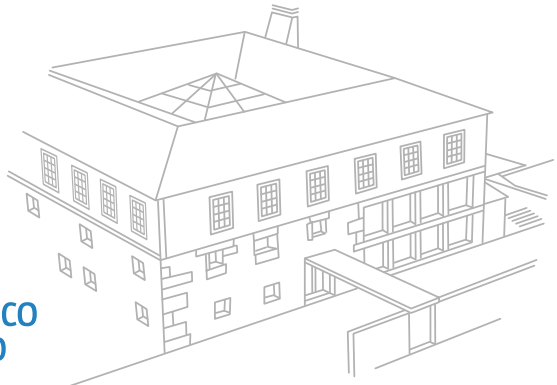

ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Algoritmos RAMP para o Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor

DESIGNAÇÃO DO MESTRADO: Mestrado em Engenharia Informática

AUTOR: Óscar António Maia de Oliveira

ORIENTADOR(ES): Dorabela Regina Chiote Ferreira Gamboa

ANO: 2012

Para os meus filhos, Rúben e Martim, espero vir a compensar todo o tempo que não vos pude dedicar para concretizar este projeto…

Agradecimentos

À minha esposa, agradeço de forma muito profunda todo o apoio e compreensão.

Aos meus pais, sem o apoio deles não teria sido possível concretizar mais esta etapa da minha vida.

À minha orientadora, Professora Doutora Dorabela Gamboa, pelo apoio e constante motivação, mas principalmente por acreditar que eu seria capaz de finalizar este projeto mesmo quando eu tinha deixado de acreditar.

Aos meus colegas do Centro de Inovação e Investigação em Ciências Empresariais e Sistemas de Informação (CIICESI), por proporcionarem um local de trabalho onde impera a cooperação e a boa disposição.

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

Resumo

Os Problemas de Localização de Instalações são problemas de otimização combinatória complexos que têm centrado a atenção da comunidade científica. A importância dada à resolução destes problemas deve-se principalmente à sua relevância nas mais variadas áreas, tais como, economia, indústria, saúde, entre muitas outras.

Neste estudo é considerado o Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor (*Single Source Capacitated Facility Location Problem* - SSCFLP). No SSCFLP, dado um conjunto de possíveis localizações para a abertura de instalações e um conjunto de clientes a servir, o objetivo é determinar que instalações abrir de forma a satisfazer com custo mínimo a procura dos clientes, garantindo que cada cliente é servido apenas por uma instalação. Neste problema são considerados os custos de abertura das instalações e os custos de afetação dos clientes. O SSCFLP tem várias aplicações práticas, como por exemplo, no planeamento de sistemas de distribuição e na conceção de redes informáticas.

Os métodos exatos conseguem garantir a obtenção da solução ótima dos problemas à custa de recursos computacionais elevados, tornando pertinente a investigação de abordagens alternativas, nomeadamente heurísticas/metaheurísticas, que permitam com recursos mais reduzidos, a obtenção de soluções de elevada qualidade.

As heurísticas/metaheurísticas têm centrado a sua atenção apenas num dos lados do espaço de soluções dos problemas de otimização combinatória. A dualidade dos problemas tem sido, maioritariamente, utilizada para a criação de soluções iniciais para uma exploração mais intensiva do espaço de soluções por parte de heurísticas primais.

A metaheurística RAMP (*Relaxation Adaptive Memory Programming*), proposta por Rego [1], pretende criar algoritmos que explorem de forma mais eficiente a relação primal-dual dos problemas de otimização combinatória, permitindo, de forma iterativa, a manipulação da informação que é obtida de ambos os lados do espaço de soluções.

A aplicação do método RAMP a vários problemas de otimização combinatória, demonstrou a enorme potencialidade desta metaheurística, obtendo algoritmos de estado-da-arte para todos esses problemas.

O objetivo deste trabalho é verificar se a aplicação do método RAMP ao SSCFLP também é capaz de rivalizar com outros métodos propostos para a resolução deste problema.

Neste trabalho, são apresentados dois novos algoritmos para a resolução do SSCFLP, ambos baseados no método RAMP, que designamos por Dual RAMP e PD-RAMP.

O primeiro algoritmo (Dual RAMP) segue a abordagem RAMP na sua versão mais simples. O Dual RAMP baseia-se na resolução do dual lagrangeano do SSCFLP, através de otimização por subgradiente. A solução dual é projetada para o espaço de soluções primal através da aplicação de um método simples de projeção, e a solução primal obtida é sujeita a um método de melhoramento baseado numa abordagem simples da pesquisa tabu. Iterativamente, a informação obtida do lado primal é utilizada para o ajuste dos parâmetros do dual.

O segundo algoritmo (PD-RAMP) baseia-se numa versão mais sofisticada da abordagem RAMP. Este algoritmo integra o Dual RAMP com um método evolutivo de forma a fortalecer a relação primal-dual do problema. Na implementação proposta, o método primal do PD-RAMP é baseado numa pesquisa por dispersão com um conjunto de referência atualizado por ambos os lados, primal e dual.

Os resultados obtidos pelo Dual RAMP e pelo PD-RAMP permitem concluir que a aplicação da metaheurística RAMP ao SSCFLP consegue resultados excelentes, obtendo soluções de elevada qualidade em tempos computacionais reduzidos. Acresce ainda o facto de, ao contrário da maioria das abordagens existentes na literatura, ambos os algoritmos propostos demonstrarem ser extremamente robustos, conseguindo muito bons resultados para todos os conjuntos de testes utilizados.

Palavras-chave: Problemas de localização, SSCFLP, Dual RAMP, PD-RAMP.

Abstract

Facility Location Problems are complex combinatorial optimization problems that have been focusing the attention of the scientific community. The importance given to the solution of these problems, is mainly due to their relevance in diversified areas, such as, economics, industry, health, among many others.

This study considers the Single Source Capacitated Facility Location Problem (SSCFLP), where, given a set of possible locations for opening facilities and a set of clients to serve, the goal is to determine which facilities to open in order to fulfill with minimum cost the demand of the clients, ensuring that each client is served by only one facility. This problem considers the costs for opening facilities and the client's assignment costs. SSCFLP has several practical applications, such as, distribution systems planning and computer networks design.

Exact methods ensure the achievement of the problem's optimal solution at the expense of high computational resources, justifying the exploration of alternative approaches, such as heuristics/metaheuristics, that can obtain high quality solutions with lower resources.

Heuristics/metaheuristics have focused their attention on only one side of the combinatorial optimization problems solution space. The problems duality has been mostly used for creating initial solutions for a more intensive exploration of the solution space by primal heuristics.

The RAMP (Relaxation Adaptive Memory Programming) metaheuristic proposed by Rego [1] aims to create algorithms that exploit more efficiently the primal-dual relationship of combinatorial optimization problems, allowing iteratively, the manipulation of information that is obtained by both sides of the solutions space.

The RAMP application to several combinatorial optimization problems, demonstrated the great potential of this metaheuristic, obtaining state-of-the-art algorithms for all of those problems.

With this study we intend to verify if the application of the RAMP method to the SSCFLP is also capable of competing with other proposed methods for the solution of this problem.

In this work, we present two new algorithms for solving the SSCFLP, both based on the RAMP method, designated by Dual RAMP and PD-RAMP.

The first algorithm (Dual RAMP) follows the RAMP approach in its simplest version. The Dual RAMP is based on the solution of the Lagrangean dual through subgradient optimization. The dual solution is projected to the primal solution space through the application of a simple projection method, and the obtained solution is subjected to an improvement method based on a simple tabu search approach. Iteratively, the information obtained from the primal side is used to adjust the dual parameters.

The second algorithm (PD-RAMP) is based on a more sophisticated version of the RAMP approach. This algorithm integrates the Dual RAMP algorithm with an evolutionary method in order to strengthen the primal-dual relationship of the problem. In the proposed implementation, the PD-RAMP primal method is based on Scatter Search with a reference set updated by both sides, primal and dual.

The results obtained by the Dual RAMP and the PD-RAMP algorithms showed that the application of the RAMP metaheuristic to the SSCFLP attains excellent results, obtaining high quality solutions in reduced computational times. Moreover, unlike most of the existing approaches in the literature, both proposed algorithms proved to be extremely robust, achieving very good results for all sets of tests.

Keywords: Location problems, SSCFLP, Dual RAMP, PD-RAMP.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

O estudo dos problemas de localização de instalações (*Facility Location Problems*) foi iniciado, em 1909, por Alfred Weber [2], que analisou a localização de uma fábrica em relação a duas fontes de matéria-prima e um mercado consumidor, de modo a minimizar os custos de transporte.

Os problemas de localização de instalações são caracterizados pela existência de um conjunto de localizações onde é possível abrir instalações (ou centros de oferta) com o objetivo de satisfazer a procura de um conjunto de clientes (ou centros de procura). A resolução destes problemas consiste em determinar as localizações onde abrir as instalações, de modo a satisfazer todos os clientes, tendo um determinado objetivo, como por exemplo, a minimização dos custos (custos de abertura de instalações, custos de transporte, entre outros). Estes problemas podem estar sujeitos a restrições, tal como a quantidade máxima de procura que as instalações podem satisfazer. Dependendo do objetivo do problema e das restrições impostas, os problemas de localização de instalações apresentam um vasto conjunto de variantes.

Com ampla aplicação na logística, como por exemplo, no apoio à tomada de decisão para a localização de novas unidades fabris, estes problemas têm visto a sua aplicabilidade alargada a outras áreas, tais como, sistemas de redes, unidades de saúde, entre outras. Vários exemplos de aplicação de problemas de localização de instalações podem ser encontrados em Current *et al.* [3].

Os problemas de localização de instalações fazem parte de um grupo de problemas de otimização combinatória de difícil resolução (NP-difíceis) [4], não sendo conhecidos, para estes problemas, algoritmos que garantam a obtenção da solução ótima, para todas as instâncias do problema, em tempo polinomial.

Os métodos de resolução exatos, embora garantam a obtenção da solução ótima (a melhor solução possível) de um problema, são usualmente muito exigentes nos recursos computacionais utilizados. Por outro lado, os métodos heurísticos, embora não garantam a resolução dos problemas à otimalidade, são geralmente menos exigentes nos recursos

computacionais, e têm apresentado uma crescente qualidade nos resultados obtidos, fomentando o surgimento de uma grande diversidade de métodos, principalmente metaheurísticas, para a resolução de problemas complexos de otimização combinatória.

As metaheurísticas referem-se a metodologias gerais de resolução de problemas de otimização combinatória que são facilmente adaptáveis a problemas específicos e que permitem criar heurísticas que explorem mais eficientemente o espaço de soluções.

As metaheurísticas têm focado a sua atenção apenas num dos lados do espaço de soluções dos problemas de otimização combinatória, por exemplo, a pesquisa por dispersão (*Scatter Search*) explora apenas o espaço de soluções primal (o problema original) e as heurísticas lagrangeanas exploram principalmente o lado dual (espaço de soluções conseguido através da relaxação do problema original). Tendo em conta esta observação, Rego [1] propõe uma metaheurística, denominada de RAMP (*Relaxation Adaptive Memory Programming*), que explora de forma eficiente a relação entre os lados primal e dual de um problema de otimização combinatória, utilizando princípios de memória adaptativa para orientar a pesquisa.

A abordagem RAMP demonstrou ser um método capaz de produzir resultados de excelente qualidade para vários problemas de otimização combinatória, como são exemplo o Problema de Ordenação Linear (*Linear Ordering Problem* - LOP), o Problema de Localização de Instalações sem Restrições de Capacidade (*Uncapacitated Facility Location Problem* - UFLP) [5], o Problema da Árvore de Expansão Mínima com Restrições de Capacidade (*Capacitated Minimum Spanning Tree* - CMST) [6], e o Problema do Sequenciamento de Projetos com Recursos Limitados (*Resource Constraint Project Scheduling Problem* - RCPSP) [7].

O problema escolhido neste estudo, é o Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor (*Single Source Capacitated Facility Location Problem* - SSCFLP). Neste problema, temos um conjunto de possíveis localizações para a abertura de instalações, e o objetivo é, garantindo que cada cliente é servido apenas por uma instalação, determinar as instalações a abrir de forma a satisfazer com custo mínimo, o conjunto de clientes. Os custos a minimizar são os custos de abertura das instalações e os custos de transporte entre a instalação e o cliente, também designados por custos de afetação.

O Problema de Localização de Instalações com Restrições de Capacidade (*Capacitated Facility Location Problem* - CFLP) pertence à classe dos problemas NP-difíceis [8] e o SSCFLP é um caso especial deste problema, em os clientes podem apenas ser servidos por

uma instalação. O SSCFLP é, geralmente, de mais difícil resolução pois todas as variáveis de decisão são binárias (no caso do CFLP, as variáveis de decisão relativas à afetação dos clientes são contínuas).

O objetivo deste estudo é verificar se a aplicação da metaheurística RAMP ao SSCFLP produz resultados de qualidade semelhante aos já obtidos com este método.

A dissertação está estruturada da seguinte forma.

- No capítulo 2, faz-se uma breve descrição dos métodos para a resolução de problemas de otimização combinatória, com especial ênfase nas metaheurísticas, nomeadamente, o *Greedy Randomized Adaptive Search Procedure* (GRASP), a pesquisa tabu (*Tabu Search*), a pesquisa por dispersão e o RAMP. No final deste capítulo, é feita uma apresentação sucinta de algumas técnicas de relaxação matemática.
- No capítulo 3, são apresentados os problemas de localização de instalações, com principal foco no problema em estudo, o SSCFLP, sendo também elencados os algoritmos propostos por diversos autores para a resolução deste problema. Segue-se a descrição dos algoritmos RAMP propostos para a resolução do SSCFLP (Dual RAMP e PD-RAMP), finalizando o capítulo com a análise dos resultados computacionais obtidos.
- Finalmente, o capítulo 4 apresenta as conclusões retiradas deste estudo e as propostas de trabalho futuro.

2. Métodos para a Resolução de Problemas de Otimização Combinatória

Os problemas de otimização combinatória podem ser representados através de modelos matemáticos que apresentam o objetivo do problema, os recursos, as restrições e as variáveis de decisão.

A otimização combinatória é o processo de se encontrar a melhor solução de entre um conjunto de soluções possíveis para o problema, e envolve determinar a configuração para variáveis de decisão, tendo em conta os recursos e as restrições impostas pelo modelo, que permita atingir o melhor resultado possível.

A aplicabilidade prática dos problemas de otimização combinatória é bem saliente no nosso quotidiano como, por exemplo, planeamento de rotas, planeamento de produção, embalamento e empacotamento, processamento de imagens, localização de instalações, entre muitos outros.

Pela importância que os problemas de otimização combinatória apresentam nas mais diversas áreas (economia, indústria, medicina, etc.), a conceção de algoritmos que permitam a obtenção de resultados de elevada qualidade, em tempos computacionais aceitáveis, tem sido alvo de crescente investigação.

De seguida são apresentados, de forma resumida, os métodos de resolução de problemas de otimização combinatória (remetemos para El-Ghazali [9] para uma exposição mais aprofundada). Nesta secção, são igualmente apresentadas as metaheurísticas relevantes para este estudo, assim como, algumas técnicas de relaxação matemática.

2.1. Métodos Exatos e Métodos Aproximados

Os métodos de resolução de problemas de otimização combinatória podem ser divididos em duas categorias: métodos exatos e métodos aproximados.

Os métodos exatos garantem a obtenção da solução ótima para qualquer instância de um problema de otimização combinatória. A obtenção da solução ótima é, usualmente, garantida à custa recursos computacionais elevados mesmo para instâncias de pequena e média dimensão. A procura da solução ótima é feita através de enumeração, em alguns casos de forma implícita, de todo o espaço de soluções. De entre os métodos exatos mais utilizados podem-se referir os algoritmos de enumeração implícita baseados em busca em árvore como, por exemplo, o *Branch and Bound*.

Os métodos exatos garantem a obtenção da solução ótima, apenas se não existir um limite temporal para a sua execução e se forem assegurados os recursos computacionais necessários, o que para instâncias de grande dimensão pode ser totalmente impraticável. Destas condições, surge a necessidade de métodos que possam, com recursos computacionais menos exigentes e em tempos aceitáveis, obter soluções de qualidade.

Os métodos aproximados não garantem a obtenção da solução ótima para problemas de otimização combinatória mas, geralmente, obtêm soluções de boa qualidade com recursos computacionais bastante inferiores aos exigidos pelos métodos exatos.

Os métodos aproximados são desenvolvidos para a resolução específica de um problema e podem ser divididos em duas classes distintas, os algoritmos de aproximação e as heurísticas.

Os algoritmos de aproximação não garantem a obtenção da solução ótima para um problema, mas garantem que a solução se encontra dentro de um limite de qualidade ou de recursos computacionais requeridos.

As heurísticas são, geralmente, menos exigentes em recursos computacionais que os algoritmos de aproximação, mas não garantem que a solução obtida esteja dentro de um intervalo de qualidade ou de recursos computacionais necessários. As heurísticas podem ser classificadas como construtivas, de pesquisa local ou metaheurísticas.

As heurísticas construtivas geram uma solução adicionando iterativamente um elemento à solução até ser gerada uma solução admissível. No caso das heurísticas gulosas (*greedy heuristics*) a escolha do próximo elemento a inserir na solução é feita pelo que mais favorece o objetivo do problema no momento. As heurísticas construtivas são normalmente utilizadas para criar soluções iniciais a utilizar por outros métodos (por exemplo: pesquisa local).

As heurísticas de pesquisa local (ou de melhoramento) partem de uma solução inicial e, em cada iteração, através da exploração do espaço de soluções vizinhas, tentam encontrar uma

solução que seja melhor que a corrente. A pesquisa termina quando numa determinada iteração não é possível melhorar a solução corrente com a estrutura de vizinhança[1] utilizada.

A Figura 1, apresenta o algoritmo genérico de pesquisa local para um problema de minimização, representando $N(S_c)$ a vizinhança da solução corrente.

1. Obter uma solução inicial admissível, S_0, cujo valor para a função objetivo é $f(S_0)$
2. Inicializar a solução corrente S_c, fazendo $S_c = S_0$
3. Considerar uma solução $S_v \in N(S_c)$
4. Se $f(S_v) < f(S_c)$ então
5. $\quad S_c = S_v$
6. Senão
7. $\quad N(S_c) = N(S_c) - S_v$
8. Se $N(S_c) \neq \emptyset$ então
9. $\quad$ Ir para ao passo 3

Figura 1 - Algoritmo genérico de pesquisa local

As metaheurísticas referem-se a metodologias gerais de resolução de problemas de otimização combinatória que são facilmente adaptáveis a problemas específicos e que permitem criar heurísticas que explorem mais eficientemente o espaço de soluções.

