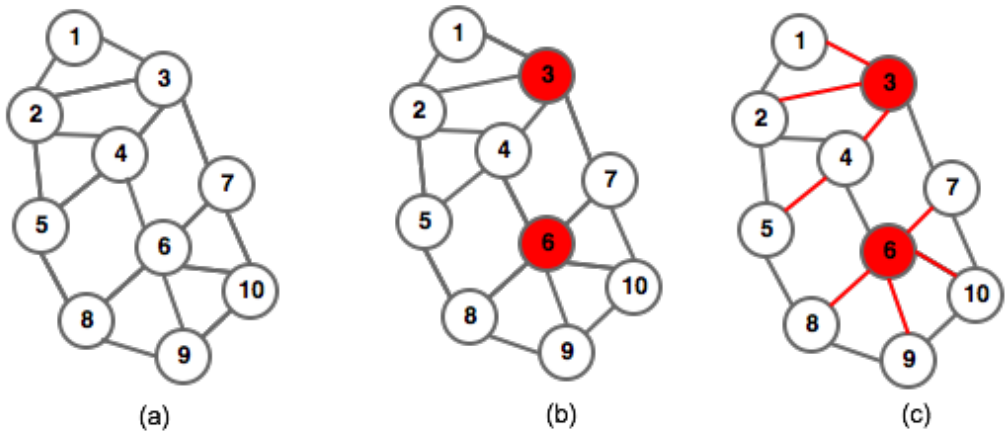

Figura 9 - Esquematização da resolução de um problema *P-Median.*

Atendendo à classificação proposta por Daskin [41], o problema *P-Median* é um problema de otimização combinatória que pode ser classificado como sendo discreto, estático, determinístico, sem restrições de capacidade e de nível único. A alocação é feita à instalação mediana mais próxima e o objetivo é a escolha da localização para as instalações desejadas.

Desde a sua introdução por Hakimi na década de sessenta que o *P-Median* se tornou num dos problemas de localização de instalações mais estudado, com um grande crescimento principalmente na última década [43], surgindo com naturalidade diversas propostas de resolução. As três primeiras heurísticas primárias a serem testadas na resolução deste problema foram a heurística *Greedy* [44], a *Alternate* [45] e a *Vertex Substitution* [46]. Estas três heurísticas viriam a ser combinadas resultando na aplicação de outros métodos de resolução. Apresentam-se de seguida as abordagens mais relevantes propostas para a resolução do problema *P-Median*.

Kuehn e Hamburger [44], introduziram uma heurística construtiva gulosa (*Greedy*) para a localização de armazéns que viria a ser utilizada para a resolução do *P-Median*.

Galvão [47], apresenta uma relaxação linear do problema cujas soluções duais ótimas são utilizadas como limites num algoritmo *branch-and-bound*. O método de resolução proposto por Galvão tem por base o algoritmo *dual ascent* (DUALOC) descrito por Erlenkotter [48] para o UFLP (*Uncapacitated Facility Location Problem*).

Maranzana [45], apresentou um algoritmo de pesquisa local conhecido por *Alternate.* O algoritmo começa por escolher arbitrariamente um conjunto de p vértices que serão inicialmente as medianas. De seguida, os restantes vértices são divididos em células segundo a sua proximidade. No final de cada iteração existirá uma coleção de p vértices que reduzem, ou mantêm, o valor do somatório total das distâncias de cada vértice à mediana mais próxima.

Teitz e Bart [46], introduziram a pesquisa local clássica para o problema *P-Median*: a heurística *Interchange*. O algoritmo escolhe inicialmente p medianas e, em cada iteração, efetua a melhor troca entre um vértice não mediana e outro vértice mediana. Apenas são admitidos movimentos que melhorem o valor da solução corrente. Este método viria a ser melhorado por Whitaker [49] e, mais tarde Resende e Werneck [50] apresentaram uma versão mais sofisticada denominada *Swap Local Search*.

Beasley [51], apresentou uma *framework* para o desenvolvimento de heurísticas robustas com base na relaxação lagrangeana e optimização por subgradiente para a resolução de diversos problemas de localização, entre os quais o *P-Median*. Anteriormente, Narula e Ogbu [52], haviam proposto aquela que é considerada a heurística lagrangeana clássica para o *P-Median,* capaz de obter limites inferiores de boa qualidade. A relaxação proposta passa pela eliminação do conjunto de restrições (20) que implica que o vértice esteja ligado a uma e só uma mediana, ou seja ele próprio uma mediana.

Mladenovic *et al.* [53], propõem uma heurística chamada *Chain-interchange* baseada na Pesquisa Tabu. Este método é visto como uma extensão do algoritmo *Interchange*. Outros autores aplicaram a Pesquisa Tabu na resolução do *P-Median,* tais como, Rolland *et al.* [54] e Goncharov e Kochtov [55].

Hansen e Mladenovic [56], apresentam um algoritmo baseado na Pesquisa de Vizinhança Variável, que explora vizinhanças cada vez mais distantes para evitar a armadilha do ótimo local.

Hosage e Goodchild [57], aplicaram pela primeira vez Algoritmos Genéticos para a resolução do problema *P-Median*, admitindo que a abordagem proposta tem tendência para ficar presa em ótimos locais. Outros autores aplicaram Algoritmos Genéticos a este problema, como Moreno-Perez *et al.* [58] e mais recentemente Pullan [59] que apresenta uma metaheurística híbrida com base num algoritmo genético capaz de conseguir resultados de excelente qualidade.

García-López *et al.* [60], apresentam uma paralelização de um algoritmo baseado na Pesquisa por Dispersão.

Chiyoshi e Galvão [61], desenvolveram uma abordagem que combina a heurística *Interchange* com a metaheurística Arrefecimento Simulado. Levanova e Loresh [62] exploram a implementação de um algoritmo baseado em Arrefecimento Simulado e em Colónia de Formigas.

Resende e Werneck [63], aplicam o procedimento GRASP na resolução do problema *P-Median*. A abordagem proposta incluí a combinação de soluções com recurso ao *Path-relinking*, visto como uma especialização da Pesquisa por Dispersão.

Recentemente, Ren *et al.* [64], apresentaram um algoritmo guiado pela procura de variáveis gordas (variáveis ausentes de todas as soluções ótimas) e variáveis de coluna (variáveis presentes em todas as soluções ótimas). Este algoritmo detém, atualmente, os melhores resultados da literatura.

Para uma análise mais completa das abordagens ao problema *P-Median* existentes na literatura remete-se para a leitura dos trabalhos de Reese [43] e Mladenovic *et al.* [65].

4. Algoritmos RAMP para o problema *P-Median*

Neste capítulo são apresentados e descritos os novos algoritmos RAMP (Dual RAMP e PD-RAMP) resultantes deste estudo. Para cada um dos algoritmos é descrito cada um dos seus componentes e o modo como é resolvido o espaço primal e espaço dual do problema *P-Median*.

4.1. Algoritmo Dual RAMP

O algoritmo Dual RAMP implementado, correspondente à abordagem menos sofisticada da metaheurística RAMP, foca-se na resolução do espaço dual do problema *P-Median*, que resulta da aplicação de uma relaxação lagrangeana ao mesmo.

O algoritmo começa por construir uma solução inicial admissível no espaço de soluções primal com recurso a uma heurística construtiva. O valor da função objetivo da solução construída corresponde ao limite superior (Z_{ub}) inicial. De seguida, é iniciada a fase iterativa do algoritmo, na qual, em cada iteração, é resolvido o dual lagrangeano seguido da projeção da solução dual obtida para o espaço primal de soluções para que lhe seja aplicado um procedimento de pesquisa local.

A Figura 10 esquematiza o funcionamento geral do algoritmo Dual RAMP.

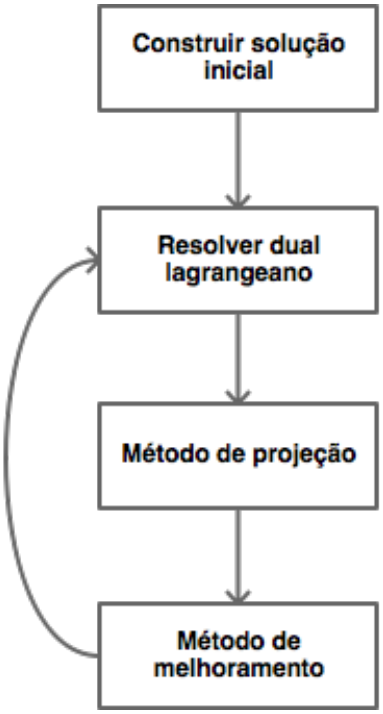

Figura 10 – Funcionamendo geral do algoritmo Dual RAMP.

O cálculo dos multiplicadores de *Lagrange* utilizados para a resolução do problema dual recorre ao método da otimização por subgradiente. Da resolução do problema dual resulta um limite inferior (Z_{λ}^{*}) que será utilizado nos passos subsequentes do algoritmo. A atualização do limite superior, utilizado no método do subgradiente, é feito sempre que uma melhor solução seja obtida no espaço primal. A qualidade do limite inferior e superior obtidos tem impacto direto na qualidade das soluções conseguidas pelo algoritmo Dual RAMP, bem como na velocidade com que o algoritmo converge para essas soluções. É importante relembrar que, no contexto de um problema de minimização, o valor do limite superior não pode ser inferior ao valor da solução ótima do problema ao inverso do que acontece no que respeita ao limite inferior. Podemos concluir que no limite, o valor do limite inferior é igual ao do limite superior. Neste caso considera-se que foi encontrada a solução ótima do problema e o algoritmo Dual RAMP termina. Esta situação encontra-se representa na Figura 11.

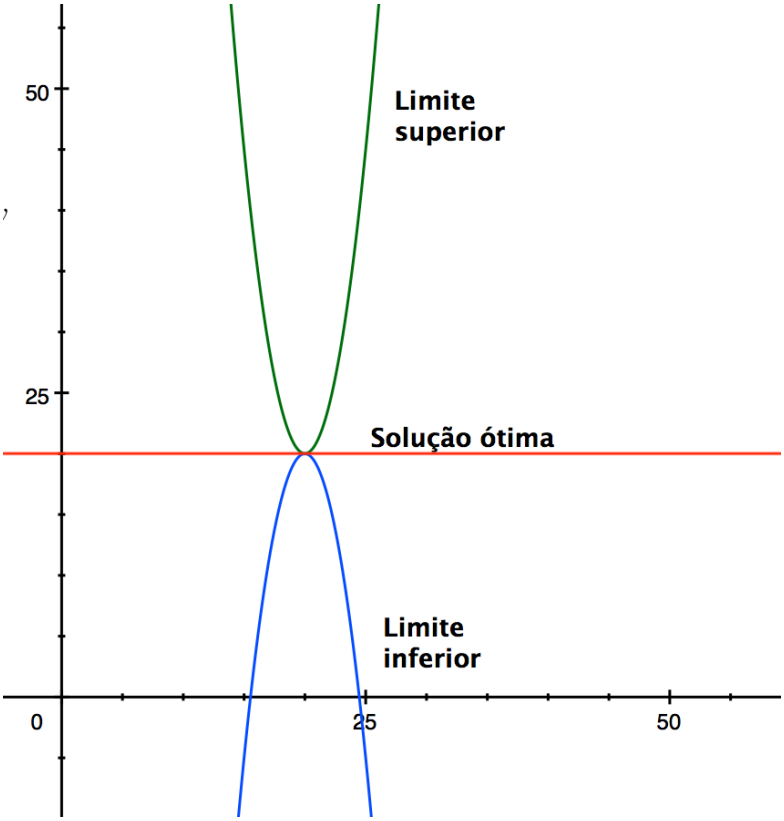

Figura 11 – Representação gráfica do limite inferior e superior.

Para além do critério de paragem supracitado, o algoritmo termina a sua execução assim que atinge um número máximo de iterações estabelecido ou quando o valor do parâmetro *agility* (π) assumir um valor inferior ao pretendido.

4.1.1. Método primal

A resolução do problema primal é efetuada em duas fases distintas do algoritmo. A primeira fase consiste na construção (com recurso a uma heurística construtiva) de uma solução inicial cujo valor da função objetivo será utilizado como limite superior Z_{ub}. Este procedimento apenas é efetuado no inicio do algoritmo. Por sua vez, a segunda fase diz respeito à aplicação de um algoritmo de melhoramento que, partindo da solução obtida pela resolução do problema dual e após a projeção desta para o espaço primal de soluções, procura a obtenção de uma solução de melhor qualidade que eventualmente será o novo limite superior.

De seguida são descritas estas duas fases com maior detalhe.

Construção da solução inicial

Para a construção da solução inicial foram implementadas e testadas as diversas heurísticas construtivas que são agora apresentadas. Nas descrições seguintes, considera-se que:

- V representa o conjunto de todos os vértices v (localizações) da instância;
- S representa um conjunto composto pelos vértices escolhidos como medianas;
- $|S|$ representa o tamanho do conjunto S;
- p representa o número de instalações medianas do problema;
- $\emptyset$ indica que o conjunto não contém elementos;
- $bestV$ representa o vértice que oferece a maior redução no valor da função objetivo da solução corrente, quando a ela adicionado.

Greedy

A heurística *greedy* constrói uma solução adicionando a S, em cada iteração, o vértice que maximize a redução incremental do valor corrente da função objetivo, até que p instalações tenham sido escolhidas [49]. A Figura 12 apresenta um algoritmo *greedy*:

```
1. Iniciar com S → ∅.
2. Enquanto |S| < p fazer
3.     Encontrar o vértice bestV ∈ V, best V ∉ S que ofereça a
   maior redução no valor corrente da função objetivo.
4.     S = S ∪ bestV
5. Terminar.
```

Figura 12 - Algoritmo da heurística Greedy.

First P

Como o seu nome indica, o *First P* constrói a solução inicial S adicionando a este conjunto os p primeiros vértices do conjunto V (ver Figura 13).

```
1. Iniciar com S → ∅, k = 0.
2. Enquanto k < p fazer
3.     S = S ∪ v^k.
4.     Incrementar k.
5. Terminar.
```

Figura 13 – Algoritmo da heurística First P.

Linear

A heurística *Linear* começa por definir um valor para o parâmetro h (passo) no intervalo $0 \leq h \leq p$. Com base nesse valor, S é composto pelos vértices $v^0, v^h, v^{2h}, \ldots, v^{nh}$. O vértice escolhido deverá pertencer ao conjunto V e não deverá constar em duplicado em S (ver Figura 14).

```
1. Iniciar com S → ∅, k = 0.
2. Definir um valor para h no intervalo 0 ≤ h ≤ p.
3. Enquanto |S| < p fazer
4.     Se k + h ≤ |V| então
5.         k = k + h.
6.     Senão
7.         k = h − (|V| − k).
8.     S = S ∪ v^k.
9. Terminar.
```

Figura 14 - Algoritmo da heurística Linear.

Sample

Este método apresenta semelhanças ao *Greedy* mas ao contrário deste último, seleciona o melhor vértice de entre aqueles de um conjunto Q mais pequeno que V, formado por vértices escolhidos aleatoriamente de V. O algoritmo desta heurística pode ser observado na Figura 15.

```
1. Iniciar com S → ∅.
2. Definir o tamanho q do conjunto Q no intervalo 0 < q < |V|.
3. Enquanto |S| < p fazer
4.     Enquanto |Q| < q fazer
5.         Escolher aleatoriamente um vértice v ∈ V, v ∉ S.
6.         Q = Q ∪ v.
7.     Encontrar o vértice bestV ∈ Q que mais reduz o valor
   corrente da função objetivo.
8.     S = S ∪ bestV.
9. Terminar.
```

Figura 15 - Algoritmo da heurística Sample.

Random

Segundo este método p vértices pertencentes a V são escolhidos aleatoriamente, um em cada iteração (ver Figura 16).

```
1. Iniciar com S → ∅.
2. Enquanto |S| < p fazer
3.     Escolher aleatoriamente um vértice v ∈ V, v ∉ S.
4.     S = S ∪ v
5. Terminar.
```

Figura 16 - Algoritmo da heurística Random.