2.2. Metaheurísticas

As heurísticas mais simples são desenvolvidas baseando-se na criação de uma solução inicial e no recurso a um método de pesquisa local, de forma a melhorar a solução corrente até ser encontrado um ótimo local[2]. Esta forma de pesquisa tem como defeito a incapacidade de escapar a ótimos locais, não explorando de forma mais alargada o espaço de soluções.

Blum e Roli [10] caracterizam as metaheurísticas da seguinte forma:

- São estratégias que guiam o processo de pesquisa;
- O objetivo é explorar eficientemente o espaço de soluções de forma a encontrar soluções (quase) ótimas;

[1] Assumindo S como o espaço de soluções admissíveis de um problema P e $s \in S$, chama-se vizinhança de s, $N(s)$, ao conjunto de soluções $N(s) \subseteq S$ que é possível alcançar através de um movimento (especifico para cada estrutura de vizinhança). Cada solução do conjunto $N(s)$ é denominada de solução vizinha de s.

[2] Denomina-se ótimo global à solução ótima de um problema e ótimo local à melhor solução que se consegue obter com uma determinada estrutura de vizinhança.

- As técnicas que constituem algoritmos metaheurísticos podem ser desde simples procedimentos de pesquisa local até processos de aprendizagem complexos;
- São algoritmos aproximados e usualmente não determinísticos;
- Podem incorporar mecanismos para evitar ficar preso a áreas confinadas do espaço de soluções;
- Não são específicas para a resolução de um problema;
- Metaheurísticas mais avançadas podem incorporar alguma forma de memória para orientar a pesquisa.

As metaheurísticas podem ser classificadas segundo diversos critérios, por exemplo, se são dotadas de estruturas de memória, se são inspiradas na natureza, se são baseadas em populações de soluções, entre outros. Remetemos para Blum e Roli [10] e Sirenko [11] para diversas classificações de metaheurísticas.

Nas metaheurísticas são muito importantes os conceitos de diversificação e intensificação da pesquisa tal com o seu correto balanceamento.

A diversificação da pesquisa permite que o espaço de soluções a explorar não seja confinado apenas a zonas na vizinhança de soluções entendidas como mais promissoras.

Por outro lado, a intensificação da pesquisa, refere-se a uma exploração mais profunda de zonas de vizinhança de soluções consideradas como mais promissoras.

De seguida, apresentam-se as metaheurísticas mais relevantes para este estudo, nomeadamente, o procedimento GRASP (*Greedy Randomized Adaptive Search Procedure*), a pesquisa tabu (*Tabu Search*), a pesquisa por dispersão (*Scatter Search*) e o RAMP (*Relaxation Adaptive Memory Programming*).

No capítulo 3, serão feitas referências a estas metaheurísticas, daí a importância da sua descrição prévia.

2.2.1. GRASP

O GRASP, originalmente proposto por Feo e Resende [12], é constituído por duas fases, a fase de construção de soluções e a fase de pesquisa local. Na fase de construção de soluções, é utilizada uma função gulosa para a escolha dos elementos a adicionar à solução corrente. Esta escolha é parcialmente aleatória, sendo usualmente escolhido um elemento de entre os elementos de uma lista restrita de candidatos (*Restricted Candidate List* - RCL). Os elementos que incorporam a lista são escolhidos segundo um determinado critério, que é normalmente, alterado após um número fixo de iterações.

A cada solução gerada é aplicado um método de pesquisa local até ser encontrado um ótimo local. Este processo é repetido até que um critério de paragem seja atingido, como por exemplo o número máximo de soluções geradas.

A Figura 2, apresenta um algoritmo genérico para o GRASP.

```
1.  Enquanto critério de paragem não satisfeito
2.      Gerar solução de forma gulosa
3.      Aplicar método de melhoramento
```

Figura 2 - Algoritmo genérico do GRASP

2.2.2. Pesquisa Tabu

A pesquisa tabu, proposta por Glover [13], estende o conceito de pesquisa local de forma a tornar possível a exploração de zonas no espaço de soluções não consideradas promissoras.

Esta metaheurística faz uso de estruturas de memória para orientar a pesquisa. Na sua versão mais simples, utiliza memória de curta duração. Para evitar voltar a ótimos locais, é utilizada uma lista tabu que armazena características dos movimentos que serão considerados proibidos (tabu) nas iterações seguintes.

A memória de curta duração pode não ser suficiente para possibilitar que a pesquisa escape de certas zonas do espaço de soluções, pois, usualmente, armazena apenas os atributos das soluções visitadas mais recentemente. A memória de longa duração armazena informação que pode permitir diversificar a pesquisa para regiões não exploradas ou intensificar a pesquisa de regiões mais promissoras.

O estado tabu não é permanente sendo controlado por um parâmetro, geralmente designado de *tabu tenure,* que pode ser, por exemplo, um número determinado de iterações de pesquisa.

Os movimentos considerados tabu apenas poderão ser efetuados nas iterações seguintes caso cumpram os critérios definidos pelo critério de aspiração, por exemplo, o movimento dar origem à melhor solução encontrada até ao momento.

A Figura 3, apresenta um algoritmo genérico para a pesquisa tabu para um problema de minimização, considerando $N(S_c)$ a vizinhança da solução corrente e T representando a lista de movimentos considerados tabu. O conjunto $S(T)$ representa as soluções da vizinhança

$N(S_c)$ que são proíbidas, pois os movimentos necessários para as obter pertencem ao conjunto T.

1. Obter uma solução inicial, S_0, cujo valor para a função objetivo é $f(S_0)$
2. Inicializar a solução corrente S_c e a melhor solução encontrada S_b, fazendo $S_b = S_c = S_0$
3. Inicializar o contador de iterações $k = 0$
4. Se $N(S_c) - S(T) \neq \emptyset$
5. Incrementar k e selecionar a melhor solução $S_v \in N(S_c) - S(T)$
6. Se $f(S_v) < f(S_c)$ então
7. $S_b = S_v$
8. Senão
9. $N(S_c) = N(S_c) - S_v$
10. Atualizar T
11. Se ainda não foi atingido o numero fixo de iterações totais, ou de iterações consecutivas sem melhorar S_b, ir para o passo 4

Figura 3 - Algoritmo genérico da pesquisa tabu

2.2.3. Pesquisa por Dispersão

A pesquisa por dispersão, inicialmente proposta por Glover [14], é um método evolutivo, no qual uma população de soluções evolui com a combinação dos seus elementos.

Esta metaheurística constrói novas soluções a partir da combinação de soluções pertencentes a um conjunto de referência. O conjunto de referência deve conter soluções de elevada qualidade e soluções diversificadas para maximizar a informação útil que pode ser retirada da combinação das soluções.

A Figura 4, apresenta um algoritmo genérico para a pesquisa por dispersão baseado na divisão em cinco métodos proposta por Laguna e Martí [15].

A fase inicial da pesquisa por dispersão inicia-se com o método de geração de soluções diversificadas que é usado para gerar um conjunto de soluções diversificadas que servirão de base para todo o desenvolvimento da pesquisa por dispersão. Este método deve ser capaz de produzir um conjunto que providencie um bom equilíbrio entre a qualidade e a

diversificação das soluções. De forma a garantir a diversidade das soluções geradas é, geralmente, usado um procedimento guloso, como por exemplo, o GRASP.

```
Fase Inicial
1.  Aplicar método de Geração de Soluções Diversificadas
2.  Aplicar método de Melhoramento
3.  Aplicar método de Atualização do Conjunto de Referência
Fase de Pesquisa por Dispersão
4.  Enquanto existirem novas soluções no conjunto de referência
5.      Aplicar método de Geração de Subconjuntos
6.      Aplicar método de Combinação de Soluções
7.      Aplicar método de Melhoramento
8.      Aplicar método de Atualização do Conjunto de Referência
```

Figura 4 - Algoritmo genérico da pesquisa por dispersão

As soluções geradas pelo método de geração de soluções diversificadas, sujeitas ao método de melhoramento (passo 2), são integradas num conjunto, usualmente referido com a população inicial (P). O tamanho da população inicial é, geralmente, pequeno, não ultrapassando as 100 soluções.

O método de melhoramento é utilizado para transformar uma solução, proveniente do método de geração de soluções diversificadas (passo 1) ou do método de combinação de soluções (passo 6), numa ou mais soluções de melhor qualidade. Este é o único método opcional para a implementação da pesquisa por dispersão, mas a sua inclusão é recomendada para a obtenção de soluções de qualidade mais elevada.

Após a geração de todas as soluções da população inicial P, as soluções constantes neste conjunto são sujeitas ao método de atualização do conjunto de referência (passo 3), de forma a escolher um subconjunto, de tamanho moderado (b), para servir de conjunto de referência.

O tamanho do conjunto de referência (b) é geralmente de 10 soluções. Embora outros critérios possam ser considerados, usualmente, as primeiras $b1$ ($b1 = b/2$) soluções são as melhores em termos de qualidade, e as restantes $b2$ ($b2 = b/2$), as mais diversificadas em respeito às soluções que constituem o conjunto. O conjunto de referência é ordenado pela qualidade da solução, no qual o primeiro elemento é a melhor solução obtida.

A cada iteração da fase da pesquisa por dispersão, o método de geração de subconjuntos (passo 5) opera sobre o conjunto de referência para produzir subconjuntos que servirão de

base ao método de combinação de soluções. A implementação típica deste método consiste em gerar todos os pares possíveis de soluções que contêm pelo menos uma solução nova, embora possa ser considerada a geração de subconjuntos de maior dimensão. Uma solução é considerada nova se ainda não foi sujeita ao método de combinação de soluções.

O método de combinação de soluções (passo 6) transforma um determinado subconjunto, criado no método de geração de subconjuntos (passo 5), numa ou mais soluções através da sua combinação. As soluções resultantes da combinação, são sujeitas ao método de melhoramento e colocadas na $pool$ de soluções que irão ser consideradas para incorporar o conjunto de referência. Em algumas implementações, este método pode gerar soluções não admissíveis e nesse caso, o método de melhoramento, para além de tentar melhorar a solução, deve tentar repor a admissibilidade da solução.

O método de atualização do conjunto de referência (passo 8), responsável por controlar a entrada de soluções no conjunto de referência, verifica se as soluções existentes na $pool$ de soluções, resultantes do método de combinação de soluções, satisfazem os critérios para incorporar o conjunto de referência.

A atualização do conjunto de referência pode ser executada de duas formas distintas, através de *Static Update* ou de *Dynamic Update*. O *Static Update* efetua a atualização do conjunto de referência, com as soluções existentes na $pool$, quando é terminada a combinação de todas soluções. Numa implementação com *Dynamic Update,* se uma nova solução satisfizer os critérios para ser admitida no conjunto de referência, este é imediatamente atualizado, antes mesmo de ser efetuada a próxima combinação. Nesta abordagem não é necessário a utilização de uma $pool$ de soluções. Esta abordagem tem por vantagem substituir rapidamente soluções de fraca qualidade existente no conjunto de referência, mas tem como desvantagem, a eliminação de futuras combinações que poderiam ser promissoras antes de poderem ser exploradas.

Como referido, a fase de pesquisa por dispersão é executada enquanto existirem soluções novas no conjunto de referência, e, embora opcional, pode ainda ser implementado um processo de reconstrução do conjunto de referência, que permita que o processo não pare na primeira vez que o conjunto não contenha novas soluções. Nas implementações onde a reconstrução do conjunto de referência é incluída, soluções do conjunto de referência são substituídas por soluções diversificadas. Usualmente, todas as soluções, excetuando as melhores em termos de qualidade, são substituídas por soluções provenientes de P ou de uma nova geração de soluções através do método de geração de soluções diversificadas.

Em implementações do *Scatter Search,* com processos de reconstrução do conjunto de referência, a fase de pesquisa por dispersão repete-se enquanto o critério de paragem é satisfeito, como por exemplo: o número de reconstruções ou o tempo de execução.

O método de geração de soluções diversificadas, o método de melhoramento e o método de combinação de soluções são específicos para cada problema, enquanto o método de geração de subconjuntos e método de atualização do conjunto de referência são genéricos.

Devido à sua flexibilidade, a pesquisa por dispersão, permite o desenvolvimento de implementações alternativas com variados níveis de sofisticação. Laguna e Martí [15] descrevem implementações avançadas para os cinco métodos.

2.2.4. RAMP

O método RAMP, proposto por Rego [1], é uma metaheurística baseada na exploração da relação primal-dual dos problemas de otimização combinatória. O termo primal diz respeito ao espaço de soluções do problema original e o termo dual refere-se ao espaço de soluções do problema dual resultante de uma relaxação do problema original.

"*O RAMP combina princípios fundamentais de relaxação matemática, com conceitos de memória adaptativa e técnicas de pesquisa metaheurísticas abrangendo os espaços de soluções primal e dual com o objetivo de obter direções de pesquisa que incorporam informação obtida de ambos os lados do espaço de soluções*" Rego [1].

O RAMP está organizado para tirar partido de vários métodos existentes utilizando-os como componentes para construir estratégias de pesquisa mais avançadas. Rego [1] apresenta diferentes estratégias para os componentes primal e dual permitindo diferentes níveis de sofisticação na implementação do RAMP. Na parte dual salienta-se o uso de relaxações, como a relaxação lagrangeana, a relaxação por restrições substitutas ou a relaxação paramétrica cruzada. Na parte primal, salienta-se, para as versões mais simples a pesquisa tabu e a pesquisa por religação de caminhos (*Path Relinking*), e para versões mais complexas a pesquisa por dispersão. A estrutura do RAMP permite que sejam usadas outras formas de relaxação e outros métodos evolutivos.

O RAMP alterna entre o primal e o dual até, que um critério de paragem seja satisfeito. No lado dual é utilizada uma técnica de relaxação. De forma a criar uma solução admissível para o espaço de soluções primal, é utilizado um método de projeção, usualmente sujeito a um método de pesquisa local. As estruturas de memória adaptativa são empregues para tirar partido de informação gerada tanto do lado primal como do dual.

O RAMP permite diferentes níveis de sofisticação. As versões mais simples efetuam uma exploração mais intensa no lado dual e são usualmente referidas apenas como RAMP ou Dual RAMP. As versões mais avançadas exploram intensamente ambos os espaço de soluções primal e dual, e são designadas por PD-RAMP (Primal-Dual RAMP), de modo a reforçar a relação primal-dual.

Na versão mais simples do RAMP a pesquisa desenvolve-se principalmente no espaço de soluções dual, e a interação primal-dual é conseguida com simples formas de projeção das soluções duais para o espaço de soluções primal, seguidamente sujeita a um método de melhoramento.

No PD-RAMP, são integradas estratégias mais avançadas, permitindo uma pesquisa mais alargada no espaço de soluções primal e a criação de estruturas de memória adaptativa mais elaboradas que afetam ambos os lados da relação primal-dual. No PD-RAMP as soluções primais e duais são combinadas de forma evolutiva, utilizando um conjunto de referência comum e atualizado por ambos os lados.

A Figura 5 [1], representa o modelo genérico do método PD-RAMP.

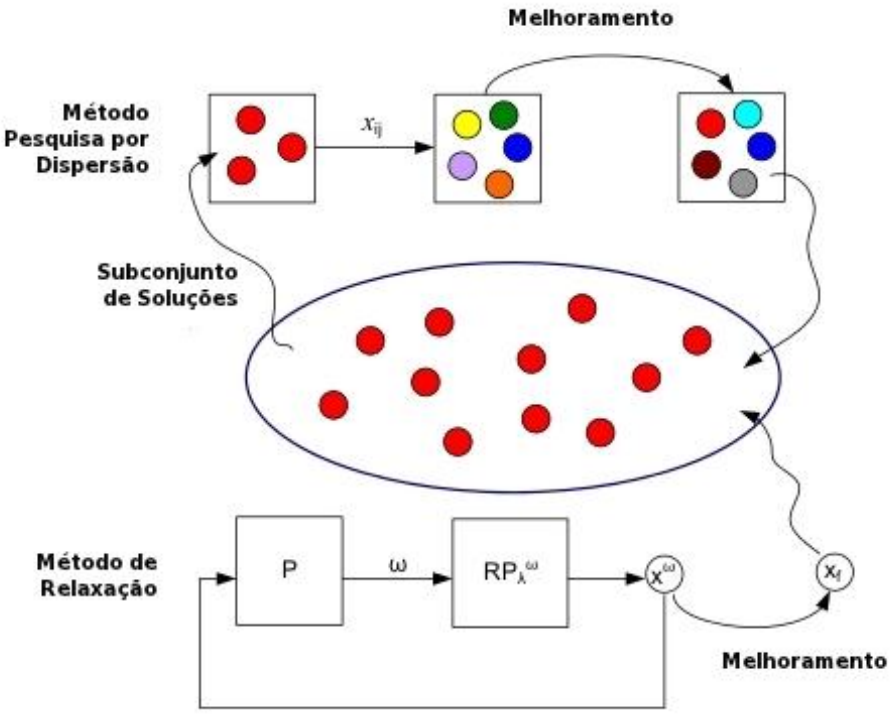

Figura 5 - Modelo genérico do método PD-RAMP [1]

O método inicia-se com a criação e inicialização do conjunto de referência com as soluções geradas por um método de relaxação do problema original. Nesta etapa, as soluções duais são projetadas para o espaço de soluções primal e, após sujeitas ao método de

melhoramento, incluídas no conjunto de referência, passando este a conter informação relevante obtida pelo lado dual.

O método de pesquisa por dispersão seleciona e combina as soluções do conjunto de referência, gerando, a partir destas, novas soluções, que após melhoramento, poderão ser incluídas no conjunto de referência.

O método alterna entre o lado primal e o lado dual, terminando quando não são incluídas novas soluções no conjunto de referência, ou quando um determinado critério de paragem é alcançado.

O RAMP pode ser implementado de forma incremental, iniciando-se com a sua versão mais simples (Dual RAMP) e passando sucessivamente a formas mais complexas do método, na tentativa de obter os melhores resultados possíveis (ou esperados) com a aplicação do método.

O RAMP demostrou ser muito eficiente para uma variedade de problemas de otimização combinatória, como são exemplo os trabalhos de Gamboa [5], Rego *et al.* [6] e Riley *et al.* [7], tendo produzido algoritmos de estado-da-arte para os problemas em que foi aplicado.