RGreedy

A heurística *RGreedy* (*Randomized Greedy*) consiste numa variação da *Greedy*, mas ao contrário desta última, que seleciona o vértice $v \in V$ que maximiza a redução incremental do valor corrente da função objetivo, a heurística *RGreedy* constrói em cada iteração uma lista RCL (*Restricted Candidate List)* de tamanho α com os melhores vértices, seguindo-se a escolha aleatória de um vértice v desta lista. O algoritmo desta heurística é apresentado na Figura 17.

```
1. Iniciar com S → ∅.
2. Enquanto |S| < p fazer
3.     Construir RCL com os melhores vértices v ∈ V, v ∉ S.
4.     Escolher aleatoriamente um vértice v ∈ RCL.
5.     S = S ∪ v
6. Terminar.
```

Figura 17 - Algoritmo da heurística RGreedy.

Das heurísticas construtivas testadas aquela que apresenta melhores resultados é a *Greedy* (ver Tabela 3), que por esse motivo foi a heurística escolhida para construir a solução inicial do algoritmo Dual RAMP. A justificação para o facto de a *Greedy* produzir melhores soluções, está ligada ao seu carácter determinístico, associado ao uso de informações do problema, como a distância entre vértices. Pela análise da Tabela 3, podemos ainda verificar que as heurísticas que têm grande nível de aleatoriedade apresentam piores resultados. Exemplo disso é a heurística *Random,* que faz as escolhas dos vértices de forma completamente aleatória, ignorando propriedades do problema *P-Median* e que por isso tende a gerar soluções de má qualidade. Embora não seja a heurística mais rápida, a qualidade das soluções obtidas pelo *Greedy* torna-se compensatória já que o desempenho do algoritmo Dual RAMP beneficia da inicialização com um bom limite superior.

Na tabela que se segue, são apresentados os resultados globais obtidos (tempo computacional médio e desvio médio) pelas várias heurísticas construtivas nas instâncias ORLIB (no capítulo referente à descrição dos resultados computacionais serão descritas as instâncias usadas e a fórmula de cálculo do desvio médio):

Heurística	Tempo (s)	Desvio médio (%)
Greedy	0,418	2,008
First P	0,322	75,547
Linear	0,322	60,448
Sample	0,354	61,867
Random	0,325	65,793
RGreedy	0,429	37,203

Tabela 3 - Resultados obtidos pelas heurísticas construtivas nas instâncias ORLIB.

Método de melhoramento

O método de melhoramento utilizado no algoritmo Dual RAMP tem por base o *Swap-Based Local Search* descrito por Resende e Werneck [50]. Este procedimento é apresentado como sendo uma implementação capaz de obter resultados com maior rapidez comparativamente ao algoritmo proposto por Whitaker [49], necessitando em contrapartida de uso extra de memória. Ambas as abordagens têm a sua origem na heurística *Vertex Substitution*, também conhecida por *Interchange*, introduzida por Teitz e Bart em [46], que se tornou a pesquisa local clássica para a resolução do problema *P-Median*.

O funcionamento do *Swap-Based Local Search* assenta na troca de instalações, definindo um vértice como mediana em detrimento de outro que o é. Em termos mais concretos, o procedimento de pesquisa local encontra, iterativamente, para cada instalação $f_i \notin S$ (em que S representa uma solução, ou seja, o conjunto de vértices correspondentes a instalações medianas) a instalação $f_r \in S$ (se existir) que mais melhoraria o valor corrente da função objetivo caso f_i e f_r fossem trocadas, ou seja, se f_i se tornasse uma mediana e f_r deixasse de o ser [50]. Em cada iteração do método apenas será efetuado o movimento de troca entre o par (f_i, f_r) que ofereça a maior redução no valor da função objetivo. Se numa dada iteração não for encontrado um movimento que melhore a solução o procedimento de pesquisa local termina e considera-se que foi atingido um ótimo local.

De forma calcular o ganho (*profit*) obtido por uma possível troca entre duas instalações é utilizada a seguinte expressão [49] [50]:

$$profit(f_i, fr) = \sum_{v:\phi_1(v)\neq f_r} \max\{0, [d_1(v) - d(v, f_i)]\} - \sum_{v:\phi_1(v)=f_r} (\min\{d_2(v), d(v, f_i)\} - d_1(v)) \quad (25)$$

Onde, um *profit* positivo indica que a troca representará uma redução no valor da função objetivo da solução corrente. Considere ainda a seguinte notação:

- $d(v, f)$ representa a distância entre os vértices v e f;
- $d_1(v)$ representa a distância entre o vértice v e o vértice mediana que se encontra mais próximo;
- $d_2(v)$ representa a distância entre o vértice v e o segundo vértice mediana mais próximo;
- $\phi_1(v)$ identifica a mediana mais próxima de v;

- $\phi_2(v)$ identifica a segunda mediana mais próxima de v.

Efetuar um movimento de troca entre duas instalações (f_i, f_r) implica que os vértices cliente que se encontravam alocados à instalação f_r (que deixa de ser mediana) deverão ser realocados à respetiva mediana mais próxima. Além disso, clientes que se encontravam alocados a medianas não envolvidas no movimento, poderão encontrar na nova mediana f_i a mediana mais próxima, devendo também neste caso ser efetuada a realocação dos clientes que se enquadrem neste cenário.

A Figura 18, representa um movimento de troca (*swap*) entre os vértices 3 e 4, em que é possível verificar a alteração da morfologia da solução provocada pelo processo de realocação dos restantes vértices. As arestas entre os vértices representam a alocação às medianas.

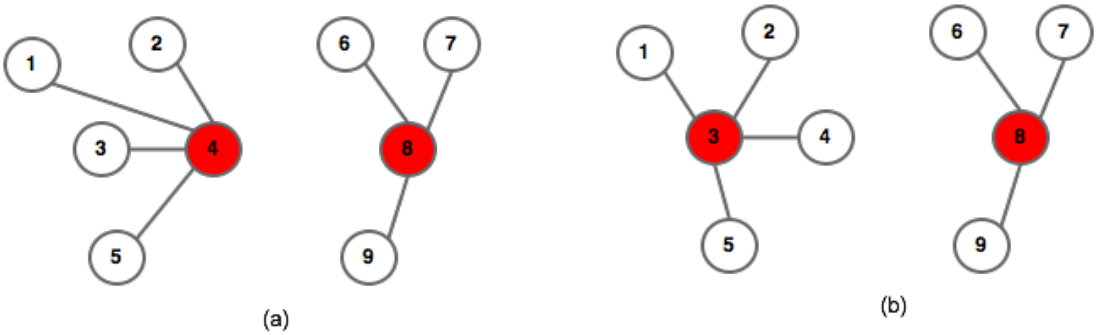

Figura 18 - Movimento swap entre dois vértices.

Importa destacar que nenhum movimento poderá prejudicar a admissibilidade da solução no espaço de soluções do problema primal.

Em relação ao algoritmo proposto por Whitaker [49] que procura a melhor troca (f_i, f_r) com recurso da expressão (25), o *Swap-Based Local Search* tira partido de algumas estruturas de memória extra que lhe garantem maior rapidez. Tal como descrito em [50], essas estruturas são:

- $gain(f_i)$, que representa para cada instalação $f_i \notin S$ a redução obtida no valor da função objetivo pela adição de f_i à solução (simulando que a solução teria $p+1$ vértices medianas). Esse ganho é calculado através da seguinte expressão:

$$gain(f_i) = \sum_{v \in V} \max\{0, d_1(v) - d(v, f_i)\} \tag{26}$$

- $loss(f_r)$, que representa para cada instalação $f_r \in S$ o aumento do valor da função objetivo resultante da remoção de f_r (simulando que a solução teria $p-1$ vértices medianas). A fórmula de cálculo é a seguinte:

$$loss(f_r) = \sum_{v:\phi_1(v)=f_r} \max\left[d_2(v) - d(v, f_r)\right] \tag{27}$$

- $extra(f_i, f_r)$, que simboliza um peso calculado através de (28) que tem em consideração, para cada vértice v, se:
  - a instalação v já esteve alocada a f_r em algum momento do algoritmo;
  - a instalação v nunca esteve alocada a f_r;
  - a instalação v deve ser alocada ou realocada a f_i.

  A fórmula de cálculo é a seguinte:

$$extra(f_i, f_r) = \sum_{v:[\phi_1(v)=f_r] \wedge [d(v,f_i)<d_2(v)]} \left[d_2(v) - \max\left\{d(v, f_i), d(v, f_r)\right\}\right] \tag{28}$$

O uso destas estruturas de memória auxiliares permite ao algoritmo manter informações de iterações anteriores do procedimento, razão pela qual o processo se torna mais acelerado. Graças às estruturas $gain$, $loss$ e $extra$, o cálculo do ganho (*profit*) obtido por um movimento de troca (f_i, f_r) torna-se possível através da seguinte expressão:

$$profit(f_i, f_r) = gain(f_i) - loss(f_r) + extra(f_i, f_r) \tag{29}$$

Na Figura 19, é apresentado o algoritmo do *Swap-Based Local Search* tal como apresentado em [50]. Para além da notação já apresentada anteriormente, considere-se que A representa o conjunto dos clientes afetados, ou seja, os vértices referentes aos clientes que de alguma forma se encontram envolvidos no movimento de troca entre duas instalações.

```
1. Construir uma solução inicial S com recurso a uma
   heurística construtiva.
2. Inicializar A = V.
3. Inicializar a zero as estruturas gain, loss e extra.
4. Enquanto existirem movimentos que melhoram a solução
   fazer:
5.     Para cada v ∈ A fazer
6.         Atualizar as estruturas gain, loss e extra.
7.         Atualizar φ1 e φ2.
8.     Encontrar o melhor movimento (fi, fr) e respetivo profit.
9.     Se profit ≤ 0 então terminar a execução do algoritmo.
10.    Definir A = ∅.
11.    Para cada v ∈ V fazer
12.        Se φ1(v) = fr ou φ2(v) = fr ou d(v, fi) < d(v, φ2(v)) então
13.            A = A ∪ v.
14.    Para cada v ∈ A fazer
15.        Reverter as atualizações feitas nos passos 6 e 7.
16.    Inserir fi em S e remover fr.
17.    Atualizar φ1 e φ2.
```

Figura 19 - Algoritmo Swap-Based Local Search.

O algoritmo apresentado, parte de uma solução inicial construída com recurso a uma heurística construtiva. Nesta fase inicial do algoritmo, considera-se que o conjunto A é composto por todos os vértices $v \in V$ e todas as entradas das estruturas $gain$, $loss$ e $extra$ tomam o valor zero. São ainda determinados, para cada um dos vértices, os valores de $\phi_1(v)$ e $\phi_2(v)$. De seguida, inicia-se a parte iterativa do algoritmo, executada até que não sejam encontrados movimentos capazes de melhorar o valor da função objetivo da solução corrente. Nesta fase, o primeiro passo é atualizar as estruturas $gain$, $loss$ e $extra$ para cada um dos vértices do conjunto A. O segundo passo consiste em encontrar o melhor movimento e respetivo ganho calculando $profit(f_i, f_r)$ para todos os pares (f_i, f_r) com recurso à expressão (29). Uma vez determinado o movimento que será efetuado procede-se à atualização do conjunto A para que este contenha apenas os vértices afetados pelo movimento selecionado. Antes de efetuar o movimento propriamente dito, importa reverter a informação $gain$, $loss$ e $extra$ para cada um dos vértices que agora constam em A, de modo a preparar estas estruturas para as próximas iterações do algoritmo. Por fim, é efetuado o

movimento correspondente à melhor troca encontrada e é atualizada a informação referente às medianas mais próximas ($\phi_1(v)$ e $\phi_2(v)$) para cada um dos vértices. Para uma descrição mais detalhada do algoritmo e dos elementos que o compõem remete-se para a leitura do trabalho de Resende e Werneck [50].

De modo a avaliar a real mais valia que o *Swap-Based Local Search* de Resende e Werneck representa comparativamente ao algoritmo proposto por Whitaker foram implementados os dois procedimentos de pesquisa local. Os resultados globais obtidos (tempo e desvio médios) em quatro dos grupos de instâncias testados são apresentados na Tabela 4:

Instância/Autor	**Whitaker** [49]		**Resende e Werneck** [50]	
	Tempo (s)	Desvio médio (%)	Tempo (s)	Desvio médio (%)
ORLIB	3,413	0,352	0,448	0,244
SL	73,293	0,948	2,686	1,030
Ruspini	0,006	2,978	0,002	1,495
GR	0,023	0,664	0,007	0,640

Tabela 4 - Resultados dos algoritmos de melhoramento.

Como é possível observar na tabela apresentada, *Swap-Based Local Search* não só é mais rápido que a abordagem apresentada por Whitaker, como também supera a esta última na qualidade global das soluções obtidas.

Testes de redução

A pesquisa levada a cabo pelo *Swap-Based Local Search* implica a avaliação de todas as trocas possíveis, de modo a determinar qual o melhor movimento, situação que torna este processo mais moroso quanto maior for o número de clientes do problema e o número de instalações medianas que deverão ser escolhidas. De forma a reduzir o espaço de pesquisa e, deste modo, melhorar o desempenho do algoritmo de melhoramento descrito foram implementados testes de redução que permitem determinar arestas que não constam da solução ótima do problema ($x_{ij} = 0$) e arestas que obrigatoriamente farão parte dessa solução e que por isso deverão ser fixadas ($x_{ij} = 1$). O mesmo acontece em relação aos vértices que poderão ser excluídos ou fixados, sendo que um vértice fixado será escolhido como mediana já que será parte integrante de uma solução ótima. Note-se que nem todas as arestas/vértices que constituem um determinado problema poderão ser fixados ou eliminados.

Os testes de redução aqui apresentados são os descritos por Santos em [42], que consistem numa adaptação dos testes efetuados por Briant e Naddef [66]. Estes testes usufruem de informação obtida no espaço primal (limite superior) e dual (limite inferior, custos auxiliares e custos lagrangeanos, estes dois últimos explicados em 4.1.2) do problema. Uma vez que a metaheurística RAMP se caracteriza pela eficiente exploração da relação entre estes espaços de soluções, torna-se pertinente a incorporação dos testes de redução nos algoritmos RAMP implementados.

No total foram quatro os testes de redução implementados. Os dois primeiros procuram determinar arestas que devem ser excluídas (ou seja, fixadas a zero), enquanto que os dois últimos têm o objetivo de fixar vértices a zero ou a um, respetivamente. Considere na definição dos testes que:

- f^k identifica o vetor de custos auxiliares na iteração k;
- d^k identifica a matriz de custos lagrangeanos na iteração k;
- V_0 e V_1 identificam o conjunto de vértices fixados a 0 ou 1, respetivamente;
- A_0 e A_1 identificam o conjunto de arestas fixadas a 0 ou 1, respetivamente.

Teste de redução 1

O primeiro teste de redução tem o objetivo de fixar arestas a zero (exclusão de arestas). Uma aresta será excluída sempre que a sua adição à solução dual corrente resulte num limite inferior maior que o melhor limite superior encontrado, tendo em consideração o custo lagrangeano d^k associado a essa aresta. A Figura 20 apresenta um algoritmo referente a este teste:

1. Para todos os vértices mediana i fazer
2. Para todos os vértices não mediana j fazer
3. Se $(i,j) \notin A_0$ e $d_{ij}^k > 0$ e $Z_\lambda^k + d_{ij}^k > Z^*$ então
4. $A_0 = A_0 \cup (i,j)$.
5. Terminar.

Figura 20 – Algoritmo do teste de redução 1.

Teste de redução 2

Também neste teste o objetivo é encontrar arestas que deverão ser excluídas, desta vez a partir da avaliação do impacto que teria a adição de uma aresta (j,l) que não consta na

solução dual (nenhum dos vértices é mediana) teria em detrimento da remoção de outra aresta associada à mediana com o maior custo auxiliar, tornando j a nova mediana. A figura 21 representa o algoritmo correspondente.

1. Para todas as arestas (j,l) em que j e l não são medianas fazer:
2. Se $(j,l) \notin A_0$ e $d_{jl}^k > 0$ e $Z_\lambda^k - f_p^k + f_j^k + d_{ij}^k > Z^*$ então
3. $A_0 = A_0 \cup (j,l)$.
4. Terminar.

Figura 21 - Algoritmo do teste de redução 2.

Teste de redução 3

Este é o primeiro teste que procura a fixação de vértices. Neste caso, um vértice j será fixado a zero sempre que a sua adição como mediana à solução dual corrente em detrimento da exclusão da atual mediana com maior custo auxiliar resulte num limite inferior maior que o melhor limite superior até então obtido (ver Figura 22).

1. Para todos os vértices j não mediana fazer
2. Se $j \notin V_0$ e $Z_\lambda^k - f_p^k + f_j^k > Z^*$ então
3. $V_0 = V_0 \cup j$.
4. Terminar.

Figura 22 - Algoritmo do teste de redução 3.

Como consequência da exclusão de um vértice j de uma solução ótima, nenhuma aresta que conecte outro vértice l a j poderá constar nessa solução. Deste modo, $A_0 = A_0 \cup (j,l)\ \forall\ (j,l) \in V$.

Teste de redução 4

A finalidade do quarto e último teste, é encontrar vértices que pertencem ao conjunto de medianas de uma solução ótima e que, portanto, devem ser fixados a um. O teste consiste em remover o estatuto de mediana à mediana i com o menor custo auxiliar e atribuir esse estatuto ao vértice j não mediana com o menor custo auxiliar. Se desta operação resultar

um limite inferior maior que o melhor limite superior corrente, então i deverá ser fixado a um (ver Figura 23).

1. Para todos os vértices i mediana fazer
2. Se $i \notin V_1$ e $Z_\lambda^k - f_i^k + f_{p+1}^k > Z^*$ então
3. $V_1 = V_1 \cup j$.
4. Terminar.

Figura 23 - Algoritmo do teste de redução 4.

Se i é uma mediana, então não poderá existir nenhuma aresta que aloque i a qualquer outro vértice j, ou seja, $A_0 = A_0 \cup (j,i)\ \forall\ (j,i) \in A$. Outra das consequências, é que toda a aresta $(i,j) \notin A_1$ com cujo custo lagrangeano d_{ij} seja inferior a zero, e que respeita a inequação 30 deverá pertencer ao conjunto A_1. Consequentemente, j deverá ser fixado a zero e todas as implicações já referidas desta ação são aplicadas.

$$Z_\lambda^k - d_{ij}^k > Z^* \qquad (30)$$

Dos quatro testes descritos, emergem algumas implicações lógicas que permitem detetar outros vértices que deverão ser fixados a um. A primeira implicação permite fixar um vértice $j \notin V_1$ a um, se $(l,j) \in V_0\ \forall\ l \in V \backslash j$. A segunda implicação, adiciona uma aresta (i,l) ao conjunto A_1, se esta for a única aresta de saída de l ($l \in V_0$) e todas as outras arestas pertençam ao conjunto A_0. Se a aresta (i,l) é a única de saída do vértice l implica que l se encontra alocado a i e, portanto, que i é uma mediana e deverá ser adicionado ao conjunto V_1, caso ainda não pertença a V_1 e todas as implicações resultantes desta operação deverão ser consideradas.

A utilização dos testes de redução, permite acelerar o procedimento *Swap-Based Local Search* na medida em que o algoritmo, na sua busca pela melhor troca não considerará inserir na solução vértices pertencentes ao conjunto V_0, uma vez que já foi determinado que os elementos deste conjunto não poderão constar na solução ótima. De igual modo, serão excluídos movimentos que pretendam remover da solução vértices, que pertencem ao conjunto V_0.

A eficácia dos testes de redução varia em função das caraterísticas da instância de um problema *P-Median* e também do momento da execução do algoritmo em que estes são efetuados. Segundo Santos [42], os testes começam a ser efetivos quando o desvio entre o

limite superior e o limite inferior (GAP_λ) se encontra abaixo dos 5%. Este desvio é calculado com recurso à seguinte expressão:

$$GAP_\lambda = \frac{Z - Z_\lambda}{Z_\lambda} * 100 \quad (31)$$

Na Tabela 5, são apresentados os resultados obtidos pelos testes de redução em algumas das instâncias de teste utilizadas no que respeita à quantidade de vértices que foi possível fixar a zero (V_0) ou a um (V_1). A coluna GAP_λ reflete o valor do desvio entre o limite inferior e o limite superior aquando da execução dos testes.

Instância	Número de medianas	Número de vértices	Vértices eliminados (%)	Medianas fixadas (%)	GAP_λ (%)
pmed8	20	200	86%	95%	0,024%
pmed34	140	700	57%	0%	0,038%
pmed40	90	900	47%	1%	0,117%
sl700	233	700	21%	7,3%	0,057%
ruspini150	50	150	10,6%	4%	0,102%
fl1400	20	1400	0,4%	90%	0,96%
pcb3038	30	3038	2%	90%	0,77%
pcb3038	50	3038	8,7%	96%	0,93%

Tabela 5 - Estatísticas dos testes de redução.

Os dados da tabela apresentada permitem concluir que os testes de redução implementados têm uma boa taxa de eficiência e são capazes de excluir uma quantidade considerável de movimentos que seriam considerados pelo algoritmo *Swap-Based Local Search*.

4.1.2. Método dual

O método dual tem por objetivo resolver o problema dual obtido através da aplicação de uma relaxação lagrangeana ao problema *P-Median*, fazendo uso da otimização por subgradiente.

A relaxação lagrangeana do problema *P-Median* considerada neste trabalho consiste na eliminação do conjunto de restrições (20) que garante que cada cliente se encontra alocado a uma e só uma mediana ou é ele próprio uma mediana, tal como proposto por Beasley [51] e Narula e Ogbu [52] que verificaram que é a relaxação deste conjunto de restrições que produz melhores resultados.

A eliminação do referido conjunto de restrições resulta no seguinte problema dual lagrangeano LP^{λ}:

$$LP^{\lambda} = Min \sum_{i \in V} \sum_{j \in V} (c_{ij} - \lambda_j) . x_{ij} + \sum_{i \in V} \lambda_j \quad (32)$$

$s.a$:

$$\sum_{i \in V} x_{ii} = p \quad (33)$$

$$x_{ij} \leq x_{ii} \; \forall (i,j) \in A \quad (34)$$

$$x_{ii} \in \{0,1\} \forall i \in V \quad (35)$$

$$x_{ij} \geq 0 \forall (i,j) \in A \quad (36)$$

Note-se a presença na função objetivo (32) dos multiplicadores de lagrange λ que penalizam a eliminação do conjunto de restrições (20).

Na formulação apresentada, $c_{ij} - \lambda_j$ identifica os custos lagrangeanos que doravante serão identificados por d_{ij}. O custo auxiliar f_i associado a cada vértice $i \in V$, pode ser calculado com recurso à seguinte expressão:

$$f_i = d_{ii} + \sum_{j \in V \setminus i} \min \{0, d_{ij}\} \quad (37)$$

Deste modo, o problema dual lagrangeano LP^{λ} pode ser reformulado da seguinte forma [42]:

$$LP^{\lambda} = Min \sum_{i \in V} f_i x_{ii} + \sum_{j \in V} \lambda_j \quad (38)$$

$s.a$:

$$\sum_{i \in V} x_{ii} = p \quad (39)$$

$$x_{ii} \in \{0,1\} \forall i \in V \quad (40)$$

Considerando a formulação (38) a (40), resolver o problema LP^{λ} passa por encontrar os p vértices com o menor custo auxiliar.

Descrição do algoritmo

Na Figura 24, é apresentado o algoritmo Dual RAMP para o problema *P-Median*, onde:

- $\hat{Z}$ representa a solução projetada no espaço primal de soluções;
- Z_{ub} representa o limite superior;
- Z_{max} identifica o limite inferior;
- Z_{λ}^{*} identifica a solução dual;
- δ_j representa o valor do gradiente associado ao vértice j no método do subgradiente;
- $||\delta||$ identifica um valor denominado norma.

1. Inicializar $\pi = 1.5$, $k_{max} = 500$, $k = 0$, $Z_{max} = 0$, $nonImprovedIte = 0$, $\lambda_j = \min_{i \in I} c_{ij \forall j \in V}$.
2. Obter Z_{ub} pela resolução do problema original através do $Greedy$.
3. $Terminar = falso$.
4. Enquanto $Terminar = falso$ e $k_{max} > k$ e $\pi > 0.0003$ fazer
5. $\delta_j = 1 - \sum_{i \in I} x_{ij} \forall j \in V$.
6. Se $f(Z_{ub}) - f(Z_{max}) < 1$ ou $||\delta|| = 0$ então
7. $Terminar = verdadeiro$.
8. Senão
9. $\Delta = \frac{\pi(f(Z_{ub}) - f(Z_\lambda^*))}{||\delta||}$
10. $\lambda_j = \lambda_j + \Delta\delta_j \forall j \in V$
11. Obter Z_λ^* pela resolução do dual lagrangeano.
12. Se k%100=0 e obtivemos um novo limite inferior e $GAP_\lambda < 5\%$ então
13. Efetuar testes de redução.
14. Obter $\hat{Z}$ pela projeção de Z_λ^*.
15. Se obtivemos um novo limite inferior então
16. $\hat{Z} = Swap - Based\ Local\ Search$.
17. Se $f(Z_{ub}) > \hat{Z}$ então
18. $Z_{ub} = \hat{Z}$.
19. Se $f(Z_\lambda^*) > f(Z_{max})$ então
20. $Z_{max} = Z_\lambda^*$.
21. $nonImprovedIte = 0$.
22. Senão
23. $nonImprovedIte++$.
24. Se $k! = 0$ e $nonImprovedIte\%10$=0 então
25. $\pi = \pi/2$.

Figura 24 – Algoritmo Dual RAMP.

O algoritmo começa com a inicialização dos multiplicadores de lagrange para todos os vértices $j \in V$ com o valor da distância à instalação mais próxima. O parâmetro *agility* (π) assume, inicialmente, o valor 1.5.

O passo seguinte, consiste no cálculo do limite superior inicial através da construção de uma solução admissível no espaço primal de soluções do problema *P-Median* através da heurística *Greedy*.

De seguida, o algoritmo executa em cada iteração um conjunto de operações, oscilando entre o espaço primal e dual do problema, até que seja satisfeito um dos critérios de paragem. Em primeiro lugar, é calculado o gradiente para todos os vértices $j \in V$. Este cálculo avalia o número de medianas a que o vértice se encontra alocado (relembre-se que o problema dual permite a alocação de um vértice a várias medianas), sendo que idealmente, se j apenas estiver alocado a uma mediana, $\delta_j = 0$. O próximo passo, é o cálculo do parâmetro Δ que será utilizado, juntamente com δ, na atualização dos multiplicadores de lagrange. Seguidamente, é resolvido o problema dual lagrangeano. Como já referido, a resolução deste problema passa por encontrar os p vértices com o menor custo auxiliar, contudo, neste estudo foi introduzida uma alteração que permite que as solução duais obtidas convirjam mais rapidamente para melhores soluções. A referida alteração tem por base a observação que o procedimento de pesquisa local tende a escolher alguns vértices para medianas com mais frequência do que outros, existindo até vértices nunca escolhidos. Assim, achou-se pertinente efetuar o registo dessa frequência de modo a tirar partido dessa informação para impulsionar o desempenho do algoritmo. Assim definiu-se que a partir da vigésima iteração a resolução do problema dual é efetuada considerando:

$$\begin{cases} (0.1 * p) & vértices\ escolhidos\ com\ mais\ frequência\ pelo\ método\ de\ melhoramento \\ (0{,}9 * p) & vértices\ com\ o\ menor\ custo\ auxiliar \end{cases}$$

A justificação para apenas aplicar esta metodologia a partir da vigésima iteração prende-se com a necessidade de permitir ao método de melhoramento construir algum histórico. Após a resolução do problema dual, são efetuados testes de redução de modo a diminuir o espaço de pesquisa do algoritmo de melhoramento. Estes testes são efetuados a cada cem iterações em que o desvio entre o limite superior e o limite inferior é menor que cinco por cento. De seguida, é feita a projeção da solução dual para o problema primal (o método de projeção será descrito posteriormente) para que lhe seja aplicado o algoritmo de melhoramento *Swap-Based Local Search*. De modo a aumentar a performance do algoritmo, o *Swap-Based Local Search* apenas será executado assim que se considerar que da resolução do problema dual resultou uma solução promissora, neste caso em concreto, assim que é obtido um novo limite inferior. O parâmetro π será reduzido para metade a cada dez iterações sem melhorar o limite superior, sendo reinicializado a cada $30 + p$ iterações.

Os critérios de paragem utilizados pelo algoritmo são os seguintes:

- A diferença entre o limite superior e inferior, é inferior a 1;
- Foi atingido o limite máximo de iterações estabelecido;
- O valor de π é inferior a 0.0003;
- O valor de $||\delta||$ é 0 (implica que todos os vértices se encontram alocados a uma e só uma mediana).

Método de projeção

O método de projeção tem o papel de tornar admissível para o problema *P-Median* a solução obtida pela resolução do problema dual lagrangeano. Uma vez que a relaxação do problema consiste na eliminação do conjunto de restrições (20), poderão ocorrer três situações distintas:

- Existe pelo menos um vértice não alocado (solução não admissível);
- Existe pelo menos um vértice alocado a mais do que uma mediana (solução não admissível);
- Todos os vértices encontram-se alocados a uma e só uma mediana, ou são eles próprios uma mediana, situação ideal que torna a solução admissível.

Na Figura 25, estas três situações encontram-se representadas por a), b) e c), respetivamente:

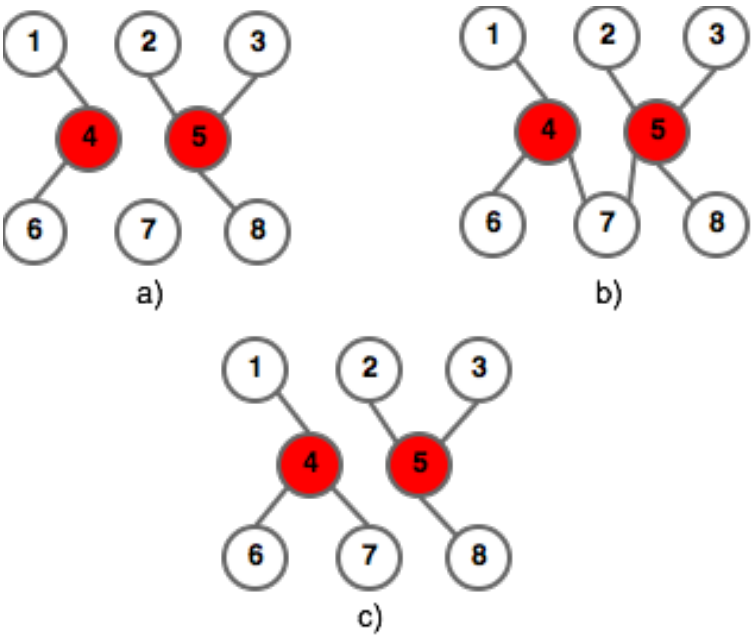

Figura 25 - Possíveis alocações.

Tornar a solução admissível passa por garantir que $\delta_j = 0$ para todos os vértices $j \in V$. Para isso, o método de projeção, nos casos em que um vértice não se encontra alocado, efetua a

alocação deste à mediana mais próxima; e nos casos em que o vértice se encontra multi alocado apenas efetuará a afetação à mediana mais próxima de entre aquelas a que se encontra alocado. Dadas as caraterísticas do problema, em que o conjunto de localizações possíveis coincide com o número de clientes, um vértice mediana deverá ser afeto a si próprio, sendo eliminadas as possíveis restantes alocações.

4.2. Algoritmo PD-RAMP

O PD-RAMP consiste na versão mais sofisticada da metaheurística RAMP. Pode ser visto como uma extensão do Dual RAMP, ao qual acrescenta um método evolutivo que lhe permite intensificar a exploração do espaço primal e extrair mais informação da relação existente entre os dois espaços do problema, o que permite a consideração de soluções não visitadas pelo Dual RAMP, resultando numa solução final cuja qualidade é, geralmente, superior.