2.3. Técnicas de Relaxação Matemática

A relaxação de um problema tem por objetivo tornar mais fácil a sua resolução, podendo fornecer limites para a função objetivo do problema original. No caso de problemas de minimização (maximização), a resolução do problema relaxado será um limite inferior (superior) para o problema original.

Nesta secção, iremos apresentar alguns tipos de relaxação tendo a sua descrição por base o problema genérico de programação linear 0-1 seguinte:

$$Z = \min cx \tag{1}$$

Sujeito a:

$$Ax \leq b \tag{2}$$

$$Dx \leq e \tag{3}$$

$$x \in \{0,1\} \tag{4}$$

2.3.1. Relaxação Linear

A relaxação linear de um problema é efetuada relaxando a restrição de integralidade das variáveis (4), sendo o problema relaxado Z_l composto por todas as soluções inteiras e reais do problema, formando desta forma um conjunto de soluções mais abrangente.

O problema relaxado pode ser resolvido por métodos conhecidos, como por exemplo o método Simplex, sendo o seu resultado um limite para o problema original. Relaxando o problema original Z através da relaxação de programação linear, obtemos o seguinte problema relaxado:

$$Z_l = \min cx \tag{5}$$

Sujeito a:

$$Ax \leq b \tag{6}$$

$$Dx \leq e \tag{7}$$

$$0 \leq x \leq 1 \tag{8}$$

2.3.2. Relaxação Lagrangeana

Muitos problemas de otimização combinatória são modelados através de um conjunto de restrições que consideradas em conjunto dificultam a resolução do problema original Z. O problema lagrangeano, mais fácil de resolver, é criado dualizando as restrições que dificultam a resolução do problema.

A relaxação lagrangeana remove da formulação original as restrições que dificultam a resolução do problema. As restrições relaxadas são colocadas na função objetivo do problema, sendo penalizada quando existir uma violação dessas restrições. Esta penalização é controlada por coeficientes chamados de multiplicadores de Lagrange (λ). Através da relaxação lagrangeana do problema original Z, obtemos o seguinte problema relaxado:

$$Z_\lambda = min\, cx + \lambda\,(ax - b) \tag{9}$$

Sujeito a:

$$Dx \leq e \tag{10}$$

$$x \in \{0,1\} \tag{11}$$

O problema dual consiste em encontrar os multiplicadores λ que maximizem o valor da função objetivo do problema relaxado Z_λ. Se o problema for de minimização (maximização) o valor obtido através do problema relaxado será um limite inferior (superior) para a solução ótima do problema original, pois constata-se que a função objetivo do problema original Z

avaliada para qualquer ponto x_v admissível será sempre igual ou superior à função objetivo de Z_λ para o mesmo ponto x_v.

As heurísticas lagrangeanas são baseadas na relaxação lagrangeana e na resolução do problema lagrangeano dual relacionado, e podem ser usadas para criar heurísticas muito eficientes, por exemplo, atuando as soluções do problema lagrangeano como soluções iniciais para uma heurística primal.

2.3.3. Relaxação por Restrições Substitutas

A relaxação por restrições substitutas (*Surrogate Constraints Relaxation*), introduzida por Glover [16], consiste no uso de um vetor de multiplicadores (ω) para transformar um conjunto de restrições numa só restrição, servindo esta restrição como substituta das restrições relaxadas. A nova restrição é designada de restrição substituta ou *surrogate*.

Se relaxarmos o conjunto de restrições (6) do problema original Z com uma restrição substituta, o problema relaxado é definido por:

$$Z^{\omega} = \min cx \tag{12}$$

Sujeito a:

$$\omega(Ax - b) \leq 0 \tag{13}$$

$$Dx \leq e \tag{14}$$

$$x \in \{0,1\} \tag{15}$$

Otimização por Subgradiente

A otimização por subgradiente é um método para a resolução do problema dual lagrangeano que, de forma iterativa, ajusta os multiplicadores de Lagrange para encontrar os valores que maximizem o limite inferior.

A otimização por subgradiente é usada para resolver o dual lagrangeano de forma a melhorar os limites inferiores, mas raramente obtém uma solução admissível para o problema original. Para garantir a construção de soluções admissíveis é necessário aplicar um método de projeção das soluções duais para o espaço de soluções primal.

Embora a utilização seja mais usual em relaxações lagrangeanas, a sua utilização também tem sido aplicada na determinação de pesos para relaxações por restrições substitutas [5].

A Figura 6, apresenta um algoritmo genérico de otimização por subgradiente baseado na descrição de Beasley [17], tendo em conta o problema original Z e o problema relaxado Z_λ.

1. $\pi = \pi_{init}$, $\lambda = 0$ e $Z_{max} = -\infty$
2. Obter uma solução inicial, Z_{ub}
3. Repetir
4. $x_\lambda^* \leftarrow$ Resolver dual lagrangeano
5. $\delta = ax_\lambda^* - b$
6. $\Delta = \frac{\pi(Z_{ub} - f(x_\lambda^*))}{\|\delta\|}$
7. $\lambda = \max(0, \lambda + \Delta\delta)$
8. Se $f(x_\lambda^*) > Z_{max}$ então
9. $Z_{max} = f(x_\lambda^*)$
10. Se Z_{max} não melhorar em 30 iterações então
11. $\pi = \pi/2$
12. Até $\pi < \pi_{min}$ ou $Z_{ub} = f(x_\lambda^*)$

Figura 6 - Algoritmo genérico de otimização por subgradiente

O algoritmo necessita de um método de resolução do problema dual lagrangeano e de um limite superior inicial Z_{ub} para o problema original. O limite superior inicial pode ser calculado encontrando uma solução admissível para o problema original, através de uma heurística.

A cada iteração da otimização por subgradiente, o problema dual lagrangeano é resolvido com os multiplicadores de Lagrange atuais. Após o cálculo da direção (δ) e do tamanho do passo (Δ), os multiplicadores de Lagrange são atualizados. Quando não são detetadas melhorias por um período de tempo alargado, o que pode significar que se estejam a dar passos demasiado largos, o parâmetro *agility* (π) é diminuído. A otimização por subgradiente repete-se até que seja encontrada a solução ótima ou que o *agility* seja um valor muito reduzido.

Remetemos para Fisher [18] para uma explicação mais aprofundada sobre a relaxação lagrangeana e a otimização por subgradiente.

2.3.4. Relaxação Paramétrica Cruzada

A relaxação paramétrica cruzada (*Cross-parametric Relaxation*), introduzida por Rego [1], combina a relaxação lagrangeana com a relaxação por restrições substitutas, utilizando a otimização por subgradiente para o cálculo do multiplicador lagrangeano (λ) do problema substituto e do vetor de multiplicadores (ω).

O problema obtido através da relaxação paramétrica cruzada é representado através do seguinte modelo matemático:

$$Z_{\lambda}^{\omega} = min\, cx + \lambda\omega\,(ax - b) \quad (16)$$

Sujeito a:

$$Dx \leq e \quad (17)$$

$$x \in \{0,1\} \quad (18)$$

A cada iteração da relaxação paramétrica cruzada, o problema substituto é criado através da otimização por subgradiente para encontrar o vetor de multiplicadores (ω) das restrições relaxadas.

A solução dual do problema substituto é encontrada por otimização por subgradiente da relaxação lagrangeana do problema substituto, gerando novos limites inferiores.

O limite superior é encontrado através da projeção do problema substituto para o espaço de soluções admissíveis do problema original. Estes limites são utilizados para gerar um novo problema substituto.

3. Problemas de Localização de Instalações

Os problemas de localização de instalações (*Facility Location Problems*) são caracterizados pela existência de um conjunto de clientes (ou centros de procura) que necessitam de ter a sua procura satisfeita e de um conjunto de localizações onde se podem abrir instalações (ou centros de oferta) que podem satisfazer a procura dos clientes.

A resolução destes problemas consiste em determinar a localização das instalações a abrir e a afetação dos clientes, tendo em conta o objetivo pretendido, como por exemplo, a minimização dos custos.

Os problemas de localização de instalações fazem parte de um grupo de problemas otimização combinatória de difícil resolução (NP-difíceis) [4] [8], em que a resolução ótima, através de métodos exatos, principalmente para instâncias de grande dimensão, pode ser comprometida pelos tempos de execução e pelos recursos computacionais necessários.

Na generalidade dos problemas de localização de instalações, a abertura de uma instalação incorre num custo de abertura (e de operação), designado de custo de instalação, e num custo de satisfação da procura de um cliente por uma instalação, designado de custo de transporte (ou de custo de afetação).

Dependendo do objetivo e das restrições impostas, é possível encontrar um vasto conjunto de variantes de problemas de localização de instalações. Nos trabalhos de Klose e Drexl [9] e Farahani e Masoud [10] podem ser encontradas a descrição de diversos problemas de localização de instalações.

O vasto conjunto de variantes de problemas de localização de instalações tornou importante a existência de formas de classificação de modo a enquadrar cada um deles segundo um conjunto de critérios ou propriedades comuns. Daskin [19] classifica os problemas de localização de instalações segundo os seguintes critérios:

- Problemas contínuos, discretos ou em rede - Um problema de localização pode ser classificado pelo seu espaço de soluções como contínuo, discreto ou em rede. Um

problema é considerado contínuo, se as instalações e os clientes puderem ser colocados em qualquer lugar no plano. Um problema é classificado como discreto, se existir um conjunto pré-definido de localizações possíveis para as instalações. Um problema é considerado em rede, se os clientes e as instalações apenas puderem ser colocados ao longo de uma rede ou grafo.

- Problemas em árvore ou em grafo completos - Os problemas em árvores apresentam-se como grafos acíclicos, e os problemas em grafos completos, como grafos em que existe uma aresta entre cada par de vértices.
- Métrica de distância - Os problemas podem ser caracterizados pelo método empregue para o cálculo da distância entre os pontos, como por exemplo, a distância euclidiana.
- Quantidade de instalações a abrir - Os problemas de localização de instalações podem ser divididos em problemas em que a quantidade de instalações a abrir é desconhecida e em problemas em que o número de instalações que é necessário abrir é um parâmetro do problema.
- Problemas dinâmicos ou estáticos - Um problema é considerado dinâmico, se incorporar uma componente temporal, tendo por objetivo saber não apenas onde, mas também quando abrir ou fechar as instalações. Se o problema não incorporar uma componente temporal, é classificado como estático.
- Problemas probabilísticos ou determinísticos - Os problemas podem introduzir incerteza (através de valores de entrada probabilísticos ou aleatórios) sendo classificados de probabilísticos, caso contrário, são considerados de determinísticos.
- Quantidade de produtos - Os problemas podem considerar a procura, por parte dos clientes, de apenas um ou de vários produtos.
- Problemas orientados para o setor privado ou para o setor público - Nos problemas orientados para o sector privado, os custos são normalmente medidos em unidades monetárias. Usualmente, os problemas orientados para o setor público consideram igualmente aspetos não monetários, como por exemplo, maximizar a quantidade de estudantes que possam ser afetos a uma nova escola numa região.
- Quantidade de objetivos **-** Um problema pode ter apenas um objetivo (por exemplo: minimizar os custos para satisfazer a procura de todos os clientes) ou ter vários objetivos (por exemplo: maximizar a cobertura por parte das instalações, minimizando os custos).
- Elasticidade da procura - Os problemas de localização podem ser classificados pela existência ou inexistência de elasticidade na procura dos clientes.

- Capacidade das instalações - Os problemas são classificados de capacitados quando as instalações têm uma capacidade máxima para satisfazer a procura dos clientes, e classificados de não capacitados quando a capacidade das instalações é ilimitada.
- Quantidade de instalações que podem servir um cliente - A procura de um cliente pode ser satisfeita unicamente por uma instalação, sendo classificado de *single source* (fonte única ou único servidor), ou ser dividida por várias instalações sendo classificado de *multiple source* (fonte múltipla).
- Problemas hierárquicos ou de nível único - Nos problemas de nível único, as instalações satisfazem a procura dos clientes de forma direta. Os problemas hierárquicos, assumem a existência de instalações em níveis intermédios entre as instalações de topo e os clientes.
- Instalações desejadas e instalações indesejadas - Alguns problemas podem ter por objetivo determinar a abertura de instalações que são indesejadas, como por exemplo, instalações que podem por em causa o bem-estar da comunidade, sendo o objetivo destes problemas, afastar o máximo possível essas instalações, por oposição às instalações desejadas, como por exemplo hospitais, que se esperam estarem o mais próximos possível.

Para considerar outro tipo de classificações, ver Tafazzoli e Mozafari [20] que apresentam um resumo de propostas de classificação, apresentadas por vários autores, para problemas de localização de instalações.

3.1. Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor

Neste estudo é considerado o Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor (*Single Source Capacitated Facility Location Problem* - SSCFLP).

Tendo em conta a classificação de Daskin [19], classificámos o SSCFLP, segundo os critérios mais relevantes, como um problema discreto, determinístico, estático, capacitado, de nível e fonte única. Este problema tem apenas um objetivo (minimização dos custos de abertura de instalações e dos custos de afetação) e considera apenas a procura de um produto por parte dos clientes.

Este problema é um caso especial de Problema de Localização de Instalações com Restrições de Capacidade (*Capacitated Facility Location Problem* - CFLP) que pertence à classe dos problemas NP-difíceis [8], o que implica que não existem algoritmos que

resolvam à otimalidade todas as instâncias do problema em tempo polinomial. Ao contrário do CFLP, no SSCFLP, a procura de cada cliente tem de ser satisfeita por uma única instalação. A resolução do SSCFLP é, geralmente, mais difícil, pois todas as variáveis de decisão são inteiras e binárias (no caso do CFLP, as variáveis de decisão relativas à afetação dos clientes são contínuas).

O SSCFLP tem várias aplicações práticas como, por exemplo, no planeamento de sistemas de distribuição e na conceção de redes informáticas.

O SSCFLP envolve um conjunto J de n clientes e um conjunto I de m localizações candidatas para abertura de instalações. As instalações têm um custo fixo de abertura f_i e uma capacidade máxima para responder à procura dos clientes s_i. Cada cliente j tem uma procura associada d_j que tem de ser servida por uma única instalação. Neste problema, satisfazer a procura do cliente j por uma instalação localizada em i tem um custo de transporte c_{ij}.

Este problema pode ser representado pelo seguinte modelo matemático:

$$SSCFLP = \ min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{19}$$

Sujeito a:

$$\sum_{i \in I} x_{ij} = 1 \ \ \forall j \in J \tag{20}$$

$$\sum_{j \in J} d_j \, x_{ij} \leq s_i y_i \ \ \forall i \in I \tag{21}$$

$$x_{ij} \in \{0,1\} \ \forall i \in I, j \in J \tag{22}$$

$$y_i \in \{0,1\} \ \forall i \in I \tag{23}$$

Considerando as variáveis x_{ij} e y_i:

$$x_{ij} = \begin{cases} 1, & se\ cliente\ j\ é\ servido\ pela\ instalação\ em\ i \\ 0, & caso\ contrário \end{cases} \tag{24}$$

$$y_i = \begin{cases} 1, & se\ existe\ uma\ instalação\ na\ localização\ i \\ 0, & caso\ contrário \end{cases} \tag{25}$$

A expressão (19) estabelece o objetivo do problema, neste caso, a minimização dos custos de abertura de instalações e dos custos de afetação. O conjunto de restrições (20) assegura que todos os clientes tenham a sua procura atendida por uma única instalação, e o conjunto de restrições (21) assegura que os pedidos dos clientes afetos à instalação i não excedam a

sua capacidade. As restrições (22) e (23) asseguram que as variáveis de decisão sejam binárias.

Vários autores propuseram abordagens para a resolução do SSCFLP, sendo a sua maioria baseadas em heurísticas lagrangeanas.

Neebe e Rao [21] modelam o SSCFLP como um Problema de Partição de Conjuntos, resolvido através de um método exato de Busca em Árvore, neste caso o *Branch and Bound*.

Barceló e Casanovas [22] propuseram uma heurística lagrangeana com a relaxação das restrições de afetação de clientes (20). Nesta heurística, o número máximo de instalações abertas (p) é predefinido, adicionando a restrição $\sum_{i \in I} y_i \leq p$ à formulação do problema.

Klincewicz e Luss [23] apresentam uma heurística lagrangeana dualizando as restrições de capacidade (21). O problema relaxado, que resulta num Problema de Localização de Instalações sem Restrições de Capacidade (*Uncapacitated Facility Location Problem* - UFLP) é resolvido com o algoritmo dual ascent de Erlenkotter [24].

Sridharan [25] e Pirkul [26] apresentam heurísticas lagrangeana relaxando as restrições de afetação de clientes (20). Sridharan utiliza, para obter soluções admissíveis, um algoritmo baseado no Problema de Transportes de Fonte Única (*Single-source Transportation Problem*), e Pirkul combina a relaxação lagrangeana com um *Branch and Bound.*

Beasley [17] apresenta uma *framework* para o desenvolvimento de heurísticas lagrangeanas (relaxação lagrangeana e otimização por subgradiente) para vários problemas de localização (*p-Median Location Problem*, UFLP, CFLP e SSCFLP). Beasley concluiu que as heurísticas apresentadas são robustas para os problemas propostos. No SSCFLP, foram relaxadas as restrições de afetação de clientes (20) e as restrições de capacidade (21). A heurística proposta obteve melhores resultados que os de Klincewicz e Luss [23], mas piores que os apresentados por Pirkul [6].

Delmaire *et al.* [27] propõem quatro heurísticas, um algoritmo evolutivo, um *Greedy Randomized Adaptative Search Procedure* (GRASP), uma pesquisa por arrefecimento simulado e uma pesquisa tabu. Neste estudo é feita a introdução de uma reformulação da função objetivo (19), representando o problema como:

$$\sum_{j \in J} cmin_j + \ min\left(\sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} \Delta_{ij} x_{ij}\right) \qquad (26)$$

Considerando:

$$cmin_j = min\left\{c_{ij}, \forall i \in I\right\} \quad (27)$$

$$\Delta_{ij} = c_{ij} - cmin_j \quad (28)$$

O custo relativo de satisfazer a procura do cliente j na instalação i com respeito ao custo mínimo (28) é utilizado em várias das heurísticas propostas como critério de escolha de clientes. O GRASP, comparativamente com as restantes heurísticas propostas, obteve tempos computacionais mais reduzidos, o que permitiu que o método de melhoramento utilizado incorporasse mais três estruturas de vizinhança. As quatro metodologias demonstraram serem bastante eficientes. Os melhores resultados foram obtidos pela pesquisa tabu e pelo GRASP, sendo este último menos exigente em termos computacionais.

Delmaire *et al.* [28] apresentam quatro heurísticas: um GRASP reativo (onde os parâmetros de decisão se autorregulam), uma pesquisa tabu e dois algoritmos híbridos que combinam elementos do GRASP, do GRASP reativo e da pesquisa tabu. O GRASP reativo e a pesquisa tabu foram mais eficientes do que os propostos no trabalho anterior [27], embora o GRASP reativo seja mais exigente em termos computacionais quando comparado com o GRASP. As duas abordagens híbridas conseguiram os melhores resultados.

Rönnqvist *et al.* [29] apresentam uma heurística baseada em *Repeated Matching*. Holmberg *et al.* [30] propõem um método exato que utiliza uma relaxação lagrangeana, dualizando as restrições de afetação de clientes (20), de forma a obter um limite inferior, e uma heurística baseada em *Repeated Matching* para obter o limite superior. Estes limites são depois incorporados num *Branch and Bound* para encontrar a solução ótima para o problema.

Hindi e Pienkosz [31] apresentam uma heurística lagrangeana com relaxação das restrições de afetação de clientes (20). Os limites superiores são conseguidos através de uma heurística gulosa baseada em máximo arrependimento (*Maximum Regret*) e pesquisa em vizinhança restrita (*Restricted Neighborhood Search*). Esta abordagem demonstrou ser eficiente para instâncias de grande dimensão.

Outra heurística que provou ser muito eficiente para problemas de grande dimensão é a apresentada por Ahuja *et al.* [32]. O método proposto é baseado em vizinhanças de grande dimensão (*Very Large Scale Neighborhood* - VLSN). No primeiro tipo de vizinhança utilizado, as soluções vizinhas podem ser alcançadas através de movimentos de um cliente ou de um conjunto de clientes. O número máximo de clientes que podem ser transferidos

simultaneamente é um parâmetro do algoritmo de modo a diminuir a vizinhança a explorar em problemas de grande dimensão. O segundo tipo de vizinhança explora o espaço das soluções que podem ser alcançadas através de três movimentos nas instalações (abertura, encerramento e transferência).

Cortinhal e Captivo [33] apresentam várias abordagens baseadas em algoritmos genéticos para a resolução do SSCFLP. Neste estudo, os autores concluem que os algoritmos genéticos não são um método de resolução eficiente para este problema.

Cortinhal e Captivo [34] propõem uma heurística lagrangeana, onde os limites inferiores são obtidos usando uma relaxação lagrangeana dualizando as restrições de afetação de clientes (2). Os resultados do problema relaxado, obtidos a cada iteração da otimização por subgradiente, são submetidos a um método de projeção e melhorados com uma pesquisa tabu.

Contreras e Díaz [35] apresentam uma pesquisa por dispersão (*Scatter Search*) para a resolução do SSCFLP. A sua implementação é baseada na divisão da pesquisa por dispersão em cinco métodos de Laguna e Martí [15]. Na fase de construção do conjunto de referência é utilizado o GRASP proposto por Delmaire *et al.* [27]. Os resultados deste método superaram os apresentados (em instâncias de pequena e média dimensão) por Ahuja *et al.* [32], mas foram claramente inferiores às abordagens híbridas propostas por Delmaire *et al.* [28].

Várias heurísticas baseadas na otimização por colónia de formigas (*Ant Colony Optimization - ACO*) foram apresentadas para a resolução do SSCFLP, como são exemplo os trabalhos de Kumweang [36], de Chen e Ting [37] [38], e Lina *et al.* [39].

Mais recentemente Yang *et al.* [40] apresentaram três versões de um método exato para a resolução do SSCFLP baseado no *Cut-and-Solve* proposto por Climer e Zhang [41].

3.2. Algoritmo Dual RAMP

O algoritmo Dual RAMP aqui proposto, baseia-se no lado dual, na otimização por subgradiente para a resolução do problema dual obtido através da relaxação lagrangeana do SSCFLP. No lado primal, as soluções projetadas do espaço de soluções dual para o primal são sujeitas a uma pesquisa tabu.

O algoritmo inclui um método de resolução do problema dual lagrangeano e necessita de um limite superior inicial para o problema original, que é encontrado através de um GRASP.

A cada iteração da otimização por subgradiente, é obtida uma solução (Z_λ^*) do problema relaxado, sendo esta solução projetada para o espaço de soluções primal através do método de projeção.

A solução obtida através do método de projeção é sujeita ao método de melhoramento e colocada numa $pool$ de soluções, ordenada de forma crescente relativamente ao valor da função objetivo.

O limite superior utilizado para o cálculo dos novos multiplicadores lagrangeanos é sempre a primeira solução da $pool$ (a melhor solução encontrada até ao momento).

O processo é repetido até que seja atingido o número máximo de iterações predefinido, podendo ser interrompido se for encontrada a solução ótima do problema ou se o valor do *agility* atingir um valor muito reduzido.

A Figura 7, apresenta os passos gerais do algoritmo Dual RAMP. Os componentes duais do algoritmo estão representados por retângulos com preenchimento azul, e os componentes primais, por retângulos sem preenchimento. A melhor solução obtida pelo algoritmo (S_b) é representada pelo círculo vermelho.

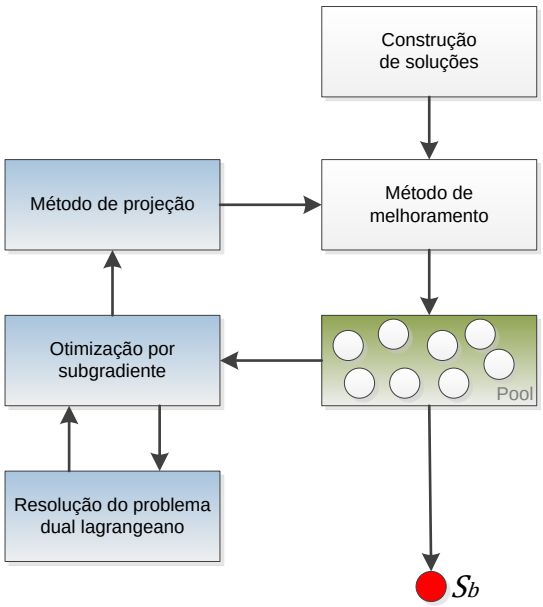

Figura 7 - Funcionamento do Dual RAMP

3.2.1. Método Primal

O método primal, do algoritmo Dual RAMP, é constituído por dois componentes, um procedimento GRASP e um método de melhoramento.

O método de melhoramento tem a responsabilidade de tentar melhorar as soluções vindas da fase de construção de soluções obtidas através do procedimento GRASP, e das soluções provenientes da projeção das soluções duais para o espaço de soluções primal.

As soluções sujeitas ao método de melhoramento são inseridas numa $pool$ de soluções (que pode ser encontrada nos dois algoritmos RAMP propostos). As soluções incluídas na $pool$ são todas diferentes e estão ordenadas tendo em conta a qualidade das soluções, sendo a primeira a melhor solução encontrada.

GRASP

O GRASP implementado é por si só uma metaheurística para a resolução do SSCFLP, mas o seu principal objetivo é o de criar soluções iniciais de boa qualidade em tempos computacionais reduzidos.

No Dual RAMP, a melhor solução encontrada pelo GRASP, servirá como limite superior inicial para o cálculo dos multiplicadores de Lagrange.

A Figura 8, apresenta o funcionamento geral do GRASP, constituído pela fase de construção de soluções e pela fase de melhoramento das soluções geradas. O círculo a vermelho representa a melhor solução obtida (S_b) pelo procedimento GRASP.

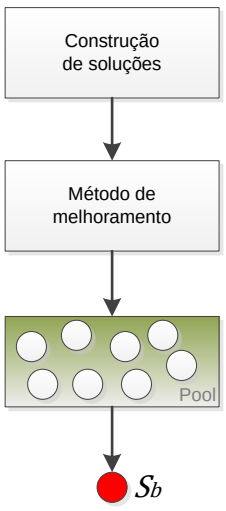

Figura 8 - Funcionamento do GRASP

De forma a garantir tempos computacionais reduzidos a quantidade de soluções geradas pelo GRASP, que identificamos como $pool_{Size}$, varia com o tamanho do problema ($número\ de\ clientes \times número\ de\ instalações$).

O GRASP gera um máximo de 100 soluções ($Sol_{MaxGRASP}$) para problemas com um tamanho inferior ou igual a 5000, sendo proporcionalmente reduzido até um mínimo de 10 soluções ($GRASP_{SolMin}$) para problemas com um tamanho superior ou igual a 10000.

As duas fases do GRASP são repetidas até se conseguir gerar todas as soluções definidas pelo parâmetro $pool_{Size}$ ou até que um número máximo de falhas ($falhas_{max}$) seja atingido.

Na implementação proposta $falhas_{max} = pool_{Size} * 2$. Uma falha é considerada quando numa iteração não foi possível gerar uma solução admissível, ou quando é gerada uma solução que já se encontra na $pool$.

A fase de construção de soluções é baseada no GRASP proposto por Delmaire *et al.* [27] que demonstrou ser capaz de gerar rapidamente soluções de qualidade e ao mesmo tempo providenciar diversidade de soluções.

Para cada solução, as instalações a abrir são selecionadas, de forma aleatória, a partir de uma lista restrita de candidatos (*Restricted Candidate List* - RCL). Para o conjunto de instalações fechadas, é calculado o custo médio por cliente a servir, através da seguinte fórmula:

$$\varphi_i = \frac{f_i + \sum_{i \in Clientes_i} \Delta_{ij}}{|Clientes_i|} \quad (29)$$

Em que $Clientes_i$ identifica o conjunto de clientes sem instalação que satisfaça a sua procura, ordenados de forma crescente de Δ_{ij} (28) e com uma quantidade agregada de procura que não ultrapassa a capacidade da instalação i.

Considerando o conjunto de instalações abertas $A = \{i \in I: y_i = 1\}$, $\varphi_{min} = min\{\varphi_i: i \in I \setminus A\}$ e $\varphi_{max} = max\{\varphi_i: i \in I \setminus A\}$, a entrada no conjunto RCL (30) é controlada pelo valor $\alpha_{corrente}$.

$$RCL = \{i: \varphi_i \leq \varphi_{min} + \alpha_{corrente}(\varphi_{max} - \varphi_{min})\} \quad (30)$$

Dos elementos que constituem a RCL, é escolhida de forma aleatória, uma instalação i a abrir, sendo atendida a procura dos clientes do conjunto $Clientes_i$. Este processo é repetido, enquanto existirem clientes com procura não satisfeita ou enquanto não seja possível efetuar novas afetações.

De forma a permitir a entrada de mais instalações na RCL, e possibilitar uma diversidade maior na escolha das instalações a abrir, o valor de $\alpha_{corrente}$ não é fixo, sendo incrementado

(α_{inc}) após um número fixo de iterações (α_{iter}) do GRASP. O valor inicial de $\alpha_{corrente}$ tem como valor inicial o parâmetro α_{ini}.

Os parâmetros α_{iter}, α_{ini} e α_{inc} variam de acordo com a quantidade de soluções a serem geradas, calculados através das seguintes formulas:

$$\alpha_{iter} = max\left\{1, \frac{pool_{Size}}{GRASP_{SolMin}}\right\} \quad (31)$$

$$\alpha_{ini} = \frac{pool_{Size}}{GRASP_{SolMax} * GRASP_{SolMin}} \quad (32)$$

$$\alpha_{inc} = \frac{(pool_{Size}/GRASP_{SolMax}) - \alpha_{ini}}{max\{1, pool_{Size}/\alpha_{iter}\}} \quad (33)$$

Se a solução gerada não for admissível, os clientes cuja procura não é satisfeita são afetos às instalações abertas com menor Δ_{ij} (28), mesmo que a restrição de capacidade seja violada. Se a solução resultante não for admissível é sujeita ao método de reposição de admissibilidade, descrito no final desta secção.

No final da iteração do GRASP, a solução, se admissível, é sujeita ao método de melhoramento, sendo este responsável por melhorar a solução e pela sua colocação na $pool$.

A Figura 9, apresenta um algoritmo do GRASP.

1. $\alpha_{corrente} = \alpha_{ini}$, $falhas = 0$, $iteração = 0$
2. Enquanto $|pool| < pool_{Size}$ e $falhas < falhas_{max}$
3. $\quad iteração = iteração + 1$
4. $\quad S = \emptyset$
5. $\quad$ Se $iteração = \alpha_{iter}$ então
6. $\quad\quad \alpha_{corrente} = \alpha_{corrente} + \alpha_{inc}$
7. $\quad\quad iteração = 0$
8. $\quad Terminar = falso$
9. $\quad$ Enquanto $S \neq J$ e $Terminar = falso$
10. $\quad\quad \varphi_i = \frac{f_i + \sum_{i \in Clientes_i} \Delta_{ij}}{|Clientes_i|} \quad \forall i \in I \backslash A$
11. $\quad\quad \varphi_{min} = min\{\varphi_i : i \in I \setminus A\}$
12. $\quad\quad \varphi_{max} = max\{\varphi_i : i \in I \setminus A\}$
13. $\quad\quad RCL = \{i : \varphi_i \leq \varphi_{min} + \alpha_{corrente}(\varphi_{max} - \varphi_{min})\}$
14. $\quad\quad$ Se $RCL = \emptyset$ então
15. $\quad\quad\quad Terminar = verdadeiro$
16. $\quad\quad$ Senão
17. $\quad\quad\quad$ Selecionar aleatoriamente i' da RCL
18. $\quad\quad\quad a(j) = i' \quad \forall j \in Clientes_{i'}$
19. $\quad\quad\quad S = S \cup Clientes_{i'}$
20. $\quad$ Se $Terminar = verdadeiro$ então
21. $\quad\quad A(j) = i' \in \{i : min_{i \in A} C_{ij}\} \forall j \in S$
22. $\quad\quad$ Se a solução não for admissível então
23. $\quad\quad\quad$ Aplicar método de reposição de admissibilidade
24. $\quad$ Se a solução for admissível então
25. $\quad\quad$ Aplicar método de melhoramento

Figura 9 - Algoritmo do GRASP

Método de Melhoramento

O método de melhoramento é responsável por tentar melhorar uma solução inicial através de um processo iterativo de pesquisa no espaço de soluções vizinhas da solução corrente. O método de melhoramento é baseado numa pesquisa tabu, na sua versão mais simples, utilizando memória de curta duração para impedir que sejam revisitadas soluções recentemente visitadas.

O método de melhoramento aqui proposto utiliza cinco estruturas de vizinhança. As figuras seguintes ilustram os movimentos das estruturas de vizinhança utilizadas, apresentado no lado esquerdo a solução corrente e no lado direito a solução vizinha conseguida com a aplicação do respetivo movimento. Os círculos representam os clientes e os retângulos as instalações. As instalações abertas são representadas por retângulos com preenchimento azul e as instalações encerradas com um preenchimento cinzento. Nas soluções vizinhas, o preenchimento negro nos círculos que representam os clientes, identifica que houve alteração na afetação dos clientes.

- *Shift Client* – Efetua a transferência de afetação de um cliente para outra instalação aberta (Figura 10).

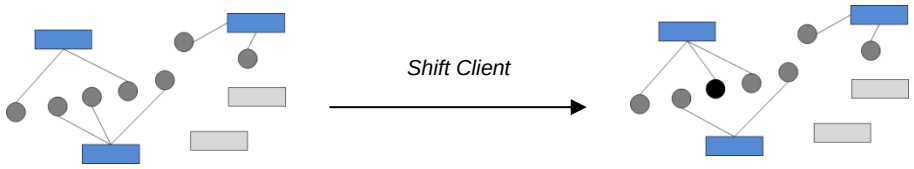

Figura 10 - Movimento *Shift Client*

- *Swap Client* – Efetua a troca de afetação entre dois clientes (Figura 11).

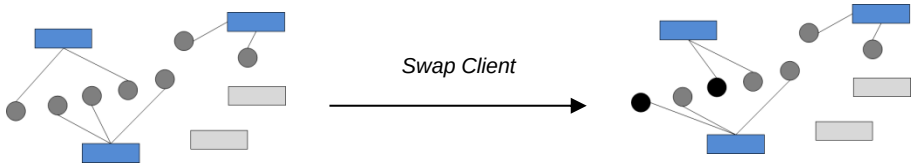

Figura 11 - Movimento *Swap Client*

- *Swap Facility* – Efetua o movimento de troca de todos os clientes entre instalações abertas (Figura 12) ou de afetação de todos os clientes de uma instalação aberta para uma instalação encerrada (Figura 13).

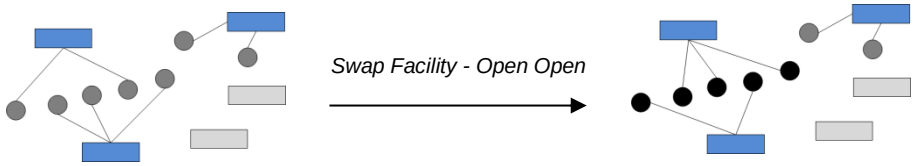

Figura 12 - Movimento *Swap Facility* entre instalações abertas

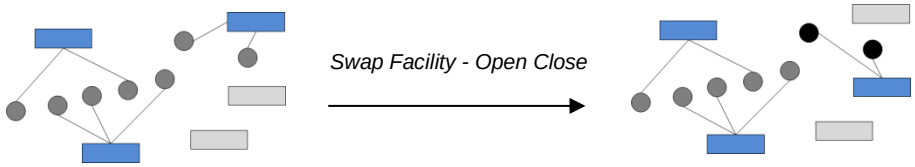

Figura 13 - Movimento *Swap Facility* entre uma instalação aberta e uma fechada

- *Close Opened* – Efetua o movimento de encerramento de uma instalação e afetação dos clientes pelas restantes instalações abertas (Figura 14).

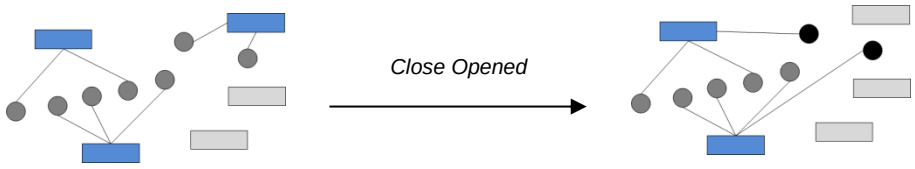

Figura 14 - Movimento *Close Opened*

- *Open Closed* – Efetua o movimento de abertura de uma nova instalação e atribuição de clientes (Figura 15).