A implementação do PD-RAMP pode ser conseguida tendo como ponto de partida o algoritmo Dual RAMP já implementado no qual se integra o referido método evolutivo (frequentemente uma pesquisa por dispersão, como acontece neste estudo) dotado do caraterístico conjunto de referência que será atualizado com soluções obtidas com base nos métodos dual e primal do algoritmo.

A motivação para a implementação deste algoritmo é inerente ao facto de o PD-RAMP ser capaz de obter resultados de melhor qualidade comparativamente à versão menos sofisticada do RAMP.

Na Figura 26, é apresentado um esquema do funcionamento geral do PD-RAMP, Onde se destacam a *pool* de soluções, o conjunto de referência e os métodos tradicionais da pesquisa por dispersão integrados com aqueles já apresentados no Dual RAMP.

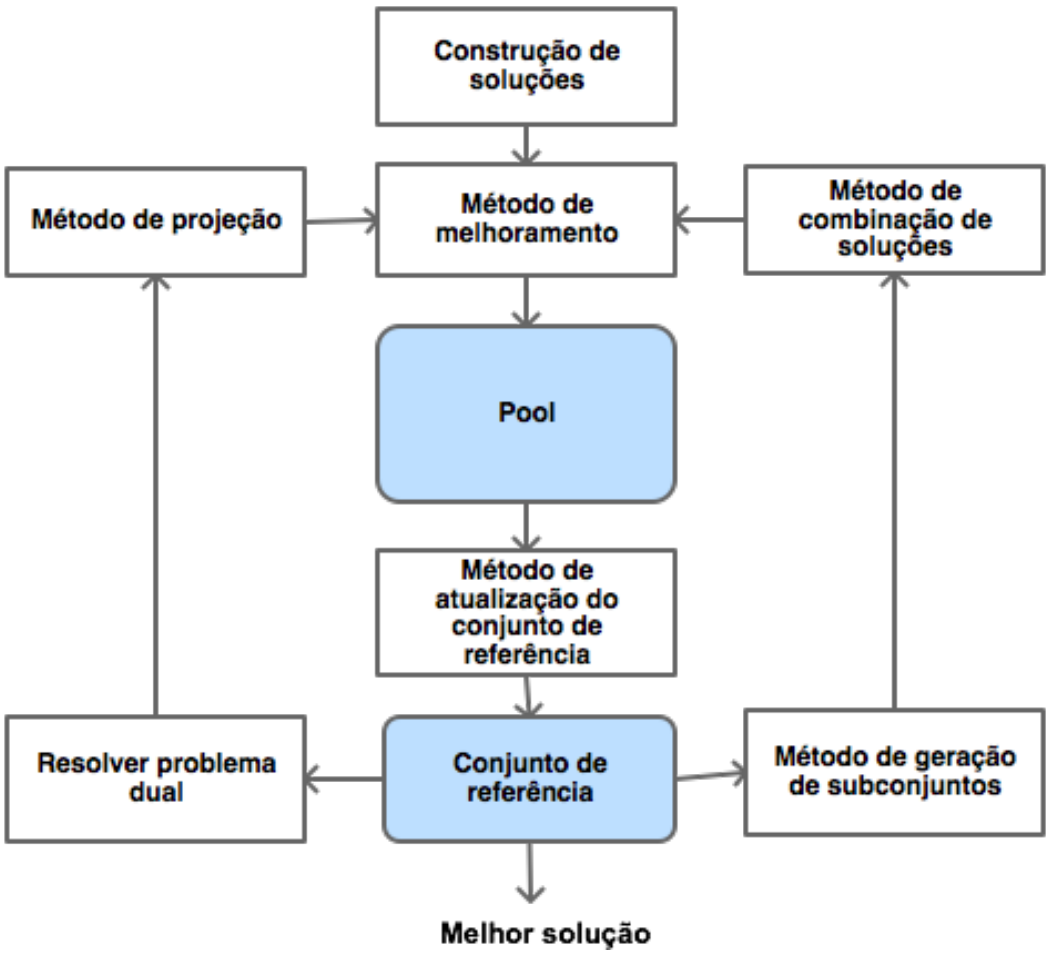

Figura 26 - Funcionamento geral do algoritmo PD-RAMP.

Na Figura 27, é apresentado o algoritmo genérico do PD-RAMP que tem em consideração os componentes presentes na figura anterior.

```
1. Procedimento GRASP.
2. Método de atualização do conjunto de referência.
3. Enquanto um dos critérios de paragem não for satisfeito
4.     Resolver o problema dual.
5.     Aplicar método de projeção à solução dual.
6.     Aplicar método de melhoramento à solução projetada.
7.     Método de atualização do conjunto de referência.
8.     Método de geração de subconjuntos.
9.     Método de combinação de soluções.
10.    Aplicar método de melhoramento.
11.    Método de atualização do conjunto de referência.
```

Figura 27 - Algoritmo genérico PD-RAMP.

O primeiro passo do algoritmo é a geração de soluções com recurso ao procedimento GRASP, soluções essas que irão popular a *pool* utilizada pela pesquisa por dispersão, após a aplicação, a cada uma delas, de um algoritmo de melhoramento. Esta é já a primeira diferença comparativamente ao algoritmo Dual RAMP, no qual apenas era construída uma solução inicial que seria utilizada como limite superior. Assim que este processo se encontra concluído, é feita a construção do conjunto de referência que deverá conter soluções de boa qualidade e soluções diversificadas.

No passo 4, dá-se início ao bloco iterativo do algoritmo com a resolução do problema dual de modo semelhante ao efetuado no algoritmo Dual RAMP. A solução obtida neste passo é sujeita ao método de projeção, com vista a repor a sua admissibilidade, para que de seguida lhe seja aplicado um método de melhoramento. A seguir, a solução é adicionada à pool de soluções e é invocado o método de atualização do conjunto de referência que irá testar a viabilidade de incluir no conjunto de referência as soluções presentes na *pool*.

Na eventualidade da existência de novas soluções no conjunto de referencia torna-se pertinente a execução do método de combinação de soluções, na tentativa de se encontrar novas soluções. A chamada deste método é precedida da aplicação do método de geração de subconjuntos que agrupa as soluções a combinar. Cada uma das soluções obtidas através da execução do método de combinação de soluções, é sujeita ao método de melhoramento, e as soluções resultantes são adicionadas à *pool* para que de seguida seja invocado, mais uma vez, o método de atualização do conjunto de referência, completando assim uma iteração do PD-RAMP.

O algoritmo será executado até que pelo menos um dos critérios de paragem estabelecidos seja atingido.

4.2.1. Método primal

O método primal do PD-RAMP engloba duas metaheurísticas principais: o procedimento GRASP, utilizado para construir a *pool* de soluções inicial, e a pesquisa por dispersão. De seguida serão analisados estes dois componentes.

Construção de soluções

Enquanto que o algoritmo Dual RAMP necessita apenas de uma solução inicial, construída com recurso à heurística *Greedy*, o PD-RAMP necessita de várias soluções que irão popular inicialmente a *pool* de soluções que irá “alimentar” a pesquisa por dispersão. Para esse

efeito foi utilizado o procedimento GRASP que permite a construção de soluções distintas que englobam soluções diversificadas e soluções de boa qualidade.

O algoritmo GRASP implementado neste estudo é apresentado na Figura 28.

```
1. Inicializar k = 0.
2. Enquanto k < poolSize fazer
3.     Obter uma solução S com recurso à heurística RGreedy.
4.     Aplicar Swap − Based Local Search.
5.     Adicionar S à pool.
6.     k++.
```

Figura 28 - Procedimento GRASP implementado.

O procedimento GRASP proposto, constrói as soluções com recurso à heurística *RGreedy*, uma vez que não faria sentido utilizar a heurística *Greedy* devido ao seu caráter determinístico (as soluções geradas seriam todas iguais). Cada uma das soluções é submetida ao algoritmo de pesquisa local *Swap-Based Local Search*, que irá melhorar a qualidade da solução.

A execução do procedimento GRASP apresentado prossegue com a adição da solução à *pool* (ordenada por ordem crescente do valor de função objetivo da solução), até que seja atingido o tamanho da *pool* ($poolSize$).

Pesquisa por dispersão

A Pesquisa por Dispersão utilizada no algoritmo PD-RAMP foi implementada com base na descrição efetuada na secção 2.4.3. Como já referido, é através da adição da pesquisa por dispersão ao Dual RAMP que se torna possível a obtenção de uma algoritmo mais sofisticado e capaz de conseguir soluções de melhor qualidade.

Cada um dos cinco principais métodos da Pesquisa por Dispersão proposta por Laguna e Martí [32] será de seguida explicado no contexto do algoritmo implementado, uma vez que alguns destes métodos deverão ser adaptados ao problema a que são aplicados. Considera-se que a *pool* de soluções é uma lista ligada de soluções, ordenada por ordem crescente, de acordo com o valor da função objetivo de cada solução, de tamanho ($PSize$) igual a 20 (para instâncias de dimensão superior a 2000 vértices $PSize$ assume o valor de 10), e que no conjunto de referência constam $b = 10$ soluções.

Método de geração de soluções diversificadas

As soluções são construídas recorrendo ao procedimento GRASP, descrito na secção anterior, que garante a obtenção de soluções de boa qualidade e de soluções diversificadas. Cada uma das soluções geradas pelo GRASP, é adicionada ordenadamente à *pool* de soluções até que $|pool| = PSize$. O desempenho do algoritmo sai beneficiado pela manutenção da *pool* ordenada, sendo também benéfico garantir que a *pool* não contêm soluções repetidas que apenas trariam redundâncias ao algoritmo.

Método de melhoramento de soluções

O método de melhoramento utilizado na Pesquisa por Dispersão implementada, é o *Swap-Based Local Search*, que parte de uma solução resultante do método de combinação de soluções e procura encontrar uma solução de melhor qualidade. Após sujeição ao método de melhoramento, cada uma das soluções é adicionada à *pool* de soluções.

Método de atualização do conjunto de referência

O método de atualização do conjunto de referência é responsável por construir e atualizar o conjunto de referência com soluções de qualidade e diversificadas e sem a existência de soluções duplicadas. No algoritmo proposto considera-se que existem no conjunto de referência $b/2$ soluções de qualidade ordenadas de acordo com o valor da função objetivo de cada solução, e $b/2$ soluções diversificadas ordenadas em função do seu grau de diversificação. A existência destas duas categorias de soluções é de extrema importância, uma vez que permite ao algoritmo alcançar soluções ainda não visitadas (se apenas existissem no conjunto de referência soluções de qualidade a tendência para o algoritmo ficar preso num ótimo local seria muito maior).

De modo a calcular o grau de diversificação de uma solução é necessário estabelecer uma medida de diversificação que permita efetuar uma avaliação equitativa de todas as soluções.

O cálculo da diversificação utilizado neste trabalho é efetuado como a seguir se descreve. Considerando que:

$$\begin{cases} 0 \; indica \; que \; o \; vértice \; não \; é \; mediana \\ 1 \; indica \; que \; o \; vértice \; é \; mediana \end{cases}$$

É possível representar uma solução S, para uma dada instância de um problema *P-Median* recorrendo à notação binária. Por exemplo, se o problema é composto por dez vértices, dos quais quatro devem ser escolhidos como medianas, uma solução S_1 em que os vértices dois, três, sete e nove são medianas pode ser representada do seguinte modo:

$$S_1 = 0110001010$$

Uma outra solução, S_2, em que os vértices mediana são o três, o cinco, o oito e o dez terá a seguinte representação:

$$S_2 = 0010100101$$

O cálculo da diversificação de uma solução em relação a outra é feito através do somatório do módulo da diferença entre o valor de cada índice k de ambas as soluções, ou seja:

$$div(S_1, S_2) = \sum_{k=0}^{n} |S_1^k - S_2^k| \qquad (41)$$

Deste modo o grau de diversificação de S_2 em relação a S_1 é assim calculado:

0	1	1	0	0	0	1	0	1	0
0	0	1	0	1	0	0	1	0	1

0 + 1 + 0 + 0 + 1 + 0 + 1 + 1 + 1 + 1 = **6**

No método de atualização de soluções é efetuado o cálculo da diversificação de cada solução existente na *pool* em relação a todas as soluções já existentes no conjunto de referência. As $b/2$ soluções com maior somatório do cálculo da diversificação são incluídas no conjunto de referência. No que diz respeito às soluções de qualidade, serão adicionadas ao conjunto de referência as $b/2$ soluções com menor valor de função objetivo.

Assim que uma solução é inserida no conjunto de referência, esta é eliminada da *pool* de soluções e é registada a iteração em que a solução foi admitida neste conjunto. No caso de se tratar de uma atualização do conjunto de referência, a solução irá substituir no conjunto de referência a pior solução existente em termos de valor de função objetivo ou do grau de diversificação, dependendo do tipo de solução.

O algoritmo descrito utiliza uma atualização estática do conjunto de referência, o que significa que o conjunto de referência apenas será atualizado após todos os conjuntos gerados pelo método de geração de subconjuntos terem sido combinados pelo método de combinação de soluções.

Método de geração de subconjuntos

O método de geração de subconjuntos é responsável pela criação de todos os subconjuntos de tamanho n com soluções do conjunto de referência. Posteriormente, o método de

combinação de soluções combina as soluções de cada subconjunto gerado. Uma vez que cada solução deverá ser combinada com todas as outras, serão obtidos subconjuntos com soluções de qualidade e diversificadas.

O número de subconjuntos únicos gerados varia, logicamente, em função de n, e pode ser calculado pela seguinte expressão:

$$numero\ de\ combinações\ possíves = (b^2 - b)/n \quad (42)$$

No algoritmo proposto são formados subconjuntos de tamanho $n = 2$. Considerando que o conjunto de referência é composto por dez soluções, é possível obter quarenta e cinco subconjuntos, cada um dos quais dará origem a uma solução.

Método de combinação de soluções

Este método é invocado logo após a execução do método de geração de subconjuntos, e combina cada um dos subconjuntos de soluções, resultando numa solução que será adicionada à *pool*.

Neste estudo, cada subconjunto é combinado através de dois métodos distintos. O primeiro método consiste em criar uma nova solução na qual os vértices medianas comuns a todas as soluções do subconjunto são também medianas na nova solução. As restantes medianas são escolhidas aleatoriamente de uma lista que contém as medianas de todas as soluções, até que a nova solução tenha também p vértices medianas. No caso de um subconjunto com as soluções:

$$S_1 = 0110001010,\ S_2 = 0010100101$$

a solução resultante pela aplicação deste método de combinação contemplará obrigatoriamente o vértice 3 como mediana, uma vez que é uma mediana comum às duas soluções. As restantes três medianas são escolhidas do conjunto $\{2,5,7,8,9,10\}$. Uma possível solução seria:

$$S_{new} = 0110100001$$

O segundo método de combinação utilizado é muito semelhante ao primeiro descrito, apenas com a diferença de não serem automaticamente escolhidos como medianas os vértices com esse estatuto em comum em todas as soluções, residindo a definição das medianas da nova solução exclusivamente na escolha aleatória entre a lista de medianas de todas as soluções.

De modo a serem evitados esforços redundantes, resultantes da combinação de um subconjunto de soluções que já foi submetido a combinação em iterações anteriores do algoritmo, um subconjunto apenas será combinado se pelo menos uma das soluções que o compõem ainda não o tiver sido.

4.2.2. Método dual

A resolução do problema dual é muito semelhante à efetuada no algoritmo Dual RAMP descrito na secção 4.1.2. Contudo o método dual usado no PD-RAMP distingue-se na medida em que o limite superior utilizado corresponde à melhor solução presente no conjunto de referência. Para além disso, a presença do *Swap-Based Local Search* no procedimento GRASP torna possível obter um histórico da frequência com que um vértice é escolhido como mediana ainda numa fase inicial do algoritmo, o que permite a abertura dos $p * 10\%$ dos vértices mais frequentes nas soluções obtidas pelo *Swap-Based Local Search* desde a primeira iteração, ao contrário do que acontece no Dual RAMP em que este procedimento apenas é efetuado a partir da vigésima iteração.