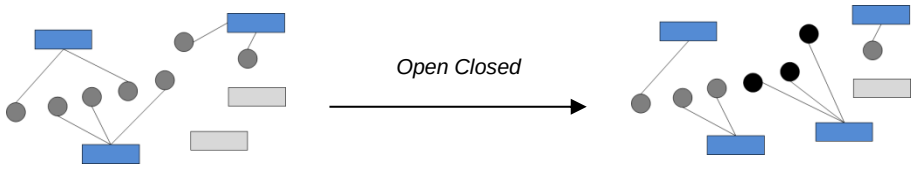

Figura 15 - Movimento *Open Closed*

Na estrutura de vizinhança de clientes *Shift Client,* a pesquisa é efetuada apenas pelas instalações em que o cliente possui um valor inferior de Δ_{ij} (28).

De forma a reduzir o tempo computacional da pesquisa na estrutura de vizinhança *Swap Client*, limita-se a vizinhança das soluções, explorando, para cada cliente, apenas as trocas com os clientes com menor Δ_{ij} (28) na instalação a que está afeto.

Nas estruturas de vizinhança de instalações (*Swap Facility, Close Opened e Open Closed*) são exploradas todas as soluções vizinhas.

Nas cinco estruturas de vizinhança apresentadas, são apenas concretizados os movimentos que mais reduzam o valor da função objetivo.

Nos movimentos *Shift Client* e *Swap Client*, é utilizada uma lista tabu de movimentos, representados por (i,j) em que i representa a localização da instalação, e j o cliente. Quando a instalação i, que satisfaz a procura de um cliente j, é alterada, a afetação prévia (i,j) é considerada tabu durante um período de tempo, desta forma prevenindo que se visitem soluções exploradas recentemente.

Nos movimentos *Close Opened e Open Closed*, é utilizada uma lista tabu de estados para as instalações. Se uma instalação for encerrada (aberta) num dos movimentos, fechar (abrir) a instalação será considerado tabu nas próximas iterações.

Um movimento tabu, pode no entanto ser efetuado, se a solução obtida desse movimento levar à melhor solução conhecida (critério de aspiração).

O estado tabu de um movimento não é permanente, sendo controlado por um intervalo de tempo (*tabu tenure*), por exemplo, o número de iterações, que têm de decorrer para o movimento ser removido da lista. No algoritmo implementado, o *tabu tenure* para a lista tabu de clientes, é igual ao número de iterações do método de melhoramento determinado pela expressão $TT_{clientes}$ (34) e o *tabu tenure* para a lista de instalações, é determinado pela expressão $TT_{instalações}$ (35).

$$TT_{clientes} = número\ de\ clientes/3 \tag{34}$$

$$TT_{instalações} = número\ de\ instalações/4 \tag{35}$$

A Figura 16, apresenta um algoritmo do funcionamento genérico do método de melhoramento implementado.

```
1.  terminar = falso
2.  Enquanto terminar = falso
3.      Explorar Shift Client enquanto melhorar a solução
4.      Se Explorar Swap Client não melhorar a solução
            e Explorar Swap Facility não melhorar a solução
            e Explorar Close Opened não melhorar a solução
            e Explorar Open Closed não melhorar a solução
5.          terminar = Verdadeiro
```

Figura 16 - Algoritmo do método de melhoramento

Método de Reposição de Admissibilidade

Para as soluções que violem as restrições de capacidade (21), é executado um conjunto de movimentos para tentar repor a admissibilidade da solução. Este conjunto integra três tipos de movimentos que são aplicados enquanto for possível melhorar a solução ou até que a admissibilidade seja atingida. Os movimentos possíveis são:

- *Reassignment Shift* - Efetua o melhor movimento de reafectação de um cliente servido por uma instalação com capacidade excedida, para uma instalação aberta com capacidade residual suficiente para satisfazer a sua procura.
- *Reassignment Swap* - Efetua a troca de clientes entre instalações que mais reduz a violação das restrições de capacidade (21).

- *Reassignment Open* – Efetua o melhor movimento de abertura de uma instalação, que permita a reafetação de um cliente ao menor custo.

O método é interrompido, quando nenhum dos movimentos consegue ser efetuado, e ainda não tenha sido atingida a admissibilidade, não conseguindo, desta forma, devolver uma solução considerada válida.

A Figura 17, apresenta um algoritmo do método de reposição de admissibilidade.

```
1.  resultado = verdadeiro
2.  Enquanto resultado = verdadeiro e solução não admissível
3.      resultado ← Reassigment Shift
4.      Se resultado = falso e solução não admissível
5.          resultado ← Reassigment Swap
6.      Se resultado = falso e solução não admissível
7.          resultado ← Reassigment Open
```

Figura 17 - Algoritmo do método de reposição de admissibilidade

3.2.2. Método Dual

O método dual baseia-se na resolução do dual lagrangeano do SSCFLP, através de otimização por subgradiente, com a projeção da solução obtida para o espaço de soluções primal.

No método dual, optou-se pela relaxação das restrições de afetação de clientes (20), pois segundo o estudo realizado por Beasley [17], os melhores resultados obtidos através da relaxação lagrangeana para o SSCFLP, são conseguidos através da relaxação deste conjunto.

Relaxação Lagrangeana do SSCFLP

Considerando os multiplicadores de Lagrange $\lambda_j, j = 1, \dots, n$ surge a seguinte relaxação lagrangeana sujeita às restrições (21) a (23):

$$Z(\lambda) = min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} \lambda_j \left(1 - \sum_{i \in I} x_{ij} \right) \qquad (36)$$

Este problema decompõe-se em m, um para cada instalação, problemas da mochila (*Knapsack problems*). Para cada instalação i obtemos o seguinte problema:

$$Z_i(\lambda) = \ min \sum_{j \in J} (c_{ij} - \lambda_j)\, x_{ij} + \ f_i y_i \tag{37}$$

Sujeito a:

$$\sum_{j \in J} d_j\, x_{ij} \leq s_i y_i \tag{38}$$

$$x_{ij} \in \ \{0, 1\}\ \forall\ j \in J \tag{39}$$

$$y_i \in \ \{0, 1\} \tag{40}$$

Se considerarmos $y_i = 1$, obtemos o seguinte problema da mochila:

$$KP_i = \ min \sum_{j \in J} (c_{ij} - \lambda_j)\, x_{ij} \tag{41}$$

Sujeito a:

$$\sum_{j \in J} d_j\, x_{ij} \leq s_i \tag{42}$$

$$x_{ij} \in \ \{0, 1\}\ \forall\ j \in J \tag{43}$$

O resultado do problema relaxado pode ser alcançado após a resolução ótima dos m problemas da mochila. O valor ótimo das variáveis $\bar{y}_i$ e $\bar{x}_{ij}$ é obtido seguindo as seguintes regras:

$$\bar{y}_i = \begin{cases} 1, & \text{se } f_i + KP_i < 0 \\ 0, & caso\ contrário \end{cases} \tag{44}$$

$$\bar{x}_{ij} = \begin{cases} 1, & se\ y_i = 1\ e\ x_{ij} = 1 \\ 0, & caso\ contrário \end{cases} \tag{45}$$

A função objetivo do problema relaxado torna-se:

$$Z(\lambda) = \ min \sum_{i \in I} \left(f_i \bar{y}_i + \sum_{j \in J} (c_{ij} - \lambda_j) \bar{x}_{ij} \right) + \sum_{j \in J} \lambda_j \tag{46}$$

Para a resolução dos problemas da mochila resultantes da relaxação lagrangeana do SSCFLP, foi utilizado o algoritmo proposto por Martello *et al.* [42]. Uma análise comparativa deste algoritmo com outros propostos na literatura pode ser encontrada no trabalho de Pisinger [43]. O código utilizado foi implementado em C e disponibilizado por Pisinger [44].

Descrição do algoritmo

Na implementação do algoritmo Dual RAMP para a resolução do SSCFLP, os multiplicadores de Lagrange são inicializados com $\lambda_j = \min_i c_{ij}\ \forall j \in J$.

O parâmetro *agility* (π) é inicializado com o valor 1.5, e a cada 5 falhas consecutivas de melhoramento do limite inferior Z_λ^*, é dividido por 2, de forma a diminuir o tamanho do passo. O *agility* é reiniciado com o valor 2 a cada 30 iterações da otimização por subgradiente.

A Figura 18, apresenta um algoritmo do Dual RAMP.

1. $\pi = 1.5,\ k_{max} = 500,\ k = 0,\ falhas = 0,\ Z_{max} = 0,\ \lambda_i = \min_{i \in I} c_{ij}\ \forall j \in J$
2. $Z_{ub} \leftarrow$ Resolver problema original através do GRASP
3. Terminar = falso
4. Enquanto Terminar = falso e $k_{max} > k$ e $\pi > 0.003$
5. $\quad k = k + 1$
6. $\quad Z_\lambda^* \leftarrow$ Resolver dual lagrangeano
7. $\quad \hat{Z} \leftarrow$ Método de projeção + Método de melhoramento
8. $\quad$ Se $f(Z_{ub}) > f(\hat{Z})$ então
9. $\quad\quad Z_{ub} = \hat{Z}$
10. $\quad$ Se $f(Z_\lambda^*) > f(Z_{max})$ então
11. $\quad\quad Z_{max} = Z_\lambda^*$
12. $\quad\quad falhas = 0$
13. $\quad$ Senão
14. $\quad\quad falhas = falhas + 1$
15. $\quad\quad$ Se $falhas \,\%\, 5$ então
16. $\quad\quad\quad \pi = \pi/2$
17. $\quad$ Se $k \,\%\, 30$ então
18. $\quad\quad \pi = 2$
19. $\quad \delta_j = 1 - \sum_{i \in I} x_{ij}\ \forall j \in J$
20. $\quad$ Se $f(Z_{ub}) - f(Z_{max}) < 1$ ou $\|\delta\| = 0$ então
21. $\quad\quad$ Terminar = verdadeiro
22. $\quad$ Senão
23. $\quad\quad \Delta = \frac{\pi(f(Z_{ub}) - f(Z_\lambda^*))}{\|\delta\|}$
24. $\quad\quad \lambda_j = \lambda_j + \Delta\delta_j \forall j \in J$

Figura 18 - Algoritmo do Dual RAMP

A otimização por subgradiente tem quatro critérios de paragem:

- O *agility* (π) ser inferior a 0.003;
- A norma ($\|\delta\|$) atingir o valor 0;
- Ser atingido o valor máximo de iterações k_{max};
- A diferença entre o valor do limite superior (Z_{ub}) e do limite inferior (Z_{max}) ser inferior a 1.

Método de Projeção

De forma o obter uma solução válida para o problema original ($\hat{Z}$) é utilizado um método de projeção baseado no método apresentado por Cortinhal e Captivo [14].

Na solução obtida pela resolução do problema relaxado, os clientes do conjunto J estão divididos em três subconjuntos:

- $J_1 = \{j \in J: \sum_{i \in I} x_{ij} = 1\}$;
- $J_2 = \{j \in J: \sum_{i \in I} x_{ij} > 1\}$;
- $J_3 = \{j \in J: \sum_{i \in I} x_{ij} = 0\}$.

Considerando $A = \{i \in I: y_i = 1\}$ como sendo o conjunto de instalações abertas e $h_i = \sum_{j \in J} d_j x_{ij}\ \forall i \in I$ a quantidade de procura satisfeita pela instalação i, a obtenção de uma solução é assegurada, garantindo que todos os clientes sejam servidos por uma instalação, procedendo da seguinte forma:

- A afetação dos clientes do subconjunto J_1 não é alterada.
- Os clientes $j \in J_2$ são afetos à instalação i_2: $c_{i_2 j} = \min_{i \in \{i \in A: x_{ij} = 1\}} c_{ij}$.
- Os clientes $j \in J_3$, ordenados por ordem decrescente de pedidos (d_j), são afetos à instalação i_3: $c_{i_3 j} = \min_{i \in \{i \in A: h_i + D_j \leq s_i\}} c_{ij}$, caso $i_3 = \emptyset$ então i_3: $c_{i_3 j} = \min_{i \in A} c_{ij}$.

A solução obtida pelo método de projeção, pode não ser admissível, sendo violado o conjunto de restrições de capacidade (21). Se a solução obtida não for admissível, é sujeita ao método de reposição de admissibilidade (apresentado na secção 3.2.1). Caso seja atingida a admissibilidade da solução, é aplicado o método de melhoramento, sendo encontrado um novo limite superior se o resultado for inferior ao limite superior atual (Z_{ub}).

3.3. Algoritmo PD-RAMP

O método RAMP pode ser implementado de forma incremental, iniciando-se por versões mais simples (Dual RAMP) evoluindo progressivamente para versões mais complexas (PD-RAMP).

Embora os resultados obtidos pelo algoritmo Dual RAMP tenham sido extremamente positivos, foi considerado pertinente verificar se uma versão mais sofisticada do método (RAMP), para a resolução do SSCFLP, poderia apresentar resultados ainda melhores. Surge assim o algoritmo PD-RAMP para o SSCFLP.

No PD-RAMP, são integradas estratégias mais avançadas, permitindo uma pesquisa mais alargada no espaço de soluções primal. As soluções primais e duais são combinadas de forma evolutiva, utilizando um conjunto de referência comum e atualizado por ambos os lados, alternando entre o primal e dual até que a combinação de soluções não consiga incorporar novas soluções no conjunto de referência.

O método RAMP está organizado para tirar partido de vários métodos existentes utilizando-os como componentes para construir estratégias de pesquisa mais avançadas. Rego [3] apresenta como principal estratégia para o componente primal do PD-RAMP, a pesquisa por dispersão, embora outras abordagens possam ser utilizadas.

A Figura 19, apresenta o funcionamento geral do PD-RAMP.

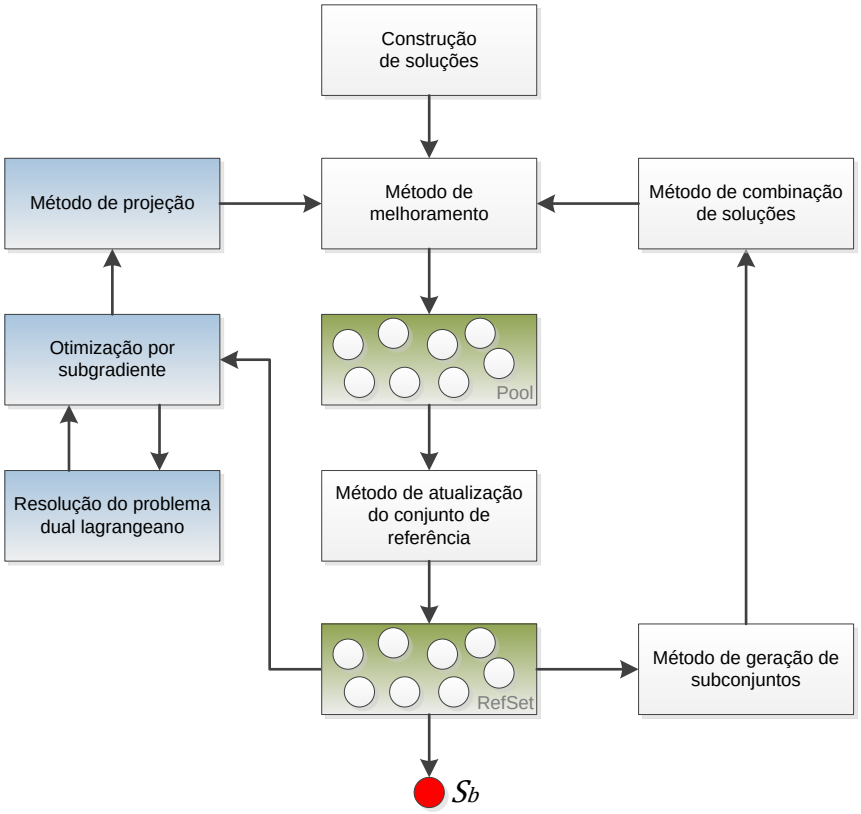

Figura 19 - Funcionamento do algoritmo PD-RAMP

O algoritmo PD-RAMP, utiliza no lado primal uma pesquisa por dispersão e no lado dual, o método dual do algoritmo Dual RAMP já apresentado.

A Figura 20, apresenta um algoritmo genérico do PD-RAMP.

```
1.  Procedimento GRASP
2.  Aplicar método de atualização do conjunto de referência
3.  Enquanto os critérios de paragem não forem satisfeitos
4.     Resolução do problema Dual
5.     Aplicar método de projeção da solução dual
6.     Aplicar método de Melhoramento
7.     Aplicar método de atualização do conjunto de referência
8.     Aplicar método de geração de subconjuntos
9.     Aplicar método de combinação de soluções
10.    Aplicar método de melhoramento
11.    Aplicar método de atualização do conjunto de referência
```

Figura 20 - Algoritmo PD-RAMP

Na fase inicial do algoritmo PD-RAMP, as soluções geradas pelo GRASP (passo 1) são sujeitas ao método de atualização do conjunto de referência (passo 2) do modo a criar um conjunto de referência de tamanho b com soluções de qualidade e soluções diversificadas.

Cada iteração do algoritmo PD-RAMP, inicia-se com a componente dual (passo 4). A solução obtida pelo método dual é projetada para o espaço de soluções primal (passo 5), sendo a solução resultante sujeita ao método de melhoramento e incluída na $pool$ de soluções (passo 6).

A $pool$ de soluções representa um conjunto de soluções provenientes do método de melhoramento, que serão sujeitas ao método de atualização do conjunto de referência (passo 7), de modo a analisar (segundo o critério de qualidade ou de diversificação) a sua possível inclusão no conjunto de referência.

O método de geração de subconjuntos (passo 8) cria conjuntos de duas e três soluções, provenientes do conjunto de referência, a serem combinadas pelo método de combinação de soluções (passo 9), gerando novas soluções. As soluções resultantes da combinação dos subconjuntos são sujeitas ao método de melhoramento (passo 10) e incluídas na $pool$ de soluções. Após a inclusão das soluções na $pool$, estas são analisadas pelo método de atualização do conjunto de referência, de modo a verificar se possuem os critérios (de qualidade ou diversificação) necessários para serem incluídas no conjunto de refrência (passo 11).

O algoritmo PD-RAMP tem como critérios de paragem:

- O *agility* (π) ser inferior a 0.003;
- A norma ($||\delta||$) atingir o valor 0;
- Ser atingido o valor máximo de iterações k_{max};
- A diferença entre o valor do limite superior (Z_{ub}) e do limite inferior (Z_{max}) ser inferior a 1.
- Não serem incluídas novas soluções no conjunto de referência.