Na Figura 29, é apresentado o algoritmo PD-RAMP no qual pode ser observada a integração dos métodos referentes à Pesquisa por Dispersão. Os critérios de paragem do algoritmo PD-RAMP são os mesmos do Dual RAMP mais um outro herdado da Pesquisa por Dispersão. Deste modo, o algoritmo é executado até que:

- A diferença entre o limite superior e inferior seja inferior a 1;
- Seja atingido o limite máximo de iterações estabelecido;
- O valor de π seja inferior a 0.0003;
- O valor de $||\delta||$ seja 0 (implica que todos os vértices se encontram alocados a uma e só uma mediana);
- Não existem novas soluções no conjunto de referência durante um máximo estabelecido de iterações.

1. Inicializar $\pi = 1.5$, $k_{max} = 500$, $k = 0$, $Z_{max} = 0$, $nonImprovedIte = 0, iteWithoutNewSol = 0$, $\lambda_j = \min_{i \in I} c_{ij \forall j \in V}$.
2. Método de geração de soluções diversificadas.
3. Método de atualização do conjunto de referência.
4. Definir Z_{ub} como sendo a melhor solução do conjunto de referência.
5. $Terminar = falso$.
6. Enquanto $Terminar = falso$ e $k_{max} > k$ e $\pi > 0.003$ fazer
7. $\delta_j = 1 - \sum_{i \in I} x_{ij} \forall j \in V$.
8. Se $f(Z_{ub}) - f(Z_{max}) < 1$ ou $||\delta|| = 0$ ou $iteWithoutNewSol = 50$ então $Terminar = verdadeiro$
9. Senão
10. $\Delta = \frac{\pi(f(Z_{ub}) - f(Z_\lambda^*))}{||\delta||}$
11. $\lambda_j = \lambda_j + \Delta\delta_j \forall j \in V$
12. Obter Z_λ^* pela resolução do dual lagrangeano.
13. Se $k > 100$ e existem novos limites superior e inferior então efetuar testes de redução.
14. Obter $\hat{Z}$ pela projeção de Z_λ^*.
15. Se obtivemos um novo limite inferior e $k > 100$ então
16. $\hat{Z} = Swap - Based\ Local\ Search$.
17. Adicionar $\hat{Z}$ à $pool$ de soluções.
18. Método de atualização do conjunto de referência.
19. Método de geração de subconjuntos.
20. Método de combinação de soluções.
21. Método de atualização do conjunto de referência.
22. Se existem novas soluções no conjunto de referência então $iteWithoutNewSol = 0$ senão $iteWithoutNewSol++$
23. Definir Z_{ub} como sendo a melhor solução do conjunto de referência.
24. Se $f(Z_\lambda^*) > f(Z_{max})$ então $Z_{max} = Z_\lambda^*$; $nonImprovedIte = 0$.
25. Senão $nonImprovedIte++$.
26. Se $k! = 0$ e $nonImprovedIte\%10=0$ então $\pi = \pi/2$.
27. Se $k! = 0$ e $k\%(30 + p) = 0$ então $\pi = 2$.
28. $k++$.

Figura 29 – Algoritmo PD-RAMP.

4.3. Estruturas de dados

O uso de estruturas de dados adequadas é de extrema importância para a obtenção de soluções de qualidade em tempos computacionais reduzidos. Nesta secção serão apresentadas e descritas as estruturas de dados básicas utilizadas no algoritmo, incluindo aquelas que se encontram diretamente ligadas ao problema *P-Median* e aquelas que foram implementadas de suporte a técnicas de resolução utilizadas no algoritmo.

Para facilitar a descrição das estruturas de dados, será considerada uma pequena instância de um problema *P-Median* com 10 vértices, cujo objetivo é escolher a localização de 4 medianas.

Matriz de custos

A matriz de custos, C_{ij}, (Figura 30) tem o papel de armazenar as distâncias entre cada um dos vértices do problema. Será com base nesta matriz, que será possível determinar quais as medianas mais próximas de um cliente e efetuar o cálculo do valor da função objetivo para uma solução do problema. Em todas as instâncias testadas, as distâncias entre dois vértices são simétricas e o custo de alocar um vértice a si próprio (nos casos em que este é uma mediana) é sempre zero. Note-se que i identifica cada uma das localizações possíveis para as medianas, que coincidem com as localizações dos clientes j.

C_{ij}	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	17	3	8	11	1	21	19	9	15
2	17	0	11	6	4	30	20	6	21	20
3	3	11	0	3	4	18	9	30	9	22
4	8	6	3	0	9	2	8	21	26	14
5	11	4	4	9	0	27	25	29	17	3
6	1	30	18	2	27	0	10	22	22	6
7	21	20	9	8	25	10	0	11	4	24
8	19	6	30	21	29	22	11	0	11	5
9	9	21	9	26	17	22	4	11	0	8
10	15	20	22	14	3	6	24	5	8	0

Figura 30 - Matriz de custos.

Vetor de estado das instalações

De modo a representar o estado de uma instalação j (mediana ou não mediana) é utilizado um vetor binário em que 1 identifica que a instalação correspondente a esse índice do vetor é uma mediana e 0 identifica o estado oposto. A figura seguinte representa esse vetor, em que os vértices mediana são 1, 3, 8 e 10.

j

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	0	0	0	0	1	0	1

Figura 31 - Vetor de estado das instalações.

Matriz de alocação

A matriz de alocação x_{ij} permite armazenar a afetação dos clientes às medianas do problema. Trata-se de uma matriz simétrica em que 1 simboliza que o cliente j se encontra alocado à mediana i, e 0 representa o estado oposto.

x_{ij}	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	1	1	0	0	1	0	0	0
4	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1
6	1	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	1	0	0	0	1	1

Figura 32 - Matriz de alocação.

Na Tabela 6, apresenta-se a alocação dos vértices, tendo em consideração os valores da matriz de alocação da Figura 32.

Mediana	Vértices alocados
1	1, 6
3	3, 4, 7
8	2, 8
10	5, 9, 10

Tabela 6 - Alocação de vértices.

Melhores medianas por vértice

O procedimento *Swap-Based Local Search* necessita que seja armazenada informação referente às duas medianas mais próximas de cada um dos vértices. Desta forma torna-se fácil a realocação de um vértice a uma mediana, assim que a sua mediana atual perde esse estatuto. Esta informação é tida em conta no cálculo do $profit$ efetuado pela expressão (29) que avalia o impacto que um movimento *swap* tem no valor da solução. A estrutura implementada é composta por um vetor que contém todos os vértices que formam o problema *P-Median*. Cada uma das posições desse vetor, armazena um apontador para uma lista duplamente ligada composta por nós, que por sua vez contêm informação referente à mediana e à distância dessa mediana à instalação representada pelo índice do vetor de vértices. Cada lista contem apenas dois nós (referentes às duas melhores medianas) ordenados por ordem crescente em função da distância.

Na Figura 33, é apresentada a estrutura descrita, de acordo com a informação da Tabela 6 e da matriz de custos representada na Figura 30. Tendo por exemplo o vértice 5, que se encontra alocado à mediana 10, esta estrutura de dados permite concluir rapidamente que, na eventualidade de o vértice 10 deixar de ser mediana, o vértice 5 deverá ser alocado à mediana 3, uma vez que passaria a ser este o vértice mediana a mais próximo do vértice 5.

j		i	c_{ij}		i	c_{ij}
1	→	1	0	←→	3	3
2	→	8	6	←→	3	11
3	→	3	0	←→	1	3
4	→	3	3	←→	1	8
5	→	10	3	←→	3	4
6	→	1	1	←→	10	6
7	→	3	9	←→	8	11
8	→	8	0	←→	10	5
9	→	10	8	←→	1	9
10	→	10	0	←→	8	5

Figura 33 - Melhores medianas por cliente.

Pool de soluções

A *pool* de soluções é mantida com recurso a uma lista duplamente ligada ordenada (ver Figura 34). Cada nó da lista contém uma estrutura solução que contém: o estado dos vértices nessa solução; o valor da função objetivo; o grau de diversificação da solução em relação a todas aquelas que se encontram no conjunto de referência; e uma variável que permite controlar a iteração em que, hipoteticamente, a solução entrou no conjunto de referência. O uso de uma lista duplamente ligada permite agilizar o processo de ordenação de soluções, com especial destaque para o facto de a lista se manter ordenada aquando da remoção de uma solução. Outra das vantagens do uso de uma lista é possibilitar um número dinâmico de soluções.

Figura 34 - Pool de soluções.

4.4. Resultados computacionais

De modo a avaliar cada um dos algoritmos propostos foram utilizados cinco conjuntos de soluções de teste: Beasley [67] (ORLIB)[1], Reinelt [68] (TSPLIB)[2], Galvão e ReVelle [69]

[1] http://www.brunel.ac.uk/~mastjjb/jeb/info.html

(GR)[3], Senne e Lorena [70] (SL) e Ruspini [71][4]. Cada uma das instâncias é composta n vértices correspondentes às localizações dos clientes e às possíveis localizações das p medianas que definem o problema. No total são testados cento e trinta e seis problemas.

O conjunto ORLIB é composto por quarenta instâncias de diferentes dimensões (n assume valores que variam entre 100 e 900 e p varia entre 5 e 200).

O conjunto TSPLIB contém três instâncias e cada delas é testada com vários valores de p, tal como mostrado na Tabela 7.

Instância	n	Valores de p testados
fl1400	1400	10, 20, 30, 40, 50, 60, 70, 80, 90,100,150, 200, 250, 300, 350, 400, 450, 500
pcb3038	3038	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000
rl5934	5934	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900,1000

Tabela 7 - Conjunto de testes TSPLIB.

O conjunto de teste SL utiliza três das instâncias que formam o conjunto ORLIB mas utiliza diferentes valores de p. Na tabela que se segue, são apresentadas as caraterísticas dessas instâncias e a correspondente instância ORLIB (entre parêntesis).

Instância	n	p
sl700 (pmed34)	700	233
sl800 (pmed37)	800	267
sl900 (pmed40)	900	300

Tabela 8 - Conjunto de testes SL.

O quarto conjunto de testes utilizado, GR contém duas instâncias de diferentes dimensões para as quais são testados diversos valores de p (Tabela 9).

Instância	n	Valores de p testados
gr100	100	5, 10, 15, 20, 25, 30, 40, 50
gr150	150	

Tabela 9 - Conjunto de testes GR.

[2] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
[3] http://www.di.unipi.it/optimize/Data/MEX.html
[4] http://www.unc.edu/~rls/s754/data/ruspini.txt

Por fim, o conjunto Ruspini é composto por uma instância com setenta e cinco localizações na qual p assume valores do conjunto $\{5, 10, 15, 20, 25, 30, 35, 40\}$.

Os conjuntos de teste descritos foram escolhidos uma vez que permitem avaliar o desempenho dos algoritmos implementados em instâncias de pequena, média e grande dimensão, e assim determinar a robustez dos novos algoritmos. Não menos importante, estes são conjuntos de teste clássicos para o problema *P-Median* cuja utilização é uma constante na literatura, o que permite comparar os resultados obtidos com aqueles de outros algoritmos já existentes.

Na Tabela 10, são apresentados os resultados obtidos por algoritmos baseados em outras metaheurísticas, incluindo os melhores algoritmos da literatura para a resolução do problema *P-Median*. São ainda apresentados os resultados obtidos pelos algoritmos implementados neste estudo (GRASP, Pesquisa por Dispersão, Dual RAMP e PD-RAMP), cujos resultados detalhados para cada uma das instâncias de teste pode ser consultado nos anexos. Os resultados apresentados fornecem uma visão global sobre o desempenho dos algoritmos nas várias instâncias de teste descritas (sempre que aplicável) e incluem o tempo médio de execução de cada instância em segundos (T(s)) e o desvio médio percentual, *gap*, entre o valor da solução ótima Z^* (ou da melhor solução conhecida) e o valor da solução Z obtida pelo algoritmo em questão (todos os valores foram arredondados a duas casas décimais). O valor do *gap* pode ser calculado através da seguinte fórmula:

$$gap = \frac{Z - Z^*}{Z^*} * 100 \qquad (43)$$

No que respeita aos resultados dos algoritmos implementados neste trabalho importa referir que todos eles correspondem a apenas uma execução do algoritmo respetivo, efetuada num computador com o sistema operativo Linux Ubuntu 14.10, processador Intel Core i5 CPU M 560 @ 2.67GHz e 4GB de memória RAM.

Autores	ORLIB		TSPLIB fl1400		TSPLIB pcb3038		TSPLIB rl5934		SL		GR gr100 e gr150		Ruspini	
	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)
Pullan (PBS) [59]	0,00	0,49	0,01	37,37	0,00	618,93	0,01	487,58	0,00	3,063	0,00	0,06		
Ren *et al* (ALCMA) [63]	0,00	-	0,00	28,29	0,00	92,00	0,00	523,14	0,00	-	0,00	-		
Hansen e Mladenovic (VNS) [56]	0,07	444,78	0,22	869,39	0,36	865,43								
Alp *et al* (GA) [72]	0,04	18,21									0,04	0,34		
Maroszek e Retting (TS) [73]	0,09	69,30	0,77	243,17										
Resende e Werneck (GRASP) [63]	0,00	14,313	0,06	118,41	0,04	607,15	0,03	2185,55	0,00	53,63	0,01	1,10		
Chiyoshi e Galvão (SA) [61]	0,09	27,28												
Levanova e Loresh (AS) [62]	0,99	794,75												
Mladenovic *et al* (TS) [53]													1,17	7,21
Algoritmos propostos														
GRASP	0,05	3,29	0,48	95,46	0,77	522,57	0,51	1791,71	0,26	46,40	0,13	0,05	0,02	0,02
Dual RAMP	0,00	4,16	0,09	141,97	0,02	970,65	0,04	4051,90	0,09	35,80	0,01	0,16	0,00	0,01
Pesquisa por dispersão	0,01	8,15	0,10	604,70	0,16	4791,44	0,07	11702,02	0,05	156,32	0,00	0,24	0,00	0,05
PD-RAMP	0,00	9,18	0,04	715,17	0,01	6015,38	0,02	14590,70	0,00	123,33	0,00	0,45	0,00	0,02

Tabela 10 - Resultados computacionais

A comparação dos tempos computacionais e da qualidade das soluções obtidas pelos vários algoritmos é uma tarefa que se torna difícil devido a diversos fatores, tais como o número de vezes que cada algoritmo é executado (enquanto que alguns autores apenas executam os seus algoritmos uma única vez, outros efetuam várias execuções e retiram a melhor solução obtida, situação que se verifica nos casos dos algoritmos não determinísticos) e o computador utilizado. Contudo, pela análise da Tabela 10, é possível retirar algumas ilações, que serão de seguida apresentadas.

O algoritmo ALCMA (*Accelerated Limit Crossing Based Multilevel Algorithm*), proposto por Ren *et al.* [61], apresenta atualmente os melhores resultados da literatura para o problema *P-Median* . Este algoritmo consegue reduzir eficientemente o espaço de pesquisa através da eliminação de vértices e da determinação de outros vértices como medianas numa solução. Para além de apresentar tempos computacionais muito competitivos, o algoritmo ALCMA foi capaz de encontrar cinquenta e duas novas melhores soluções conhecidas para as instâncias TSPLIB (o algoritmo é executado nove vezes e é retirada a melhor solução). O algoritmo PBS (baseado num algoritmo genético) proposto por Pullan [56], é também capaz de produzir resultados de muito boa qualidade para as instâncias em que foi testado. Também neste caso o algoritmo foi executado múltiplas vezes, e os testes foram efetuados num *cluster* de computadores com o sistema operativo Linux, pelo que não faz sentido comparar os tempos computacionais. Destaca-se ainda a metaheurística híbrida de Resende e Werneck [60], que recorre ao GRASP e a *Path-Relinking* mostrando-se robusta e capaz de produzir bons resultados, contudo implica tempos computacionais mais elevados, comparativamente às duas anteriores, principalmente no caso da instância TSPLIB – rl5934.