3.3.1. Método Primal

O método primal do algoritmo PD-RAMP, consiste numa pesquisa por dispersão baseada na divisão em cinco métodos proposta por Laguna e Martí [15].

A implementação da pesquisa por dispersão teve por objetivo criar um algoritmo que permitisse a obtenção de resultados de qualidade semelhante, ou superior, aos já competitivos resultados, da pesquisa por dispersão de Contreras e Díaz [35].

A Figura 21, apresenta um algoritmo da pesquisa por dispersão proposta.

```
1.  Procedimento GRASP
2.  Aplicar método de atualização do conjunto de referência
3.  Enquanto existirem novas soluções no conjunto de referência
4.     Aplicar método de geração de subconjuntos
5.     Aplicar método de combinação de soluções
6.     Aplicar método de melhoramento
7.     Aplicar método de atualização do conjunto de referência
8.     Aplicar método de eliminação
9.        Aplicar método de atualização do conjunto de referência
          às soluções admissíveis resultantes do método de
          eliminação
```

Figura 21 - Algoritmo da pesquisa por dispersão

As soluções iniciais (passo 1) são geradas através do GRASP apresentado na secção 3.2.1, que demonstrou ser capaz de criar rapidamente soluções de qualidade, e ao mesmo tempo providenciar a diversidade de soluções que a pesquisa por dispersão requer para a combinação de soluções.

O conjunto de soluções diversificadas $(pool)$ está ordenado considerando a qualidade das soluções, sendo a primeira solução do conjunto a que apresenta o menor valor para a função objetivo do problema.

O conjunto de referência é constituído por b soluções e é dividido em dois subconjuntos. No primeiro conjunto, temos as $b1$ melhores soluções (de elite) e no segundo conjunto, as restantes $b2$ soluções (diversificadas). Na implementação proposta neste trabalho $b = 10$, $b1 = 5$ e $b2 = 5$.

Para poder classificar uma solução como diversificada, é criado um vetor de instalações, que denominamos por vetor de diversidade, no qual o estado da instalação reflete o estado predominante das soluções $b1$ que constituem o conjunto de referência. O nível de diversificação de uma solução identifica o número de instalações com estado diferente quando comparado com o vetor de diversidade. Na Figura 22, é apresentado o cálculo do vetor de diversidade, representando nas colunas as instalações. Os valores 1 e 0 representam o estado da instalação (aberto e fechado, respetivamente).

Soluções de elite

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

Vetor de diversidade

1	0	1	1	0	1	0

Figura 22 - Vetor de diversidade

O nível de diversidade de uma solução é determinado comparando o vetor de instalações da solução com o vetor de diversidade, tendo em consideração o número total de diferenças.

No passo 2, é criado o conjunto de referência que inclui as $b1$ melhores soluções e as $b2$ soluções mais diversificadas provenientes da $Pool$.

No passo 4, são gerados todos os subconjuntos de duas e três soluções, provenientes do conjunto de referência, que contêm pelo menos uma solução que nunca foi sujeita ao método de combinação de soluções. Para cada subconjunto, são aplicados três tipos de combinação (passo 5), resultando três novas soluções a incluir na $pool$, que serão avaliadas pelo método de atualização de conjunto de referência.

Considerando F_j como o conjunto das instalações a que o cliente j está afeto nas soluções do subconjunto a combinar, os métodos de combinação constroem a nova solução da seguinte forma:

- Método de Combinação 1 – Na solução criada por este método, a instalação que irá servir a procura de cada cliente j, é escolhida de forma aleatória de entre as constantes em F_j.
- Método de Combinação 2 – Na solução criada por este método, a instalação que irá servir a procura de cada cliente será a que apresentar o menor valor $(c_{ij}/d_j) + (f_i/s_i)\ \forall i \in F_j$
- Método de Combinação 3 – Na solução gerada por este método, são abertas e mantidas as afetações das instalações que em todas as soluções do subconjunto servem o mesmo cliente. Nos subconjuntos de três soluções, se a capacidade das instalações abertas na nova solução for inferior ao total de procura, são abertas e mantidas as afetações das instalações que servem o mesmo cliente em duas das soluções do subconjunto. No final, os clientes sem instalação que satisfaça a sua procura na nova solução são afetos à instalação aberta com menor Δ_{ij}.

No final de cada combinação, a solução resultante pode não ser admissível, podendo existir uma violação das restrições de capacidade (21). Se a solução não for admissível, é sujeita ao método de reposição de admissibilidade, e se atingida a admissibilidade, é sujeita aos movimentos *Shift Client* e *Swap Client* do método de melhoramento.

Após cada execução do método de combinação de soluções, as soluções incluídas $pool$ são avaliadas pelo método de atualização do conjunto de referência (passo 7) de modo a verificar se satisfazem os critérios para substituir soluções do conjunto de referência. As $b1$ melhores soluções são escolhidas da união do conjunto de referência e da $pool$. As soluções $b2$ são substituídas pelas soluções da $pool$ com maior nível de diversidade e superior ao requerido na iteração. O nível de diversidade requerido, é incrementado a cada iteração do método de atualização do conjunto de referência, com um valor de $0{,}05$, iniciando-se com o valor $0{,}0$.

As soluções da $pool$ que não foram introduzidas no conjunto de referência são sujeitas ao método de eliminação (passo 8). No método de eliminação em cada solução é encerrada uma instalação escolhida de forma aleatória de entre os elementos de uma lista restrita de candidatos RCL_2 (47). Considerando a capacidade residual da instalação $r_i = s_i - \sum_{j \in J} D_j x_{ij}$, $r_{min} = min\{r_i : i \in S\}$, $r_{max} = max\{r_i : i \in S\}$, a entrada no conjunto RCL_2 é controlada pelo valor β. O valor de β que produziu melhores resultados é de 3.5.

O conjunto RCL_2 é definido por:

$$RCL_2 = \{i : r_i \geq r_{min} + \beta \times (r_{max} - r_{min})\} \quad (47)$$

Os clientes da instalação encerrada, são reafectos nas instalações abertas com menor Δ_{ij}, mesmo que as restrições de capacidade (21) sejam violadas, sendo neste caso aplicado o método de reposição de admissibilidade. A solução resultante é sujeita ao método de melhoramento.

As soluções admissíveis resultantes do método de eliminação, são avaliadas pelo método de atualização do conjunto de referência (passo 9), de forma verificar a sua possível inclusão no conjunto de referência.

A Figura 23, apresenta o funcionamento geral da pesquisa por dispersão.

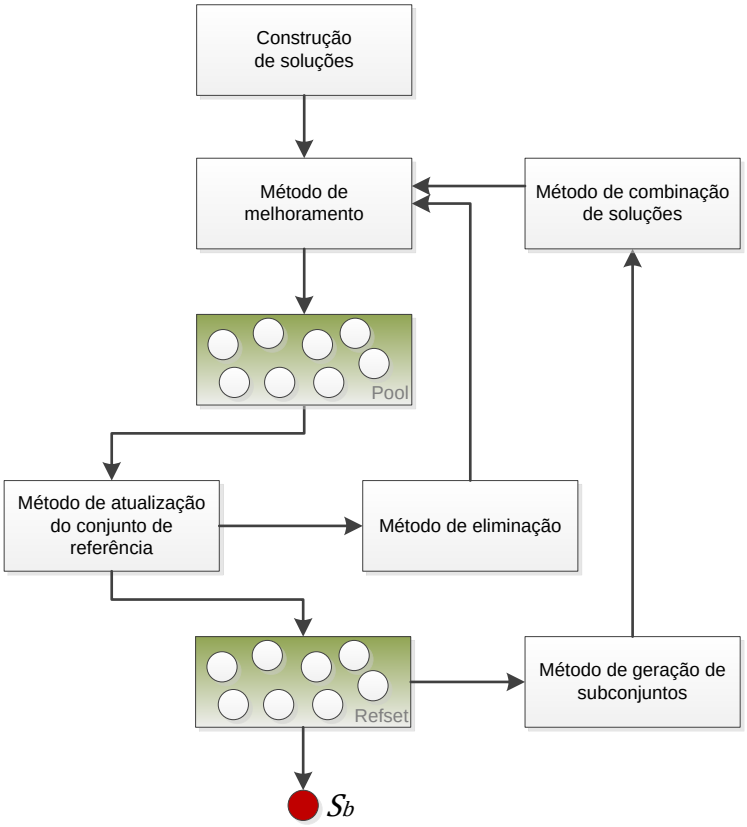

Figura 23 - Funcionamento da pesquisa por dispersão

A pesquisa por dispersão implementada tinha por objetivo obter resultados de qualidade semelhante aos obtidos pela pesquisa por dispersão apresentada por Contreras e Díaz [35]. Contreras e Díaz [35] efetuaram os testes à sua abordagem em dois conjuntos de testes, nomeadamente, o conjunto de Delmaire *et al.* [28] e de Holmberg *et al.* [30]. Ambos os conjuntos possuem problemas de pequena e média dimensão e são apresentados com mais detalhe na secção 3.5.

Com a implementação proposta, obtivemos resultados de qualidade superior aos obtidos pela pesquisa por dispersão de Contreras e Díaz, em tempos computacionais reduzidos. No entanto, para instâncias de grande dimensão (ver Tabela 1), os tempos computacionais são bastante elevados. Embora, Contreras e Díaz [35] não apresentem resultados para este conjunto de instâncias de teste, é expectável, que os tempos computacionais, da pesquisa por dispersão por eles apresentada, também sejam muito elevados.

	Delmaire et al.	**Holmberg et al.**	**Beasley (grande dimensão)**
Contreras e Díaz [35]	5,6 s	22,2 s	-
Pesquisa por dispersão proposta	0,7 s	1,5 s	135.8 s

Tabela 1 - Análise comparativa entre pesquisas por dispersão

Neste contexto, o método primal do PD-RAMP, foi simplificado para reduzir os tempos computacionais exigidos, sendo retirados o método de combinação 3 e o método de eliminação, apresentados anteriormente.

Após a combinação de soluções dos métodos de combinação 1 e 2, as soluções são sujeitas ao método de melhoramento com a totalidade das estruturas de vizinhança apresentadas na secção 3.2.1.

A simplificação do método primal do PD-RAMP, permitiu resultados de elevadíssima qualidade, rivalizando com os melhores da literatura e com tempos computacionais muito reduzidos, quando comparados com os algoritmos propostos em trabalhos anteriores.

3.3.2. Método Dual

O método dual do algoritmo PD-RAMP é o mesmo do algoritmo Dual RAMP apresentado na secção 3.2.2. As soluções geradas pelo método dual, após projeção e melhoramento, são incluídas na $pool$ de soluções para serem avaliadas, de modo a verificar se satisfazem os critérios de inclusão no conjunto de referência.

No PD-RAMP, o limite superior utilizado para o cálculo dos parâmetros da otimização por subgradiente, não é obtido da $pool$, mas sim, do conjunto de referência. O conjunto de referência encontra-se ordenado considerando o valor da função objetivo, sendo a melhor solução a primeira deste conjunto.

3.4. Estruturas de Dados

A escolha das estruturas de dados para representar um determinado problema, tem um papel crucial na eficiência da implementação dos algoritmos para a sua resolução.

Nesta secção, apresentamos as estruturas de dados utilizadas para a implementação eficiente dos algoritmos propostos nesta dissertação. As estruturas são apresentadas utilizando os dados do problema 1 do conjunto de testes de Delmaire *et al.* [28].

A Figura 24, apresenta as estruturas básicas do problema, a matriz de custos C_{ij} que armazena os custo de afetação de clientes nas instalações. O vetor d armazena a procura dos clientes, o vetor s armazena a capacidade das instalações e o vetor f o custo de abertura das instalações.

C_{ij}

i \ j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	8	9	35	60	63	96	51	81	10	95	69	38	6	42	41	80	86	46	8	65
2	85	0	21	80	44	72	36	66	62	61	79	29	8	89	17	82	31	94	56	57
3	5	4	65	82	67	38	34	17	2	92	98	95	85	91	21	63	62	87	3	8
4	49	65	94	46	47	51	53	78	26	44	17	10	45	87	34	4	49	61	11	13
5	42	59	46	16	62	26	71	70	0	54	43	41	25	2	99	82	39	15	71	19
6	8	77	29	18	74	25	0	98	71	72	53	75	65	92	22	84	46	92	18	32
7	87	26	56	78	32	34	8	28	78	39	87	28	20	54	77	75	5	18	81	89
8	93	68	47	31	5	12	21	37	26	83	88	99	67	37	90	38	3	82	77	4
9	62	91	46	45	91	97	12	59	74	25	99	51	94	69	50	87	55	5	33	25
10	84	25	75	88	44	71	95	6	41	26	9	93	82	84	73	79	95	29	76	22

d

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	12	18	18	19	26	21	18	19	18	11	22	21	13	19	13	14	22	17	27	28

s

i	1	2	3	4	5	6	7	8	9	10
	59	48	65	43	64	65	57	49	58	30

f

i	1	2	3	4	5	6	7	8	9	10
	329	144	408	202	369	440	195	162	174	197

Figura 24 - Estruturas de dados básicas

A matrix Δ_{ij} (Figura 25) armazena o custo relativo de satisfazer a procura do cliente j na instalação i com respeito ao custo mínimo (28), seguindo a reformulação proposta por Delmaire *et al.* [25].

Δ_{ij}

i \ j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	3	9	14	44	58	84	51	75	10	70	60	28	0	40	24	76	83	41	5	61
2	80	0	0	64	39	60	36	60	62	36	70	19	2	87	0	78	28	89	53	53
3	0	4	44	66	62	26	34	11	2	67	89	85	79	89	4	59	59	82	0	4
4	44	65	73	30	42	39	53	72	26	19	8	0	39	85	17	0	46	56	8	9
5	37	59	25	0	57	14	71	64	0	29	34	31	19	0	82	78	36	10	68	15
6	3	77	8	2	69	13	0	92	71	47	44	65	59	90	5	80	43	87	15	28
7	82	26	35	62	27	22	8	22	78	14	78	18	14	52	60	71	2	13	78	85
8	88	68	26	15	0	0	21	31	26	58	79	89	61	35	73	34	0	77	74	0
9	57	91	25	29	86	85	12	53	74	0	90	41	88	67	33	83	52	0	30	21
10	79	25	54	72	39	59	95	0	41	1	0	83	76	82	56	75	92	24	73	18

Figura 25 - Matriz de custos relativos de afetação

A matriz $\{\Delta_i\}$ (Figura 26) armazena o índice dos clientes ordenados de forma ascendente relativamente ao Δ_{ij}. Esta estrutura permite que a pesquisa dos clientes a afetar nas instalações seja realizada de forma mais eficiente. A avaliação do movimento de troca de afetação entre clientes (*Swap Clients*) de forma exaustiva é muito exigente em termos computacionais. Na implementação proposta, a pesquisa do melhor movimento de troca de afetação entre clientes é apenas realizada com clientes que tenham menor Δ_{ij} relativamente

à instalação em que o cliente em questão se encontra afeto. Comparativamente com a pesquisa exaustiva, esta pesquisa permitiu uma redução de cerca de 40% do tempo computacional e um aumento de desvio médio praticamente insignificante.

$\{\Delta_i\}$	j																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	13	1	19	2	9	3	15	12	14	18	4	7	5	11	20	10	8	16	17	6
2	2	3	15	13	12	17	7	10	5	20	19	6	8	9	4	11	16	1	14	18
3	19	1	9	20	2	15	8	6	7	3	17	16	5	4	10	13	18	12	11	14
4	12	16	19	11	20	15	10	9	4	6	13	5	1	17	7	18	2	8	3	14
5	4	14	9	18	6	20	13	3	10	12	11	17	1	5	2	8	19	7	16	15
6	7	4	1	15	3	6	19	20	17	11	10	13	12	5	9	2	16	18	14	8
7	17	7	18	13	10	12	6	8	2	5	3	14	15	4	16	19	11	9	1	20
8	20	5	17	6	4	7	3	9	8	16	14	10	13	2	15	19	18	11	1	12
9	18	10	7	20	3	4	19	15	12	17	8	1	14	9	16	6	5	13	11	2
10	11	8	10	20	18	2	5	9	3	15	6	4	19	16	13	1	14	12	17	7

i

Figura 26 - Matriz de ordenação de custos relativos de afetação em instalações

A matriz $\{\Delta_j\}$ (Figura 27) armazena para cada cliente os índices das instalações ordenados de forma ascendente de valor de Δ_{ij}. Esta estrutura permite que os movimento de reafectação (*shift*) sejam realizados de forma menos exigente em termos computacionais, efetuando pesquisas apenas em instalações com menor valor de Δ_{ij}.

$\{\Delta_j\}$	j																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	3	2	2	5	8	8	6	10	5	9	10	4	1	5	2	4	8	9	3	8
2	1	3	6	6	7	6	7	3	3	10	4	7	2	8	3	8	7	5	1	3
3	6	1	1	8	2	5	9	7	1	7	5	2	7	1	6	3	2	7	4	4
4	5	10	9	9	10	7	8	8	8	4	6	1	5	7	4	7	5	10	6	5
5	4	7	5	4	4	3	3	9	4	5	1	5	4	9	1	10	6	1	9	10
6	9	5	8	1	5	4	2	2	10	2	2	9	6	10	9	1	4	4	2	9
7	10	4	7	7	1	10	1	5	2	6	7	6	8	4	10	2	9	8	5	6
8	2	8	3	2	3	2	4	4	6	8	8	10	10	2	7	5	3	3	10	2
9	7	6	10	3	6	1	5	1	9	3	3	3	3	3	8	6	1	6	8	1
10	8	9	4	10	9	9	10	6	7	1	9	8	9	6	5	9	10	2	7	7

i

Figura 27 - Matriz de ordenação de custos relativos de afetação dos clientes

A Figura 28, ilustra as estruturas de dados utilizadas para armazenar as soluções obtidas. As posições de memória estão representadas na figura com um preenchimento cinzento. O vetor de instalações aponta para as posições de memória do primeiro e último cliente afeto em cada instalação. De forma a identificar mais eficientemente a afetação dos clientes, é utilizado um vetor de clientes com as posições de memória dos clientes na solução.

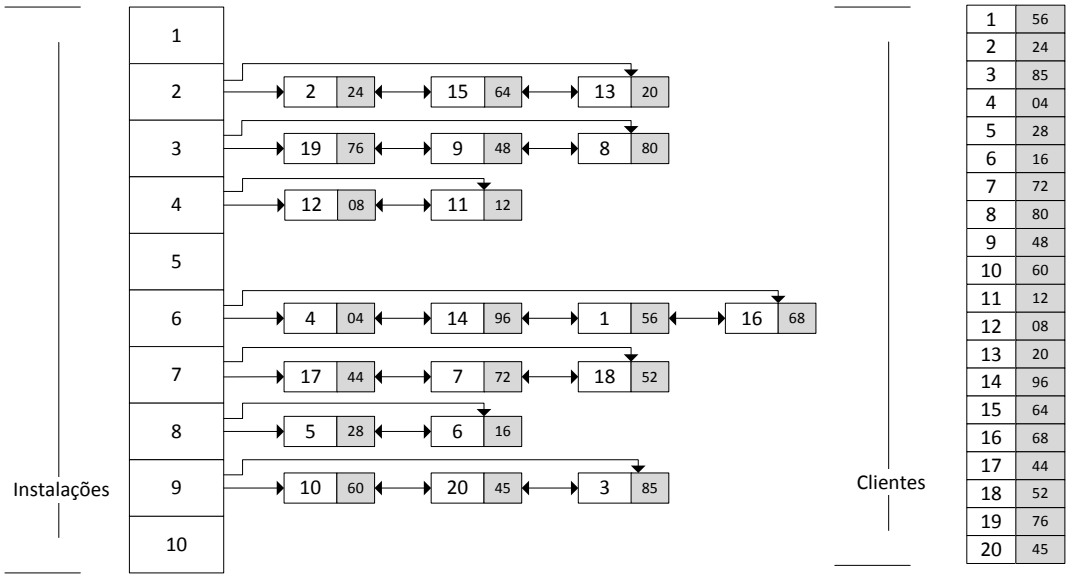

Figura 28 - Estrutura de dados das soluções

As estruturas de dados definidas para a implementação dos algoritmos RAMP para o SSCFLP demonstram ser muito eficientes como se pode verificar pelos tempos computacionais obtidos na execução dos testes (ver secção 3.5).

3.5. Resultados Computacionais

Para analisar o desempenho dos algoritmos propostos, foram efetuados testes utilizando os conjuntos de instâncias de Holmberg *et al.* [30], Delmaire *et al.* [28] e Beasley [45]. Os problemas destes conjuntos estão divididos tendo em conta a sua dimensão ($n \times m$), em que n corresponde ao número de clientes e m ao número de instalações.

O conjunto Holmberg *et al.* tem setenta e uma instâncias de teste, agrupadas em quatro subconjuntos de problemas (Tabela 2).

Subconjunto	Problemas	n	m
1	1 - 24	50	10 - 20
2	25 - 40	150	30
3	41 - 55	70 – 100	10 – 30
4	56 - 71	200	30

Tabela 2 - Conjunto de testes de Holmberg *et al.*

O conjunto de Delmaire *et al.* contém cinquenta e sete instâncias de teste, agrupadas em sete subconjuntos (Tabela 3).

Subconjunto	Problemas	n	m
1	1 - 6	20	10
2	7 - 17	30	15
3	18 - 25	30	20
4	26 - 33	50	20
5	34 - 41	60	30
6	42 - 49	75	30
7	50 - 57	90	30

Tabela 3 - Conjunto de testes de Delmaire *et al.*

O conjunto de Beasley foi gerado para o Problema de Localização de Instalações com Restrições de Capacidade (*Capacitated Facility Location Problem* - CFLP), que não contempla a restrição de um único servidor. Neste estudo apenas foram consideradas as instâncias de teste admissíveis para o SSCFLP, de entre as quais vinte e quatro de tamanho médio e doze consideradas de grande dimensão. Os subconjuntos estão divididos em grupos de quatro instâncias em que a capacidade das instalações é igual (Tabela 4).

Subconjunto	Grupo	n	m
1	cap6 e cap7	50	16
2	cap9 e cap10	50	25
3	cap12 e cap13	50	50
4	capa, capb e capc	1000	100

Tabela 4 - Conjunto de testes de Beasley

Os algoritmos propostos foram implementados em C++ e os testes foram executados num computador com processador Intel Core i3-2350M a 2.30 GHz com 6 Gb RAM e sistema operativo Linux Mint 13. Para cada instância, os algoritmos foram executados apenas uma vez.

Para avaliar a qualidade de uma solução é, normalmente, utilizado o cálculo do desvio percentual da solução obtida relativamente ao valor da solução ótima do problema (48). Considerando Z como a solução obtida, e Z^* a solução ótima do problema, o desvio (gap) é calculado da seguinte forma:

$$Gap = \left(\frac{Z - Z^*}{Z^*}\right) \times 100 \tag{48}$$

De modo a avaliar a qualidade dos resultados obtidos por um algoritmo, a maioria dos autores apresenta o desvio médio conseguido para cada conjunto de instâncias utilizado para testes.

A Tabela 5, apresenta os resultados computacionais obtidos pelos melhores algoritmos existentes na literatura para a resolução do SSCFLP, que utilizam nos seus testes, os conjuntos descritos anteriormente. Nesta tabela, são também apresentados os resultados obtidos pelos algoritmos RAMP propostos neste estudo. Nos anexos, é possível encontrar os resultados obtidos pelos algoritmos implementados neste trabalho, para todas as instâncias dos grupos de teste.

Nas tabelas apresentadas neste capitulo, a coluna T(s), identifica o tempo médio de execução dos algoritmos, medido em segundos, e a coluna NSO, o número de soluções ótimas encontradas pelos algoritmos correspondentes. Alguns autores apresentam nos seus trabalhos várias abordagens para a resolução do SSCFLP. Nestes casos, as abordagens, correspondentes aos resultados apresentados, são identificadas na coluna “Autores” com o seguinte formato: “Nome dos autores – Abordagem”.

Os resultados apresentados por Demaire *et al.* [27] referem-se apenas aos primeiros trinta e três problemas do conjunto. O desvio apresentado por Demaire *et al.* [28] e Cortinhal e Captivo [34], refere-se ao desvio das melhores soluções obtidas após várias execuções dos algoritmos propostos para cada instância. Considerando, que devido à sua aleatoriedade, o algoritmo atinge soluções diferentes cada vez que é executado, apenas seria possível uma comparação rigorosa com a nossa abordagem, caso os autores apresentassem para cada problema, o desvio médio das soluções obtidas após todas as execuções.

A apresentação dos resultados, para o conjunto de testes de Beasley, é habitualmente separada pelos conjuntos de média dimensão (subconjuntos 1, 2 e 3) e de grande dimensão (subconjunto 4).

Autores	Holmberg *et al.*			Delmaire *et al.*			Beasley (pequena dimensão)			Beasley (grande dimensão)		
	Gap	**T(s)**	**NSO**	**Gap**	**T(s)**	**NSO**	**Gap**	**T(s)**	**NSO**	**Gap**	**T(s)**	**NSO**
Demaire *et al.* [27] **–** GRASP				0.40	0.31	4						
Holmberg *et al.* [30]	0.06	7.32	54									
Rönnqvist *et al.* [29]	0.18	31.21	40									
Hindi e Pienkosz [31]										0.77	1719.1	1
Demaire *et al.* [28] **–** GRASP				0.2	17.59	19						
Demaire *et al.* [28] **–** HB				0.02	33.77	46						
Cortinhal e Captivo [34]				0.08	41.89	33	0.05	23.0	18	0.93	9077.9	2
Ahuja *et al.* [32]	0.03	6.91	53				0.00	1.2	24	0.08	74.64	1
Contreras e Díaz [35] **–** GRASP	0.26	-	35	3.98	-	0						
Contreras e Díaz [35] **–** SS	0.02	22.21	46	0.44	5.60	4						
Chen e Ting [38] **–** LH	0.79	4.56	22							0.62	282.4	1
Chen e Ting [38] **–** MAC	0.23	6.24	48							0.39	440.34	4
Chen e Ting [38] **–** LH-MACS	0.02	11.96	60							0.06	281.22	4
Yang *et al.* [40] **-** CS3	0.02	19.94	47	0.08	18.45	12						
Algoritmos Propostos												
GRASP	0.20	0.32	42	2.74	0.07	2	0.04	0.08	17	0.72	1.69	1
Dual RAMP	0.02	0.77	58	0.69	0.76	8	0.00	0.19	24	0.12	18.16	1
Pesquisa por dispersão	0.03	0.86	61	0.37	0.85	16	0.04	0.09	18	0.05	135.81	1
PD-RAMP	0.04	0.53	59	0.52	0.79	7	0.01	0.10	22	0.08	37.85	1

Tabela 5 - Resultados computacionais

Os algoritmos implementados neste estudo são comparados com os melhores existentes na literatura.

Para o conjunto de Delmaire *et al.,* o método híbrido (GRASP e pesquisa tabu - HB) de Delmaire *et al.* [28], é considerado o algoritmo de estado-da-arte por todos os autores. No entanto, uma vez que não é possível uma comparação rigorosa com este algoritmo, tal como explicado previamente, decidiu-se comparar os algoritmos RAMP com a pesquisa por dispersão de Contreras e Díaz [35] – SS-CD

Na Tabela 6, apresenta-se um resumo dos resultados obtidos com as nossas implementações (GRASP, SS, Dual RAMP e PD-RAMP) e os resultados reportados por Contreras e Díaz [35]. (SS-CD).

	SS-CD	**GRASP**	**SS**	**Dual RAMP**	**PD-RAMP**
% Desvio	0.44	2.74	0.37	0.69	0.52
T(s)	5.60	0.07	0.85	0.76	0.79
NSO	4	2	16	8	7

Tabela 5 - Análise dos resultados no conjunto de Delmaire *et al.*

No conjunto de Delmaire *et al.*, os algoritmos RAMP conseguem resultados de qualidade elevada em tempos computacionais muito reduzidos. Como referido anteriormente, ambas as abordagens de pesquisa por dispersão conseguem resultados excelentes, em termos de qualidade de soluções, neste conjunto de problemas, mas quando testadas em problemas de grande dimensão, requerem tempos computacionais muito elevados. É de referir que a nossa abordagem SS, apresenta resultados de qualidade bastante superior à de Contreras e Díaz [35] no conjunto de Delmaire *et al.*.

Considerámos que os desvios apresentados, pelos nossos algoritmos, se devem mais às características do próprio conjunto de testes, do que da abordagem seguida. Neste conjunto, as afetações dos clientes estão muito ligadas às capacidades das instalações. Como referido nos trabalhos futuros, temos já em desenvolvimento uma abordagem, que tudo indica, nos permitirá obter resultados ao nível de Delmaire *et al.* [28].

Nos conjuntos de Holmberg *et al.* e de Beasley, os algoritmos RAMP são comparados com o algoritmo de Ahuja *et al.* [32], baseado em vizinhanças de grande dimensão (*Very Large Scale Neighborhood* - VLSN). Este algoritmo, apresentava-se como o mais robusto da literatura para estes conjuntos, pois obtém resultados de elevada qualidade para problemas de pequena e grande dimensão.

Na Tabela 7, apresentamos os resultados conseguidos com as nossas implementações (GRASP, SS, Dual RAMP e PD-RAMP) e os resultados apresentados por Ahuja *et al.* [32] – VLSN, no conjunto de Holmberg *et al*.

	VLSN	GRASP	SS	Dual RAMP	PD-RAMP
% Desvio	0.03	0.20	0.03	0.02	0.04
T(s)	6.91	0.32	0.86	0.77	0.53
NSO	53	42	61	58	59

Tabela 6 - Análise dos resultados no conjunto de Holmberg *et al.*

Podemos observar que a pesquisa por dispersão, embora consiga o maior número de soluções ótimas, é mais exigente nos recursos computacionais do que ambos os algoritmos RAMP, e apresenta um desvio médio superior ao conseguido pelo Dual RAMP.

Neste conjunto de problemas, o algoritmo Dual RAMP consegue o melhor desvio médio, em tempos computacionais reduzidos. Outras abordagens (ver Tabela 5), conseguem este desvio médio, mas necessitam de tempos médios de execução significativamente superiores. É de notar, que o PD-RAMP atinge, em tempos computacionais consideravelmente mais reduzidos, um número superior de soluções ótimas com um desvio médio muito aproximado ao obtido pelo Dual RAMP.

Os dois algoritmos RAMP, conseguem, claramente, superar os resultados do algoritmo VLSN neste conjunto. O desvio médio do PD-RAMP é ligeiramente superior, mas os tempos computacionais são muito mais reduzidos e encontra mais seis soluções ótimas.

Nas Tabelas 8 e 9, apresentam-se os resultados atingidos com as nossas implementações (GRASP, SS, Dual RAMP e PD-RAMP) e os resultados reportados por Ahuja *et al.* [32] – VLSN, no conjunto de Beasley (pequena e grande dimensão, respetivamente).

	VLSN	GRASP	SS	Dual RAMP	PD-RAMP
% Desvio	0.00	0.04	0.04	0.00	0.01
T(s)	1.2	0.08	0.09	0.19	0.1
NSO	24	17	18	24	22

Tabela 7 - Análise dos resultados no conjunto de Beasley (pequena dimensão)

Para os problemas de pequena dimensão do conjunto de Beasley (Tabela 7), o algoritmo Dual RAMP, consegue, tal como o algoritmo VLSN, encontrar todas as soluções ótimas, mas em tempos computacionais inferiores. O algoritmo PD-RAMP, consegue em metade do tempo de execução requerido pelo Dual RAMP, um desvio médio ligeiramente superior.

Conclui-se que as abordagens RAMP, superam o desempenho dos restantes algoritmos apresentados.

	VLSN	GRASP	SS	Dual RAMP	PD-RAMP
% Desvio	0.08	0.72	0.05	0.12	0.08
T(s)	74.64	1.69	135.81	18.16	37.85
NSO	1	1	1	1	1

Tabela 8 - Análise dos resultados no conjunto de Beasley (grande dimensão)

Para os problemas de grande dimensão do conjunto de Beasley (Tabela 8), a pesquisa por dispersão obtém o melhor desvio médio, mas com tempos considerados elevados (embora competitivos quando comparados com heurísticas recentes, como por exemplo, Chen e Ting [38]).

O algoritmo PD-RAMP, embora mais exigente nos recursos computacionais, consegue resultados de qualidade superior aos obtidos pelo Dual RAMP. O PD-RAMP consegue um desvio médio igual ao obtido pelo VLNS, sendo referenciado na literatura como um dos métodos mais eficientes para instâncias de grande dimensão.

Considerando todos os conjuntos de teste, os resultados computacionais obtidos pelo algoritmo GRASP permitem concluir que a sua utilização na criação de soluções inicias é bastante vantajosa, pois consegue, com recursos computacionais muito baixos, obter um desvio médio muito atrativo, tendo em consideração o objetivo para o qual foi desenvolvido (obtenção de soluções inicias de boa qualidade). De notar, que consegue rivalizar com metaheurísticas mais sofisticadas nos conjuntos de Holmberg *et al.* e Beasley.

A pesquisa por dispersão proposta neste estudo, obtém para os problemas de grande dimensão do conjunto de Beasley, os melhores resultados da literatura, mas com tempos computacionais elevados. Para os restantes conjuntos apresenta uma aproximação significativa aos melhores resultados apresentados na literatura.

Os algoritmos Dual RAMP e PD-RAMP, propostos neste estudo, demonstram ser os algoritmos mais robustos para a resolução do SSCFLP. Ambos os algoritmos, apresentam um desempenho semelhante, obtendo resultados de excelente qualidade em tempos de execução muito reduzidos. Embora os resultados do Dual RAMP sejam de elevada qualidade, verifica-se que a exploração mais intensa da relação entre o espaço de soluções primal e dual, obtida nas versões mais sofisticadas do RAMP, através do PD-RAMP, produz resultados de qualidade superior.

Através dos resultados apresentados verifica-se, mais uma vez, que a metaheurística RAMP é capaz de produzir algoritmos de estado-da-arte para resolução de problemas complexos de otimização combinatória.

4. Conclusões e Trabalho Futuro

Grande parte dos organismos públicos e empresas privadas, já se confrontaram em alguma altura com problemas de localização de instalações, como por exemplo, na determinação da localização de novos hospitais que sirvam o maior número de pessoas ou na determinação da localização de novas unidades fabris que permitam os menores custos de transporte para os clientes.

A complexidade destes problemas, incentiva à investigação de métodos, principalmente metaheurísticas, que permitam a obtenção de soluções de qualidade elevada, em tempos computacionais aceitáveis, dado que a resolução destes problemas através de métodos exatos, principalmente para problemas de grande dimensão, pode ser impraticável a nível de tempo e recursos computacionais requeridos.

As metaheurísticas existentes, focam a sua atenção apenas num dos lados do espaço de soluções dos problemas de otimização combinatória, não sendo explorada de forma eficiente a relação existente entre o espaço primal (espaço de soluções do problema original) e o espaço dual (espaço de soluções conseguido através da relaxação do problema original). Rego [1], propõe uma metaheurística, denominada de RAMP (*Relaxation Adaptive Memory Programming*), que pretende criar algoritmos que, de forma iterativa, permitam a manipulação da informação que é obtida de ambos os lados do espaço de soluções, de forma a, eficientemente, explorar a relação primal e dual dos problemas.

A aplicação desta metaheurística, permitiu a obtenção de algoritmos de estado-da-arte, para diversos problemas de otimização combinatória.

Com este estudo, pretendeu-se verificar se a aplicação do método RAMP ao Problema de Localização de Instalações com Restrições de Capacidade e um Único Servidor (*Single Source Capacitated Facility Location Problem* - SSCFLP) fosse também capaz de rivalizar com outros métodos propostos para a resolução deste problema.

Neste trabalho, são apresentados dois novos algoritmos para a resolução do SSCFLP, ambos baseados no método RAMP, que designamos por Dual RAMP e PD-RAMP.

O algoritmo Dual RAMP segue a abordagem RAMP na sua versão mais simples. O método dual deste algoritmo, é baseado na resolução do dual lagrangeano do SSCFLP, através de otimização por subgradiente, O método primal, é composto por um GRASP e por uma pesquisa tabu.

Sendo a abordagem RAMP incremental, após a implementação da sua versão mais simples (Dual RAMP), foi implementado o PD-RAMP, para verificar se esta forma mais complexa da abordagem RAMP, poderia produzir resultados de qualidade superior.

O algoritmo PD-RAMP incorpora o Dual RAMP com um método primal baseado numa pesquisa por dispersão, com um conjunto de referência atualizado por ambos os lados primal e dual.

Os resultados obtidos pelo Dual RAMP e pelo PD-RAMP permitem concluir que a aplicação da metaheurística RAMP ao SSCFLP consegue resultados excelentes, obtendo soluções de elevada qualidade em tempos computacionais reduzidos. Os resultados permitem concluir ainda, que os algoritmos propostos são extremamente robustos providenciando um elevado nível de qualidade para problemas de pequena e grande dimensão.