Analisando agora os algoritmos implementados neste estudo, pode-se constatar que o algoritmo PD-RAMP é capaz de produzir resultados de excelente qualidade, muito próximos daqueles considerados os melhores da literatura, sendo capaz de obter todas as melhores soluções conhecidas nos conjuntos ORLIB, SL, GR e Ruspini. Nos restantes grupos de instâncias, o desvio médio percentual é muito próximo de zero (o valor mais alto, 0.04, ocorre na instância TSPLIB – fl1400), o que comprova que o algoritmo PD-RAMP proposto é bastante robusto na resolução de instâncias de pequena, média e grande dimensão. Importa ainda referir que os resultados seriam porventura ainda melhores se, como acontece com outros algoritmos, o algoritmo PD-RAMP tivesse sido executado múltiplas vezes e fosse tida como referência a melhor solução obtida nessas execuções. Além disso, ajustes ao algoritmo, como aumentar o número máximo de iterações ou aumentar o tamanho do conjunto de referência, iriam permitir uma pesquisa mais intensa que conduziria a melhores resultados, resultando contudo num maior tempo computacional, que por si só já é algo elevado, essencialmente nas instâncias TSPLIB. Este facto justifica-se não só pelas

caraterísticas do método de melhoramento utilizado, o *Swap-Based Local Search*, que consome maior tempo computacional quanto maior for o número de vértices da instância, mas acima de tudo, quanto maior for o número de instalações medianas que deverão ser escolhidas, mas também, pelas próprias caraterísticas do algoritmo PD-RAMP, que incorpora em si um procedimento GRASP e os métodos caraterísticos de uma Pesquisa por Dispersão, tudo isto a acrescentar a todos os cálculos associados à Relaxação Lagrangeana.

Sendo o PD-RAMP a versão mais sofisticada da metaheurística RAMP, que pode ser considerado uma evolução do Dual RAMP, é natural que produza resultados de melhor qualidade. Ainda assim, o algoritmo Dual RAMP proposto é capaz de produzir bons resultados, com tempos computacionais inferiores ao PD-RAMP (dos algoritmos propostos, o Dual RAMP é aquele que apresenta a melhor relação qualidade/tempo computacional). O Dual RAMP é capaz de obter todas as soluções ótimas nos conjuntos ORLIB e Ruspini e supera o VNS de Hansen e Mladenovic [53] nas instâncias TSPLIB – fl1400 e pcb3038, o Tabu Search de Maroszek e Retting [70] na instância TSPLIB – fl1400 e o algoritmo de Resende e Werneck [60] na instância TSPLIB – pcb3038.

A Pesquisa por Dispersão proposta, é capaz de produzir soluções de melhor qualidade comparativamente ao Dual RAMP nos casos dos grupos de teste SL e GR, contudo também se verifica a necessidade de tempos computacionais mais elevados para este método.

Por fim, o algoritmo GRASP proposto, é o menos sofisticado dos algoritmos implementados e também aquele que produz soluções de qualidade inferior. A sua simplicidade resulta no entanto em tempos computacionais inferiores, tornando-o a metaheurística indicada para a produção de soluções iniciais de boa qualidade para outros algoritmos, como acontece no PD-RAMP.

Os resultados computacionais apresentados demonstram, mais uma vez, a capacidade da metaheurística RAMP para produzir algoritmos capazes de alcançar resultados de excelente qualidade na resolução de problemas de localização de instalações.

5. Conclusões e Trabalho Futuro

Os problemas de localização de instalações têm por objetivo escolher a localização de uma ou várias instalações, de modo a minimizar o custo de satisfazer as necessidades de clientes ou outras organizações, respeitando um determinado conjunto de restrições. Estes problemas têm inúmeras aplicações no mundo real, como é disso exemplo a escolha da localização de um armazém. Dada a pertinência deste tema, a comunidade científica procura, desde há muito, novos métodos de resolução cada vez mais eficientes para distintos problemas de localização, como é o caso do problema *P-Median* abordado neste estudo.

O problema *P-Median* tem por objetivo escolher p instalações como medianas, de entre um conjunto de instalações disponíveis, de modo a que seja minimizado o custo de alocação de cada cliente à respetiva mediana mais próxima. Trata-se de um problema de otimização combinatória NP-Difícil, o que indica que não pode ser obtida a solução ótima de todas as instâncias do problema em tempo polinomial recorrendo a métodos exatos, sendo sugerida a utilização de métodos heurísticos, que embora não garantam a obtenção da solução ótima para o problema, fornecem soluções de boa qualidade com recursos computacionais reduzidos.

Como consequência do estudo do problema *P-Median,* vários algoritmos metaheurísticos têm vindo a ser propostos para a sua resolução. Contudo, a grande maioria apenas explora no processo de resolução um dos espaços de soluções do problema. A metaheurística RAMP, introduzida por Rego [5], demonstrou ser capaz de corrigir essa lacuna quando aplicada a outros problemas de otimização combinatória, caraterizando-se por explorar a relação entre o espaço dual (soluções correspondentes ao problema sem uma ou várias restrições) e o espaço primal (soluções referentes ao problema original) do problema de forma eficiente. Além disso, algoritmos baseados nesta metaheurística já provaram ser capazes de produzirem resultados de estado-da-arte.

Este estudo teve como finalidade aplicar a metaheurística RAMP ao problema *P-Median*, de modo a verificar se a sua aplicação a este problema seria capaz de atingir resultados de

elevada qualidade, tendo sido propostos dois novos algoritmos para a resolução do *P-Median,* correspondentes a dois níveis distintos de sofisticação da metaheurística RAMP: o Dual RAMP e o PD-RAMP.

O algoritmo Dual RAMP proposto recorre ao uso de uma relaxação lagrangeana do problema *P-Median* e explora essencialmente o espaço dual do problema. Por sua vez, o algoritmo PD-RAMP intensifica a resolução efetuada no espaço primal e a relação existente entre os dois espaços de soluções graças à incorporação no algoritmo Dual RAMP de um método evolutivo, neste caso em concreto, de uma Pesquisa por Dispersão.

Neste estudo foram ainda implementadas estratégias que retirassem o máximo partido da informação do problema que pode ser obtida pela aplicação da metaheurística RAMP, como é o caso de testes de redução que procuram eliminar ou fixar vértices do problema com base nos limites superior e inferior e nos custos auxiliares e lagrangeanos utilizados na relaxação do problema.

Os resultados produzidos por ambos os algoritmos propostos são de excelente qualidade, especialmente os obtidos pelo PD-RAMP, que se aproximam bastante dos melhores da literatura, embora com tempos computacionais mais elevados, facto que comprova, mais uma vez, a eficácia da aplicação da metaheurística RAMP na resolução de problemas complexos de otimização combinatória.

Como perspectiva de trabalho futuro, poderão ser estudados novos testes de redução mais eficazes que os atuais implementados. Poderão também ser testados novos métodos de resolução do problema primal e relaxações matemáticas mais sofisticadas, como a relaxação paramétrica cruzada introduzida por Rego [5]. Por fim, e de modo a reduzir os tempos computacionais, os algoritmos implementados poderão ser paralelizados, o que poderá também permitir uma pesquisa mais intensa nos espaços de soluções, conduzindo também a soluções de melhor qualidade.

Referências bibliográficas

[1] N. Christofides, *Graph theory: An algorithmic approach*. New York: Academic Press, 1975.

[2] J. Current, M. Daskin, and D. Schilling, *Discrete Network Location Models*. 2001.

[3] A. Weber, *Über den standort der industrien*. JCB Mohr, 1909.

[4] R. Farahani and M. Hekmatfar, *Facility Location: Concepts, Models, Algorithms and Case Studies*. Heidelberg: Physica-Verlag HD, 2009.

[5] C. Rego, "RAMP: A new metaheuristic framework for combinatorial optimization," *Metaheuristic Optim. via Mem. Evol.*, pp. 441–460, 2005.

[6] D. Gamboa, "Algoritmos de Memória Adaptativa para a Resolução de Problemas de Optimização Combinatória de Grande Dimensão," Instituto Superior Técnico, 2008.

[7] C. Rego, F. Mathew, and F. Glover, "RAMP for the capacitated minimum spanning tree problem," *Ann. Oper. Res.*, vol. 181, no. 1, pp. 661–681, 2010.

[8] F. Maia, "RAMP para o Problema de Localização de Hubs com Afetação Múltipla e sem Restrições de Capacidade," 2011.

[9] T. Matos, "RAMP para o Problema de Localização de Instalações com Restrições de Capacidade," 2011.

[10] Ó. Oliveira, "Algoritmos RAMP para o Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor," Escola Superior de Tecnologia e Gestão de Felgueiras, 2012.

[11] S. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Oper. Res.*, p. 10, 1964.

[12] O. Kariv and S. Hakimi, "An algorithmic approach to network location problems; part2. The p-medians," *SIAM J. Appl. Math.*, vol. 37, pp. 539–560, 1969.

[13] F. Rothlauf, *Design of Modern Heuristics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[14] J. Bondy and U. Murty, *Graph theory with applications*. London: Macmillan, 1976.

[15] A. Land and A. Doig, “An automatic method of solving discrete programming problems,” *Econom. J. Econom. Soc.*, pp. 497–520, 1960.

[16] V. Vazirani, *Approximation algorithms*. Springer, 2001.

[17] F. Glover and G. Kochenberger, *Handbook of metaheuristics*. 2003.

[18] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Comput. Surv.*, vol. 35.3, pp. 268–308, 2003.

[19] S. Sirenko, “Classification of Heuristic Methods in Combinatorial Optimization,” Informarion Theor. Appl., pp. 303–322, 1993.

[20] T. Stützle, *Local search algorithms for combinatorial problems: analysis, improvements, and new applications*. Sankt Augustin, Germany: Infix, 1999.

[21] E. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[22] J. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[23] F. Glover, “Tabu search-part I,” *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, Jan. 1989.

[24] C. Voudouris, “Guided local search for combinatorial optimisation problems,” *Diss. Univ. Essex*, 1997.

[25] N. Mladenović and P. Hansen, “Variable neighborhood search,” *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.

[26] T. Feo and M. Resende, “Greedy randomized adaptive search procedures,” *J. Glob. Optim.*, pp. 109–133, 1995.

[27] F. Glover, “Tabu search—part II,” *ORSA J. Comput.*, vol. 2, no. 1, pp. 4–32, 1990.

[28] M. Gendreau, *An introduction to tabu search*. Springer US, 2003.

[29] E. Taillard, “Robust taboo search for the quadratic assignment problem,” *Parallel Comput.*, vol. 17, no. 4, pp. 443–455, 1991.

[30] F. Glover, “Heuristics for integer programming using surrograte constraints,” *Decis. Sci.*, vol. 8, no. 1, pp. 156–166, 1977.

[31] F. Glover, M. Laguna, and R. Marti, “Scatter search and path relinking: Advances and applications,” *Handb. metaheuristics*, pp. 1–35, 2003.

[32] M. Laguna, R. Marti, and R. Martí, *Scatter search: methodology and implementations in C*. Springer, 2003.

[33] F. Glover, “Surrogate constraints,” *Oper. Res.*, vol. 16, no. 4, pp. 741–749, 1968.

[34] J. E. Beasley, “Lagrangean heuristics for location problems,” *Eur. J. Oper. Res.*, vol. 65, no. 3, pp. 383–399, 1993.

[35] M. L. Fisher, “The Lagrangian Relaxation Method for Solving Integer Programming Problems,” *Manage. Sci.*, vol. 50, no. 12 Supplement, pp. 1861–1871, 2004.

[36] M. G. Narciso and L. Lorena, “Lagrangean/surrogate relaxation for generalized assignment problems,” *Eur. J. Oper. Res.*, vol. 114, no. 1, pp. 165–177, 1999.

[37] H. Kuhn, “A note on Fermat’s problem,” *Math. Program.*, vol. 4, no. July, pp. 98–107, 1973.

[38] H. Hamacher and Z. Drezner, *Facility location: applications and theory*. Springer, 2002.

[39] Z. Melzak, *Invitation to geometry*. Courier Dover Publications, 2008.

[40] M. Zacharias, *Elementargeometrie und elementare nicht-euklidische Geometrie in Synthetisher Behandlung*. Teubner, 1913.

[41] M. Daskin, *Network and discrete location: models, algorithms, and applications*. 1995.

[42] A. Santos, “Solving large p-median problems using a Lagrangean heuristic,” *Tech. Rep. RR-09-03*, 2009.

[43] J. Reese, “Methods for solving the p-median problem: An annotated bibliography,” *Networks*, vol. 48, no. 3, pp. 125–142, 2006.

[44] A. Kuehn and M. Hamburger, “A heuristic program for locating warehouses,” *Manage. Sci.*, vol. 9, no. 4, pp. 643–666, 1963.

[45] F. Maranzana, “On the location of supply points to minimize transport costs,” *OR*, pp. 261–270, 1964.

[46] M. Teitz and P. Bart, “Heuristic methods for estimating the generalized vertex median of a weighted graph,” *Oper. Res.*, vol. 16, no. 5, pp. 955–961, 1968.

[47] R. Galvão, “A dual-bounded algorithm for the p-median problem,” *Oper. Res.*, vol. 28, no. 5, pp. 1112–1121, 1980.

[48] D. Erlenkotter, “A dual-based procedure for uncapacitated facility location,” *Oper. Res.*, vol. 26, no. 6, pp. 992–1009, 1978.

[49] C. S. Bureau and S. B. Development, “A fast algorithm for the greedy interchange for large-scale clustering and median location problems,” *INFOR*, vol. 21, no. 2, pp. 95–108, 1983.

[50] R. F. Werneck and M. G. C. Resende, “On the implementation of a swap-based local search procedure for the p-median problem,” *Proc. Fifth Work. Algorithm Eng. Exp. (ALENEX’03).*, pp. 119–127, 2003.

[51] J. E. Beasley, “Lagrangean heuristics for location problems,” *Eur. J. Oper. Res.*, vol. 65, no. 3, pp. 383–399, 1993.

[52] and H. M. S. S. C. Narula, U. I. Ogbu, “Technical Note—An Algorithm for the p-Median Problem.,” *Oper. Res.*, vol. 25, no. 4, pp. 709–713, 1977

[53] N. Mladenovic, J. Moreno, and J. Moreno-Vega, “Tabu Search in solving p-facility location-allocation problems,” *Les Cah. du GERAD, G-95-38*, 1995

[54] E. Rolland, D. Schilling, and J. Current, “An efficient tabu search procedure for the p-median problem,” *Eur. J. Oper. Res.*, vol. 96, no. 2, pp. 329–342, 1997.

[55] E. Goncharov and Y. Kochetov, “Probabilistic tabu search for the unconstrained discrete optimization problems,” *Discret. Anal. Oper. Res.*, vol. 9, no. 2, pp. 13–30, 2002.

[56] P. Hansen and N. Mladenovic, “Variable neighborhood search for the p-median,” *Locat. Sci.*, vol. 5, no. 4, pp. 207–226, 1997.

[57] C. Hosage and M. Goodchild, “Discrete space location-allocation solutions from genetic algorithms,” *Ann. Oper. Res.*, vol. 6, no. 2, pp. 35–46, 1986.

[58] J. Moreno-Pérez, J. García-Roda, and J. Moreno-Vega, “A parallel genetic algorithm for the discrete p-median problem,” *Stud. Locat. Anal.*, pp. 131–141, 1994.

[59] W. Pullan, “A population based hybrid metaheuristic for the p-median problem,” *Evol. Comput. 2008. CEC 2008.(IEEE World Congr. Comput. Intell. IEEE Congr. on. IEEE)*, pp. 1–20, 2008.

[60] F. García-López, B. Melián-Batista, J. a Moreno-Pérez, and J. Marcos Moreno-Vega, “Parallelization of the scatter search for the p-median problem,” *Parallel Comput.*, vol. 29, no. 5, pp. 575–589, 2003.

[61] F. Chiyoshi and R. Galvao, “A statistical analysis of simulated annealing applied to the p-median problem,” *Ann. Oper. Res.*, vol. 96, pp. 61–74, 2000.

[62] T. Levanova and M. Loresh, “Algorithms of ant system and simulated annealing for the p-median problem,” *Autom. Remote Control*, vol. 65, no. 3, pp. 431–438, 2004.

[63] M. Resende and R. Werneck, “A hybrid heuristic for the p-median problem,” *J. heuristics*, pp. 59–88, 2004.

[64] Z. Ren, H. Jiang, J. Xuan, and Z. Luo, “An Accelerated-Limit-Crossing-Based Multilevel Algorithm for the,” vol. 42, no. 4, pp. 1187–1202, 2012.

[65] N. Mladenović, J. Brimberg, P. Hansen, and J. a. Moreno-Pérez, “The p-median problem: A survey of metaheuristic approaches,” *Eur. J. Oper. Res.*, vol. 179, no. 3, pp. 927–939, Jun. 2007.

[66] O. Briant and D. Naddef, “The optimal diversity management problem,” *Oper. Res.*, vol. 52, no. 4, pp. 515–526, 2004.

[67] J. Beasley, “OR-Library: distributing test problems by electronic mail,” *J. Oper. Res. Soc.*, pp. 1069–1072, 1990.

[68] G. Reinelt, “TSPLIB—A traveling salesman problem library,” *ORSA J. Comput.*, vol. 3, no. 4, pp. 376–384, 1991.

[69] R. D. Galvão and C. ReVelle, “A Lagrangean heuristic for the maximal covering location problem,” *Eur. J. Oper. Res.*, vol. 88, no. 1, pp. 114–123, 1996.

[70] E. Senne and L. Lorena, “Lagrangean/surrogate heuristics for p-median problems,” *Comput. Tools Model. Optim. Simul.*, pp. 115–130, 2000.

[71] E. Ruspini, “Numerical methods for fuzzy clustering,” *Inf. Sci. (Ny).*, vol. 2, no. 3, pp. 319–350, 1970.

[72] O. Alp, E. Erkut, and Z. Drezner, “An efficient genetic algorithm for the p-median problem,” *Ann. Oper. Res.*, pp. 21–42, 2003.

[73] M. Maroszek and C. Rettig, “A Tabu Search for the p-Median Problem,” *Tech. Rep.*, 2008.

Anexos

A. Resultados Computacionais – GRASP

Problema	Z^*	Z	Gap	T(s)
pmed1	5819	5819	0,00	0,02
pmed2	4093	4093	0,00	0,03
pmed3	4250	4250	0,00	0,03
pmed4	3034	3034	0,00	0,03
pmed5	1355	1355	0,00	0,05
pmed6	7824	7824	0,00	0,05
pmed7	5631	5631	0,00	0,07
pmed8	4445	4445	0,00	0,12
pmed9	2734	2740	0,22	0,19
pmed10	1255	1255	0,00	0,30
pmed11	7696	7696	0,00	0,13
pmed12	6634	6634	0,00	0,17
pmed13	4374	4374	0,00	0,37
pmed14	2968	2969	0,03	0,67
pmed15	1729	1736	0,40	1,00
pmed16	8162	8162	0,00	0,26
pmed17	6999	6999	0,00	0,32
pmed18	4809	4811	0,04	0,82
pmed19	2845	2849	0,14	1,55
pmed20	1789	1789	0,00	2,41
pmed21	9138	9138	0,00	0,47
pmed22	8579	8579	0,00	0,63
pmed23	4619	4619	0,00	2,03
pmed24	2961	2961	0,00	3,78
pmed25	1828	1832	0,22	5,72
pmed26	9917	9917	0,00	0,87
pmed27	8307	8307	0,00	1,27
pmed28	4498	4502	0,09	5,13
pmed29	3033	3034	0,03	9,81
pmed30	1989	2000	0,55	15,83
pmed31	10086	10086	0,00	1,43
pmed32	9297	9297	0,00	2,02
pmed33	4700	4700	0,00	9,49
pmed34	3013	3021	0,27	18,50
pmed35	10400	10400	0,00	2,07
pmed36	9934	9934	0,00	2,90
pmed37	5057	5057	0,00	14,37
pmed38	11060	11060	0,00	2,77
pmed39	9423	9423	0,00	3,70
pmed40	5128	5134	0,12	20,21

Tabela 1 – Resultados computacionais do GRASP para o conjunto ORLIB

P	Z^*	Z	Gap	T(s)
10	101249,47	101249,55	0,00	6,10
20	57857,55	57857,94	0,00	11,98
30	44013,02	44319,3	0,70	17,78
40	35002,02	35002,52	0,00	22,95
50	29089,71	29212,76	0,42	28,52
60	25160,40	25231,25	0,28	34,09
70	22125,46	22257,91	0,60	39,27
80	19870,28	19952,27	0,41	43,88
90	17987,91	18004,91	0,09	49,70
100	16551,20	16585,3	0,21	54,99
150	12026,41	12114,49	0,73	82,97
200	9355,34	9400,55	0,48	110,49
250	7737,72	7764,78	0,35	139,35
300	6611,60	6654,05	0,64	161,23
350	5719,08	5772,49	0,93	191,68
400	5006,75	5066,79	1,20	216,88
450	4468,29	4507,5	0,88	241,34
500	4046,85	4072,43	0,63	265,17

Tabela 2 – Resultados computacionais do GRASP para a instância TSPLIB – fl1400

P	Z^*	Z	Gap	T(s)
10	9794951,00	9823533,2	0,29	99,25
20	6718848,19	6719026,03	0,00	156,17
30	5374936,14	5405139,77	0,56	228,10
40	4550327,09	4590737,34	0,89	289,42
50	4032379,97	4057854,72	0,63	369,85
60	3642064,70	3653344,88	0,31	448,97
70	3343541,36	3355625,69	0,36	508,07
80	3094451,30	3099979,12	0,18	577,86
90	2893204,11	2905080,8	0,41	647,69
100	2724740,61	2741700,51	0,62	729,21
150	2147756,39	2158732,5	0,51	1056,74
200	1807411,06	1817079,37	0,53	1393,41
250	1569724,89	1579412,58	0,62	167,65
300	1393951,74	1403976,76	0,72	2081,39
350	1256599,00	1263135,39	0,52	2392,80
400	1145271,28	1153497,54	0,72	2754,90
450	1053024,59	1059781,24	0,64	3080,71
500	973920,22	979861,01	0,61	3377,84
600	848217,37	852103,92	0,46	3221,32
700	751972,31	755422,27	0,46	3700,30
800	676724,24	680454,16	0,55	4194,21
900	613308,88	616479,57	0,52	4720,79
1000	558783,56	562801,1	0,72	5012,67

Tabela 3 – Resultados computacionais do GRASP para a instância TSPLIB – rl5934

P	Z^*	Z	Gap	T(s)
10	1213082,03	1213082,03	0,00	22,34
20	840844,53	841752,04	0,11	37,28
30	677272,22	679329,73	0,30	47,44
40	571887,75	575477,3	0,63	65,96
50	507558,16	509177,52	0,32	82,23
60	460771,87	463183,55	0,52	93,06
70	426068,24	428319,16	0,53	108,97
80	397488,90	400310,29	0,71	124,39
90	373241,86	376620,65	0,91	142,91
100	352609,60	355776,48	0,90	155,48
150	281163,12	283218,68	0,73	222,46
200	238344,20	239696,34	0,57	280,19
250	209206,90	210492,07	0,61	348,39
300	187689,40	188942,07	0,67	432,53
350	170919,46	172188,91	0,74	498,53
400	157027,21	158197,9	0,75	552,31
450	145362,91	146452,04	0,75	632,01
500	135447,39	136449,27	0,74	695,70
550	126825,24	127918,6	0,86	738,59
600	119059,77	120333,44	1,07	803,52
650	112017,65	113112,86	0,98	886,89
700	105823,96	107144,66	1,25	937,38
750	100331,25	101573,95	1,24	1002,84
800	95374,17	96428,5	1,11	1045,21
850	90983,64	92059	1,18	1128,84
900	86972,78	87954	1,13	1186,06
950	83267,38	84262,08	1,19	1244,92
1000	79842,01	80595,8	0,94	1115,56

Tabela 4 – Resultados computacionais do GRASP para a instância TSPLIB – pcb3038

Problema	Z^*	Z	Gap	T(s)
sl700	1847	1853	0.32	29.52
sl800	2026	2032	0.30	45.00
sl900	2106	2109	0.14	64.67

Tabela 5 – Resultados computacionais do GRASP para o conjunto SL

P	Z^*	Z	Gap	T(s)
5	779,68	779,68	0,00	0,02
10	512,81	512,81	0,00	0,01
15	389,98	389,98	0,00	0,02
20	314,10	314,47	0,12	0,02
25	252,43	252,43	0,00	0,02
30	199,41	199,41	0,00	0,02
35	159,90	159,91	0,01	0,03
40	127,63	127,63	0,00	0,03

Tabela 6 – Resultados computacionais do GRASP para a instância Ruspini

P	Z^*	Z	Gap	T(s)
5	5703	5703	0,00	0,02
10	4426	4426	0,00	0,02
15	3893	3893	0,00	0,03
20	3565	3565	0,00	0,04
25	3291	3291	0,00	0,04
30	3032	3033	0,03	0,04
40	2542	2542	0,00	0,05
50	2083	2083	0,00	0,05

Tabela 7 – Resultados computacionais do GRASP para a instância gr100

P	Z^*	Z	Gap	T(s)
5	10839	10839	0,00	0,00
10	8729	8729	0,00	0,04
15	7390	7400	0,14	0,05
20	6454	6496	0,65	0,07
25	5875	5903	0,48	0,08
30	5495	5520	0,45	0,09
40	4907	4915	0,16	0,11
50	4374	4382	0,18	0,13

Tabela 8 – Resultados computacionais do GRASP para a instância gr150

B. Resultados Computacionais – Dual RAMP

Problema	Z^*	Z	Gap	T(s)
pmed1	5819	5819	0,00	0,04
pmed2	4093	4093	0,00	0,09
pmed3	4250	4250	0,00	0,08
pmed4	3034	3034	0,00	0,02
pmed5	1355	1355	0,00	0,04
pmed6	7824	7824	0,00	0,24
pmed7	5631	5631	0,00	0,20
pmed8	4445	4445	0,00	0,14
pmed9	2734	2734	0,00	0,48
pmed10	1255	1255	0,00	0,33
pmed11	7696	7696	0,00	0,56
pmed12	6634	6634	0,00	0,64
pmed13	4374	4374	0,00	0,34
pmed14	2968	2968	0,00	1,22
pmed15	1729	1729	0,00	2,16
pmed16	8162	8162	0,00	1,17
pmed17	6999	6999	0,00	1,22
pmed18	4809	4809	0,00	1,03
pmed19	2845	2845	0,00	1,88
pmed20	1789	1789	0,00	4,40
pmed21	9138	9138	0,00	0,43
pmed22	8579	8579	0,00	2,45
pmed23	4619	4619	0,00	2,30
pmed24	2961	2961	0,00	6,01
pmed25	1828	1828	0,00	6,86
pmed26	9917	9917	0,00	3,40
pmed27	8307	8307	0,00	3,88
pmed28	4498	4498	0,00	3,99
pmed29	3033	3033	0,00	6,44
pmed30	1989	1989	0,00	22,01
pmed31	10086	10086	0,00	4,87
pmed32	9297	9297	0,00	5,96
pmed33	4700	4700	0,00	9,72
pmed34	3013	3013	0,00	11,27
pmed35	10400	10400	0,00	6,44
pmed36	9934	9934	0,00	7,12
pmed37	5057	5057	0,00	11,73
pmed38	11060	11060	0,00	8,50
pmed39	9423	9423	0,00	9,48
pmed40	5128	5128	0,00	17,34

Tabela 9 – Resultados computacionais do Dual RAMP para o conjunto ORLIB

P	Z^*	Z	Gap	T(s)
10	101249,47	101249,55	0,00	20,86
20	57857,55	57857,94	0,00	31,85
30	44013,02	44057,96	0,10	44,13
40	35002,02	35002,52	0,00	53,85
50	29089,71	29090,23	0,00	57,38
60	25160,40	25187,18	0,11	59,76
70	22125,46	22126,03	0,00	68,03
80	19870,28	19894,80	0,12	66,53
90	17987,91	17988,60	0,00	80,99
100	16551,20	16555,27	0,02	100,83
150	12026,41	12048,49	0,18	159,82
200	9355,34	9369,06	0,15	172,29
250	7737,72	7755,15	0,23	262,95
300	6611,60	6630,66	0,29	245,50
350	5719,08	5731,51	0,22	221,74
400	5006,75	5006,77	0,00	275,22
450	4468,29	4470,05	0,04	301,84
500	4046,85	4052,07	0,13	331,97

Tabela 10 – Resultados computacionais do Dual RAMP para a instância TSPLIB – fl1400

P	Z^*	Z	Gap	T(s)
10	9794951,00	9797673,65	0,03	651,38
20	6718848,19	6719017,83	0,00	975,21
30	5374936,14	5375040,22	0,00	1131,40
40	4550327,09	4551066,06	0,02	1126,35
50	4032379,97	4034949,86	0,06	1402,99
60	3642064,70	3645146,01	0,08	1954,01
70	3343541,36	3345529,26	0,06	2194,21
80	3094451,30	3095928,63	0,05	2299,47
90	2893204,11	2895817,54	0,09	2677,35
100	2724740,61	2726282,37	0,06	2739,09
150	2147756,39	2148843,72	0,05	3387,51
200	1807411,06	1808209,71	0,04	4053,73
250	1569724,89	1570087,83	0,02	4120,23
300	1393951,74	1394550,19	0,04	4877,05
350	1256599,00	1257090,76	0,04	4935,29
400	1145271,28	1146175,04	0,08	5535,27
450	1053024,59	1053529,61	0,05	5617,80
500	973920,22	974245,08	0,03	5123,98
600	848217,37	848548,56	0,04	6441,36
700	751972,31	752130,74	0,02	6729,11
800	676724,24	676846,26	0,02	7222,86
900	613308,88	613501,67	0,03	8579,17
1000	558783,56	558925,09	0,03	9418,90

Tabela 11 – Resultados computacionais do Dual RAMP para a instância TSPLIB – rl5934

P	Z^*	Z	Gap	T(s)
10	1213082,03	1213082,03	0,00	114,59
20	840844,53	840881,33	0,00	156,02
30	677272,22	677304,91	0,00	211,95
40	571887,75	571887,75	0,00	202,29
50	507558,16	507711,67	0,03	272,55
60	460771,87	460781,89	0,00	313,38
70	426068,24	426068,24	0,00	389,37
80	397488,90	397591,48	0,03	409,56
90	373241,86	373281,20	0,01	359,62
100	352609,60	352650,03	0,01	455,01
150	281163,12	281311,85	0,05	660,22
200	238344,20	238511,06	0,07	614,07
250	209206,90	209235,23	0,01	704,42
300	187689,40	187718,26	0,02	726,68
350	170919,46	170962,48	0,03	751,82
400	157027,21	157056,86	0,02	899,23
450	145362,91	145378,37	0,01	875,55
500	135447,39	135463,53	0,01	1042,90
550	126825,24	126845,31	0,02	1001,08
600	119059,77	119061,33	0,00	1203,91
650	112017,65	112034,39	0,01	1282,45
700	105823,96	105840,65	0,02	1463,36
750	100331,25	100360,72	0,03	1594,57
800	95374,17	95408,84	0,04	1822,30
850	90983,64	91022,54	0,04	2050,80
900	86972,78	86984,84	0,01	2287,57
950	83267,38	83287,89	0,02	2567,52
1000	79842,01	79875,23	0,04	2745,51

Tabela 12 – Resultados computacionais do Dual RAMP para a instância TSPLIB – pcb3038

Problema	Z^*	Z	Gap	T(s)
sl700	1847	1847	0,00	15,58
sl800	2026	2026	0,00	24,69
sl900	2106	2108	0,09	67,12

Tabela 13 – Resultados computacionais do Dual RAMP para o conjunto SL

P	Z^*	Z	Gap	T(s)
5	779,68	779,68	0,00	0,01
10	512,81	512,81	0,00	0,01
15	389,98	389,95	0,00	0,01
20	314,10	314,09	0,00	0,01
25	252,43	252,43	0,00	0,02
30	199,41	199,41	0,00	0,02
35	159,90	159,91	0,01	0,02
40	127,63	127,63	0,00	0,02

Tabela 14 – Resultados computacionais do Dual RAMP para a instância Ruspini

P	Z^*	Z	Gap	T(s)
5	5703	5703	0,00	0,07
10	4426	4426	0,00	0,09
15	3893	3893	0,00	0,09
20	3565	3565	0,00	0,10
25	3291	3291	0,00	0,11
30	3032	3032	0,00	0,10
40	2542	2542	0,00	0,05
50	2083	2083	0,00	0,03

Tabela 15 – Resultados computacionais do Dual RAMP para a instância gr100

P	Z^*	Z	Gap	T(s)
5	10839	10839	0,00	0,15
10	8729	8729	0,00	0,17
15	7390	7390	0,00	0,20
20	6454	6462	0,12	0,22
25	5875	5875	0,00	0,26
30	5495	5495	0,00	0,30
40	4907	4908	0,02	0,27
50	4374	4374	0,00	0,29

Tabela 16 – Resultados computacionais do Dual RAMP para a instância gr150

C. Resultados Computacionais – Pesquisa por Dispersão

Problema	Z^*	Z	Gap	T(s)
pmed1	5819	5819	0,00	0,03
pmed2	4093	4093	0,00	0,04
pmed3	4250	4250	0,00	0,05
pmed4	3034	3034	0,00	0,06
pmed5	1355	1355	0,00	0,07
pmed6	7824	7824	0,00	0,08
pmed7	5631	5631	0,00	0,11
pmed8	4445	4445	0,00	0,23
pmed9	2734	2734	0,00	0,62
pmed10	1255	1255	0,00	1,08
pmed11	7696	7696	0,00	0,20
pmed12	6634	6634	0,00	0,21
pmed13	4374	4374	0,00	0,59
pmed14	2968	2968	0,00	1,78
pmed15	1729	1735	0,35	4,70
pmed16	8162	8162	0,00	0,41
pmed17	6999	6999	0,00	0,48
pmed18	4809	4809	0,00	1,96
pmed19	2845	2845	0,00	6,03
pmed20	1789	1789	0,00	12,72
pmed21	9138	9138	0,00	0,60
pmed22	8579	8579	0,00	0,95
pmed23	4619	4619	0,00	2,85
pmed24	2961	2961	0,00	15,91
pmed25	1828	1829	0,05	16,00
pmed26	9917	9917	0,00	1,21
pmed27	8307	8307	0,00	1,61
pmed28	4498	4498	0,00	10,06
pmed29	3033	3034	0,03	29,64
pmed30	1989	1990	0,05	67,51
pmed31	10086	10086	0,00	1,88
pmed32	9297	9297	0,00	2,48
pmed33	4700	4700	0,00	24,69
pmed34	3013	3013	0,00	45,28
pmed35	10400	10400	0,00	2,56
pmed36	9934	9934	0,00	4,23
pmed37	5057	5057	0,00	24,45
pmed38	11060	11060	0,00	3,36
pmed39	9423	9423	0,00	4,52
pmed40	5128	5129	0,02	34,55

Tabela 17 – Resultados computacionais da Pesquisa por Dispersão para o conjunto ORLIB

P	Z^*	Z	Gap	T(s)
10	101249,47	101249,55	0,00	7,28
20	57857,55	57857,94	0,00	14,26
30	44013,02	44013,48	0,00	41,43
40	35002,02	35002,52	0,00	38,09
50	29089,71	29090,23	0,00	56,87
60	25160,40	25161,12	0,00	108,83
70	22125,46	22151,99	0,12	73,43
80	19870,28	19872,72	0,01	117,24
90	17987,91	17988,60	0,00	154,75
100	16551,20	16553,07	0,01	155,03
150	12026,41	12036,67	0,09	464,91
200	9355,34	9359,99	0,05	585,94
250	7737,72	7744,01	0,08	970,46
300	6611,60	6622,56	0,17	1066,41
350	5719,08	5731,69	0,22	1233,12
400	5006,75	5025,79	0,38	1833,62
450	4468,29	4486,77	0,41	2136,40
500	4046,85	4056,74	0,24	1826,60

Tabela 18 – Resultados computacionais da Pesquisa por Dispersão para a instância TSPLIB – fl1400

P	Z^*	Z	Gap	T(s)
10	9794951,00	9797675,62	0,03	253,04
20	6718848,19	6718848,19	0,00	1054,55
30	5374936,14	5375518,70	0,01	1811,60
40	4550327,09	4550439,50	0,00	2071,58
50	4032379,97	4032496,88	0,00	1799,40
60	3642064,70	3642548,97	0,01	4583,96
70	3343541,36	3345171,09	0,05	3849,94
80	3094451,30	3094754,16	0,01	4124,60
90	2893204,11	2895699,16	0,09	3748,85
100	2724740,61	2726784,89	0,08	10024,39
150	2147756,39	2149285,67	0,07	10329,50
200	1807411,06	1808783,02	0,08	15113,07
250	1569724,89	1571360,85	0,10	16718,86
300	1393951,74	1394626,24	0,05	12594,56
350	1256599,00	1257671,67	0,09	14708,38
400	1145271,28	1146777,88	0,13	17308,99
450	1053024,59	1054688,28	0,16	18787,12
500	973920,22	974973,34	0,11	15616,26
600	848217,37	848648,76	0,05	20707,27
700	751972,31	753056,85	0,14	16370,48
800	676724,24	677606,16	0,13	21974,66
900	613308,88	614228,78	0,15	26248,06
1000	558783,56	559502,26	0,13	29347,27

Tabela 19 – Resultados computacionais da Pesquisa por Dispersão para a instância TSPLIB – rl5934

P	Z^*	Z	Gap	T(s)
10	1213082,03	1213082,03	0,00	27,71
20	840844,53	840844,53	0,00	210,50
30	677272,22	677272,22	0,00	418,92
40	571887,75	571887,75	0,00	528,40
50	507558,16	507688,82	0,03	754,59
60	460771,87	460906,76	0,03	1239,57
70	426068,24	426252,91	0,04	1101,35
80	397488,90	397540,76	0,01	1077,86
90	373241,86	373580,81	0,09	1793,57
100	352609,60	352753,59	0,04	1938,83
150	281163,12	281314,54	0,05	1698,38
200	238344,20	238467,68	0,05	280,19
250	209206,90	209417,37	0,10	3232,37
300	187689,40	187760,69	0,04	2850,70
350	170919,46	171131,77	0,12	4279,66
400	157027,21	157401,95	0,24	4302,36
450	145362,91	145567,13	0,14	3874,35
500	135447,39	135728,46	0,21	4540,14
550	126825,24	127169,90	0,27	5742,19
600	119059,77	119415,40	0,30	6459,20
650	112017,65	112443,35	0,38	8297,12
700	105823,96	106198,81	0,35	7996,12
750	100331,25	100706,10	0,37	9978,71
800	95374,17	95736,81	0,38	10279,93
850	90983,64	91316,18	0,37	12364,35
900	86972,78	87333,63	0,41	12227,81
950	83267,38	83524,44	0,31	13740,47
1000	79842,01	80056,89	0,27	12940,51

Tabela 20 – Resultados computacionais da Pesquisa por Dispersão para a instância TSPLIB – pcb3038

Problema	Z^*	Z	Gap	T(s)
sl700	1847	1847	0,00	122,07
sl800	2026	2026	0,00	132,08
sl900	2106	2107	0,05	214,82

Tabela 21 – Resultados computacionais da Pesquisa por Dispersão para o conjunto SL

P	Z^*	Z	Gap	T(s)
5	779,68	779,68	0,00	0,01
10	512,81	512,81	0,00	0,02
15	389,98	389,95	0,00	0,03
20	314,10	314,09	0,00	0,05
25	252,43	252,43	0,00	0,07
30	199,41	199,41	0,00	0,06
35	159,90	159,91	0,01	0,08
40	127,63	127,63	0,00	0,09

Tabela 22 – Resultados computacionais da Pesquisa por Dispersão para a instância Ruspini

P	Z^*	Z	Gap	T(s)
5	5703	5703,00	0,00	0,02
10	4426	4426,00	0,00	0,05
15	3893	3893,00	0,00	0,12
20	3565	3565,00	0,00	0,12
25	3291	3291,00	0,00	0,14
30	3032	3032,00	0,00	0,10
40	2542	2542,00	0,00	0,12
50	2083	2083,00	0,00	0,14

Tabela 23 – Resultados computacionais da Pesquisa por Dispersão para a instância gr100

P	Z^*	Z	Gap	T(s)
5	10839	10839,00	0,00	0,08
10	8729	8729,00	0,00	0,11
15	7390	7390,00	0,00	0,20
20	6454	6454,00	0,00	0,26
25	5875	5875,00	0,00	0,43
30	5495	5495,00	0,00	0,53
40	4907	4909,00	0,04	0,60
50	4374	4374,00	0,00	0,81

Tabela 24 – Resultados computacionais da Pesquisa por Dispersão para a instância gr150

D. Resultados Computacionais – PD-RAMP

Problema	Z^*	Z	Gap	T(s)
pmed1	5819	5819	0,00	0,06
pmed2	4093	4093	0,00	0,09
pmed3	4250	4250	0,00	0,09
pmed4	3034	3034	0,00	0,04
pmed5	1355	1355	0,00	0,05
pmed6	7824	7824	0,00	0,29
pmed7	5631	5631	0,00	0,37
pmed8	4445	4445	0,00	0,39
pmed9	2734	2734	0,00	0,77
pmed10	1255	1255	0,00	1,11
pmed11	7696	7696	0,00	0,70
pmed12	6634	6634	0,00	0,76
pmed13	4374	4374	0,00	1,54
pmed14	2968	2968	0,00	1,63
pmed15	1729	1729	0,00	3,62
pmed16	8162	8162	0,00	1,25
pmed17	6999	6999	0,00	1,61
pmed18	4809	4809	0,00	2,76
pmed19	2845	2845	0,00	4,51
pmed20	1789	1789	0,00	12,46
pmed21	9138	9138	0,00	0,57
pmed22	8579	8579	0,00	2,93
pmed23	4619	4619	0,00	3,74
pmed24	2961	2961	0,00	10,98
pmed25	1828	1828	0,00	29,29
pmed26	9917	9917	0,00	4,20
pmed27	8307	8307	0,00	4,53
pmed28	4498	4498	0,00	12,23
pmed29	3033	3033	0,00	19,07
pmed30	1989	1989	0,00	67,42
pmed31	10086	10086	0,00	5,53
pmed32	9297	9297	0,00	7,59
pmed33	4700	4700	0,00	18,68
pmed34	3013	3013	0,00	36,24
pmed35	10400	10400	0,00	7,79
pmed36	9934	9934	0,00	8,96
pmed37	5057	5057	0,00	28,71
pmed38	11060	11060	0,00	9,57
pmed39	9423	9423	0,00	11,54
pmed40	5128	5128	0,00	43,39

Tabela 25 – Resultados computacionais do PD-RAMP para o conjunto ORLIB

P	Z^*	Z	Gap	T(s)
10	101249,47	101249,55	0,00	25,83
20	57857,55	57857,94	0,00	42,28
30	44013,02	44013,48	0,00	63,75
40	35002,02	35002,52	0,00	79,61
50	29089,71	29090,22	0,00	87,24
60	25160,40	25161,12	0,00	138,20
70	22125,46	22126,03	0,00	138,69
80	19870,28	19874,84	0,02	175,84
90	17987,91	17988,60	0,00	192,36
100	16551,20	16552,24	0,01	200,08
150	12026,41	12029,80	0,03	530,97
200	9355,34	9364,25	0,10	743,72
250	7737,72	7745,01	0,09	1023,80
300	6611,60	6623,60	0,18	1021,60
350	5719,08	5724,76	0,10	1415,46
400	5006,75	5010,40	0,07	1558,23
450	4468,29	4468,34	0,00	2745,87
500	4046,85	4052,20	0,13	2698,48

Tabela 26 – Resultados computacionais do PD-RAMP para a instância TSPLIB – fl1400

P	Z^*	Z	Gap	T(s)
10	9794951,00	9794973,65	0,00	1100,08
20	6718848,19	6718848,17	0,00	1250,64
30	5374936,14	5374936,14	0,00	2169,60
40	4550327,09	4550327,09	0,00	2878,83
50	4032379,97	4032538,93	0,00	2840,10
60	3642064,70	3642517,31	0,01	5639,14
70	3343541,36	3344412,87	0,03	6937,64
80	3094451,30	3094782,27	0,01	7332,90
90	2893204,11	2895573,95	0,08	8125,27
100	2724740,61	2725238,97	0,02	8600,28
150	2147756,39	2148482,31	0,03	10971,70
200	1807411,06	1807654,71	0,01	14876,63
250	1569724,89	1569789,77	0,00	18973,80
300	1393951,74	1394023,76	0,01	16403,12
350	1256599,00	1257241,14	0,05	14633,07
400	1145271,28	1145534,53	0,02	24934,01
450	1053024,59	1053406,80	0,04	24196,52
500	973920,22	974157,04	0,02	24031,41
600	848217,37	848421,08	0,02	27708,10
700	751972,31	751981,88	0,00	32267,62
800	676724,24	676913,08	0,03	24113,18
900	613308,88	613390,76	0,01	27249,78
1000	558783,56	558867,21	0,01	38003,41

Tabela 27 – Resultados computacionais do PD-RAMP para a instância TSPLIB – rl5934

P	Z^*	Z	Gap	T(s)
10	1213082,03	1213082,03	0,00	163,58
20	840844,53	840844,53	0,00	344,50
30	677272,22	677272,21	0,00	530,45
40	571887,75	571887,75	0,00	662,12
50	507558,16	507558,16	0,00	884,70
60	460771,87	460771,87	0,00	1356,51
70	426068,24	426068,24	0,00	1480,09
80	397488,90	397530,27	0,01	1412,57
90	373241,86	373241,86	0,00	2634,56
100	352609,60	352643,99	0,01	1644,89
150	281163,12	281225,14	0,02	2414,48
200	238344,20	238455,10	0,05	3601,97
250	209206,90	209216,93	0,00	3728,02
300	187689,40	187696,47	0,00	3749,96
350	170919,46	170941,60	0,01	5248,02
400	157027,21	157062,40	0,02	3222,76
450	145362,91	145368,98	0,00	5032,22
500	135447,39	135461,05	0,01	5372,58
550	126825,24	126831,96	0,01	7258,78
600	119059,77	119064,12	0,00	8354,45
650	112017,65	112034,25	0,01	10299,99
700	105823,96	105828,92	0,00	9945,80
750	100331,25	100340,68	0,01	11902,62
800	95374,17	95390,01	0,02	12764,17
850	90983,64	90995,87	0,01	15863,06
900	86972,78	86977,10	0,00	18389,67
950	83267,38	83277,82	0,01	15520,13
1000	79842,01	79860,60	0,02	14651,42

Tabela 28 – Resultados computacionais do PD-RAMP para a instância TSPLIB – pcb3038

Problema	Z^*	Z	Gap	T(s)
sl700	1847	1847	0,00	39,61
sl800	2026	2026	0,00	76,98
sl900	2106	2106	0,00	253,41

Tabela 29 – Resultados computacionais do PD-RAMP para o conjunto SL

P	Z^*	Z	Gap	T(s)
5	779,68	779,68	0,00	0,01
10	512,81	512,81	0,00	0,02
15	389,98	389,98	0,00	0,02
20	314,10	314,10	0,00	0,02
25	252,43	252,43	0,00	0,03
30	199,41	199,41	0,00	0,03
35	159,90	159,90	0,00	0,03
40	127,63	127,63	0,00	0,03

Tabela 30 – Resultados computacionais do PD-RAMP para a instância Ruspini

P	Z^*	Z	Gap	T(s)
5	5703	5703	0,00	0,08
10	4426	4426	0,00	0,11
15	3893	3893	0,00	0,16
20	3565	3565	0,00	0,16
25	3291	3291	0,00	0,22
30	3032	3032	0,00	0,20
40	2542	2542	0,00	0,14
50	2083	2083	0,00	0,13

Tabela 31 – Resultados computacionais do PD-RAMP para a instância gr100

P	Z^*	Z	Gap	T(s)
5	10839	10839	0,00	0,20
10	8729	8729	0,00	0,24
15	7390	7390	0,00	0,28
20	6454	6454	0,00	0,46
25	5875	5875	0,00	0,71
30	5495	5495	0,00	0,82
40	4907	4907	0,00	0,92
50	4374	4374	0,00	1,25

Tabela 32 – Resultados computacionais do PD-RAMP para a instância gr150