Considerando que, após a determinação da localização de um conjunto de instalações, a determinação da afetação dos clientes recai num problema de afetação generalizado (*Generalized Assigment Problem* - GAP), um desenvolvimento natural deste estudo, passa pela implementação de um procedimento de resolução do GAP, para conseguir uma ótima (ou quase ótima) afetação dos clientes. Testes preliminares, sugerem que a inclusão de um procedimento robusto que consiga resolver instâncias GAP de elevada complexidade, conseguirá obter ainda melhores resultados nos problemas SSCFLP mais complexos (como os encontrados em Delmaire *et al.* [28]).

De forma a melhorar os resultados obtidos pelo método dual dos algoritmos RAMP propostos, alternativas à relaxação lagrangeana poderão ser experimentadas, como por exemplo, a relaxação paramétrica cruzada, introduzida por Rego [1].

Bibliografia

[1] C. Rego, “RAMP: A new metaheuristic framework for combinatorial optimization,” *Metaheuristic Optimization via Memory and Evolution*, 2005.

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

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

[4] R. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*. Physica-Verlag, 2009.

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

[6] C. Rego, F. Mathew, and F. Glover, “RAMP for the capacitated minimum spanning tree problem,” *Annals of Operations Research*, vol. 181, no. 1, pp. 661–681, Oct. 2010.

[7] C. Riley, C. Rego, and H. Li, “A simple dual-RAMP algorithm for resource constraint project scheduling,” *Proceedings of the 48th Annual Southeast*, 2010.

[8] P. Mirchandani and R. Francis, *Discrete location theory*. NewYork: Wiley, 1990.

[9] T. El-Ghazali, “Metaheuristics: from design to implementation,” *Jonh Wiley and Sons Inc., Chichester*, 2009.

[10] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[11] S. Sirenko, “Classification of heuristic methods in combinatorial optimization,” *Information Theories & Applications*, vol. 16, no. 4, pp. 303–322, 2009.

[12] T. Feo and M. . G. C. Resende, “Greedy randomized adaptive search procedures,” *Journal of Global Optimization*, pp. 109–133, 1995.

[13] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[14] F. Glover, “Heuristics for integer programming using surrogate constraints,” *Decision Sciences*, vol. 8, no. 1, pp. 156–166, 1977.

[15] M. Laguna, R. Martí, and R. Marti, *Scatter search: methodology and implementations in C*, vol. 1. Springer Netherlands, 2003.

[16] F. Glover, “Surrogate Constraints,” *Operations Research*, vol. 16, no. 4, pp. 741–749, Jul. 1968.

[17] J. E. Beasley, “Lagrangean heuristics for location problems,” *European Journal of Operational Research*, vol. 65, no. 3, pp. 383–399, Mar. 1993.

[18] M. L. Fisher, “The Lagrangian relaxation method for solving integer programming problems,” *Management science*, vol. 50, no. 12 Supplement, pp. 1861–1871, Dec. 2004.

[19] M. Daskin, *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley-Interscience, 1995.

[20] S. Tafazzoli and M. Mozafari, “Classification of Location Models and Location Softwares,” *Facility Location: Concepts, Models, Algorithms and Case Studies*, pp. 505–521, 2009.

[21] A. Neebe and M. R. Rao, “An algorithm for the fixed-charge assigning users to sources problem,” *Journal of the Operational Research Society*, vol. 34, no. 11, pp. 1107–1113, 1983.

[22] J. Barceló and J. Casanovas, “A heuristic Lagrangian relaxation algorithm for the capacitated plant location problem,” *European Journal of Operational Research*, vol. 15, no. 2, pp. 212–226, 1984.

[23] J. Klincewicz and H. Luss, “A Lagrangian relaxation heuristic for capacitated facility location with single-source constraints,” *Journal of the Operational Research Society*, vol. 37, no. 5, pp. 495–500, 1986.

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

[25] R. Sridharan, “A heuristic Lagrangean algorithm for the capacitated plant location problem - A comment,” *European journal of operational research*, vol. 23, no. 2, pp. 264–265, 1986.

[26] H. Pirkul, “Efficient algorithms for the capacitated concentrator location problem,” *Computers & Operations Research*, vol. 14, no. 3, pp. 197–208, 1987.

[27] H. Delmaire, J. A. Díaz, E. Fernández, and M. Ortega, “Comparing New Heuristics for the Pure Integer Capacitated Plant Location Problem,” Barcelona, 1997.

[28] H. Delmaire, J. A. Díaz, E. Fernández, and J. A. Diaz, “Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem,” *Infor*, vol. 37, no. 3, 1999.

[29] M. Rönnqvist, S. Tragantalerngsak, and J. Holt, “A repeated matching heuristic for the single-source capacitated facility location problem,” *European Journal of Operational Research*, vol. 116, no. 1, pp. 51–68, 1999.

[30] K. Holmberg, M. Rönnqvist, and D. Yuan, “An exact algorithm for the capacitated facility location problems with single sourcing,” *European Journal of Operational Research*, vol. 113, no. 3, pp. 544–559, Mar. 1999.

[31] K. Hindi and K. Pienkosz, “Efficient solution of large scale, single-source, capacitated plant location problems,” *Journal of the operational Research*, vol. 50, pp. 268–274, 1999.

[32] R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutellà, “A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location Problem,” *Management Science*, vol. 50, no. 6, pp. 749–760, Jun. 2004.

[33] M. J. Cortinhal and M. E. Captivo, “Genetic Algorithms for the Single Source Capacitated Location Problem: a Computational Study,” in *4th Metaheuristics International Conference 355*, 2001, no. 2, pp. 355–359.

[34] M. J. Cortinhal and M. E. Captivo, “Upper and lower bounds for the single source capacitated location problem,” *European Journal of Operational Research*, vol. 151, no. 2, pp. 333–351, Dec. 2003.

[35] I. A. Contreras and J. a. Díaz, “Scatter search for the single source capacitated facility location problem,” *Annals of Operations Research*, vol. 157, no. 1, pp. 73–89, Aug. 2007.

[36] K. Kumweang, “Solving a SSCFLP in a Supply Chain with ACO,” *Suranaree Journal of Science and*, vol. 12, no. 1, pp. 28–38, 2005.

[37] C.-H. Chen and C.-J. Ting, “Applying Multiple Ant Colony System to Solve Single Source Capacitated Facility Location Problem,” in *Ant Colony Optimization and Swarm Intelligence*, vol. 4150, no. 6, M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, Eds. Springer Berlin / Heidelberg, 2006, pp. 508–509.

[38] C.-H. Chen and C.-J. Ting, “Combining Lagrangian heuristic and Ant Colony System to solve the Single Source Capacitated Facility Location Problem,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 6, pp. 1099–1122, Nov. 2008.

[39] Y. Lina, S. U. N. Xu, and C. H. I. Tianhe, “A hybrid ant colony optimization algorithm with local search strategies to solve single source capacitated facility location problem,” *Engineering and Technology*, vol. 11, no. 9, pp. 1920–1924, 2011.

[40] Z. Yang, F. Chu, and H. Chen, “A cut-and-solve based algorithm for the single-source capacitated facility location problem,” *European Journal of Operational Research*, vol. 221, no. 3, pp. 521–532, Sep. 2012.

[41] S. Climer and W. Zhang, “Cut-and-solve: An iterative search strategy for combinatorial optimization problems,” *Artificial Intelligence*, vol. 170, no. 8–9, pp. 714–738, Jun. 2006.

[42] S. Martello, D. Pisinger, and P. Toth, “Dynamic programming and strong bounds for the 0-1 knapsack problem,” *Management Science*, vol. 45, no. 3, pp. 414–425, Mar. 1999.

[43] D. Pisinger, “Where are the hard knapsack problems?,” *Computers & Operations Research*, vol. 32, no. 9, pp. 2271–2284, Sep. 2005.

[44] D. Pisinger, “David Pisinger’s optimization codes.” [Online]. Url: http://www.diku.dk/hjemmesider/ansatte/pisinger/codes.html. [Último acesso: 28-Nov-2012].

[45] J. E. Beasley, “OR-Library: distributing test problems by electronic mail,” *Journal of the Operational Research Society*, vol. 41, pp. 1069–1072, 1990.

Anexos

A. Resultados Computacionais - GRASP

Problema	Z^*	Z	Gap	T(s)
1	8848,0	8848,0	0,00	0,03
2	7913,0	7913,0	0,00	0,03
3	9314,0	9314,0	0,00	0,05
4	10714,0	10714,0	0,00	0,02
5	8838,0	8838,0	0,00	0,02
6	7777,0	7777,0	0,00	0,05
7	9488,0	9488,0	0,00	0,04
8	11088,0	11088,0	0,00	0,02
9	8462,0	8462,0	0,00	0,02
10	7617,0	7617,0	0,00	0,08
11	8932,0	8932,0	0,00	0,03
12	10132,0	10132,0	0,00	0,02
13	8252,0	8252,0	0,00	0,06
14	7137,0	7137,0	0,00	0,07
15	8808,0	8808,0	0,00	0,05
16	10408,0	10408,0	0,00	0,02
17	8227,0	8227,0	0,00	0,07
18	7125,0	7125,0	0,00	0,06
19	8886,0	8886,0	0,00	0,14
20	10486,0	10486,0	0,00	0,04
21	8068,0	8068,0	0,00	0,12
22	7092,0	7092,0	0,00	0,12
23	8746,0	8746,0	0,00	0,06
24	10273,0	10273,0	0,00	0,04

Tabela 1 - Resultados do GRASP para o subconjunto 1 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
25	11630,0	11636,0	0,05	0,83
26	10771,0	10786,0	0,14	0,89
27	12322,0	12378,0	0,45	0,86
28	13722,0	13898,0	1,28	0,80
29	12371,0	12399,0	0,23	0,88
30	11331,0	11415,0	0,74	0,90
31	13331,0	13385,0	0,41	0,93
32	15331,0	15529,0	1,29	0,90
33	11629,0	11629,0	0,00	0,92
34	10632,0	10638,0	0,06	0,94
35	12232,0	12238,0	0,05	0,93
36	13832,0	13832,0	0,00	0,93
37	11258,0	11258,0	0,00	1,11
38	10551,0	10551,0	0,00	1,12
39	11824,0	11824,0	0,00	1,07
40	13024,0	13024,0	0,00	1,06

Tabela 2 - Resultados do GRASP para o subconjunto 2 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
41	6589,0	6604,0	0,23	0,03
42	5663,0	5663,0	0,00	0,06
43	5214,0	5214,0	0,00	0,11
44	7028,0	7034,0	0,09	0,04
45	6251,0	6251,0	0,00	0,08
46	5651,0	5651,0	0,00	0,11
47	6228,0	6257,0	0,47	0,05
48	5596,0	5596,0	0,00	0,10
49	5302,0	5302,0	0,00	0,21
50	8741,0	8788,0	0,54	0,06
51	7414,0	7460,0	0,62	0,11
52	9178,0	9183,0	0,05	0,05
53	8531,0	8536,0	0,06	0,08
54	8777,0	8806,0	0,33	0,05
55	7654,0	7685,0	0,41	0,09

Tabela 3 - Resultados do GRASP para o subconjunto 3 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
56	21103,0	21186,0	0,39	0,25
57	26039,0	26375,0	1,29	0,22
58	37239,0	37530,0	0,78	0,20
59	27282,0	27593,0	1,14	0,21
60	20534,0	20534,0	0,00	0,44
61	24454,0	24454,0	0,00	0,26
62	32643,0	32768,0	0,38	0,26
63	25105,0	25149,0	0,18	0,32
64	20530,0	20530,0	0,00	0,57
65	24445,0	24445,0	0,00	0,41
66	31415,0	31636,0	0,70	0,36
67	24848,0	24904,0	0,23	0,28
68	20538,0	20538,0	0,00	0,35
69	24532,0	24532,0	0,00	0,32
70	32321,0	32825,0	1,56	0,27
71	25540,0	25648,0	0,42	0,28

Tabela 4 - Resultados do GRASP para o subconjunto 4 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
1	2014,0	2014,0	0,00	0,01
2	4251,0	4417,0	3,91	0,01
3	6051,0	6137,0	1,42	0,03
4	7168,0	7373,0	2,86	0,03
5	4551,0	4580,0	0,64	0,05
6	2269,0	2277,0	0,35	0,01

Tabela 5 - Resultados do GRASP para o subconjunto 1 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
7	4366,0	4452,0	1,97	0,02
8	7926,0	8277,0	4,43	0,03
9	2480,0	2647,0	6,73	0,02
10	23112,0	24435,0	5,72	0,05
11	3447,0	3536,0	2,58	0,01
12	3711,0	3739,0	0,76	0,02
13	3760,0	3773,0	0,35	0,04
14	5965,0	6139,0	2,92	0,01
15	7816,0	8158,0	4,38	0,03
16	11543,0	12300,0	6,56	0,01
17	9884,0	9953,0	0,70	0,03

Tabela 6 - Resultados do GRASP para o subconjunto 2 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
18	15607,0	16303,0	4,46	0,03
19	18683,0	19016,0	1,78	0,03
20	26561,0	27764,0	4,53	0,07
21	7295,0	7602,0	4,21	0,03
22	3271,0	3401,0	3,97	0,02
23	6036,0	6402,0	6,06	0,02
24	6327,0	6413,0	1,36	0,04
25	8947,0	8978,0	0,35	0,05

Tabela 7 - Resultados do GRASP para o subconjunto 3 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
26	4448,0	4613,0	3,71	0,04
27	10921,0	11002,0	0,74	0,04
28	11117,0	11182,0	0,59	0,03
29	9832,0	9950,0	1,20	0,04
30	10816,0	11332,0	4,77	0,02
31	4466,0	4590,0	2,78	0,04
32	9881,0	10051,0	1,72	0,04
33	39463,0	41453,0	5,04	0,04

Tabela 8 - Resultados do GRASP para o subconjunto 4 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
34	4701,0	4718,0	0,36	0,10
35	5456,0	5663,0	3,79	0,11
36	16781,0	17265,0	2,88	0,08
37	14668,0	14840,0	1,17	0,08
38	47249,0	49747,0	5,29	0,06
39	41007,0	41240,0	0,57	0,07
40	61633,0	64681,0	4,95	0,06
41	17246,0	17246,0	0,00	0,09

Tabela 9 - Resultados do GRASP para o subconjunto 5 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
42	7887,0	7964,0	0,98	0,12
43	5114,0	5203,0	1,74	0,13
44	36022,0	37890,0	5,19	0,07
45	17676,0	17701,0	0,14	0,11
46	48701,0	50101,0	2,88	0,11
47	66230,0	68410,0	3,29	0,09
48	58964,0	59065,0	0,17	0,14
49	79614,0	82658,0	3,82	0,11

Tabela 10 - Resultados do GRASP para o subconjunto 6 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
50	5937,0	6010,0	1,23	0,25
51	9060,0	9151,0	1,00	0,19
52	34652,0	35040,0	1,12	0,14
53	30038,0	31455,0	4,72	0,13
54	43853,0	47304,0	7,87	0,23
55	69610,0	71065,0	2,09	0,12
56	64474,0	69103,0	7,18	0,20
57	49791,0	49811,0	0,04	0,22

Tabela 11 - Resultados do GRASP para o subconjunto 7 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
cap61	932615,75	932615,75	0,00	0,06
cap62	977799,40	977799,40	0,00	0,06
cap63	1014099,61	1014099,61	0,00	0,05
cap64	1053197,44	1055801,24	0,25	0,08
cap71	932615,75	932615,75	0,00	0,05
cap72	977799,40	977799,40	0,00	0,04
cap73	1010641,45	1012476,98	0,18	0,02
cap74	1034976,97	1034976,98	0,00	0,02

Tabela 12 - Resultados do GRASP para o subconjunto 1 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap91	796648,44	796648,44	0,00	0,09
cap92	858109,32	858109,33	0,00	0,07
cap93	900760,11	900760,11	0,00	0,06
cap94	950608,42	950608,43	0,00	0,05
cap101	796648,44	797508,73	0,11	0,08
cap102	854704,20	854704,20	0,00	0,06
cap103	893782,11	895027,19	0,14	0,03
cap104	928941,75	928941,75	0,00	0,03

Tabela 13 - Resultados do GRASP para o subconjunto 2 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap121	793439,56	793439,56	0,00	0,13
cap122	854900,45	854900,45	0,00	0,31
cap123	898266,08	898266,08	0,00	0,11
cap124	950608,43	950644,44	0,00	0,08
cap131	793439,56	794299,85	0,11	0,17
cap132	851495,32	851495,33	0,00	0,12
cap133	893076,71	894095,76	0,11	0,09
cap134	928941,75	928941,75	0,00	0,07

Tabela 14 - Resultados do GRASP para o subconjunto 3 de Beasley

Problema	Z^*	Z	Gap	T(s)
capa1	19240822,45	19320651,97	0,42	1,49
capa2	18438046,45	18503722,68	0,36	1,36
capa3	17765201,95	17765201,95	0,00	1,23
capa4	17160439,01	17161219,58	0,01	1,63
capb1	13656379,58	13883424,04	1,66	1,72
capb2	13361927,45	13585059,17	1,67	1,50
capb3	13198556,43	13450203,57	1,91	1,49
capb4	13082516,50	13148146,01	0,50	1,50
capc1	11646596,97	11827250,72	1,55	1,55
capc2	11570340,29	11588440,64	0,16	2,86
capc3	11518743,74	11567041,40	0,42	2,32
capc4	11505767,39	11505861,86	0,00	1,67

Tabela 15 - Resultados do GRASP para o subconjunto 4 de Beasley

B. Resultados Computacionais - Pesquisa por Dispersão

Problema	Z^*	Z	Gap	T(s)
1	8848,0	8848,0	0,00	0,05
2	7913,0	7913,0	0,00	0,04
3	9314,0	9314,0	0,00	0,05
4	10714,0	10714,0	0,00	0,12
5	8838,0	8838,0	0,00	0,03
6	7777,0	7777,0	0,00	0,03
7	9488,0	9488,0	0,00	0,02
8	11088,0	11088,0	0,00	0,09
9	8462,0	8462,0	0,00	0,04
10	7617,0	7617,0	0,00	0,07
11	8932,0	8932,0	0,00	0,03
12	10132,0	10132,0	0,00	0,06
13	8252,0	8252,0	0,00	0,12
14	7137,0	7137,0	0,00	0,08
15	8808,0	8808,0	0,00	0,18
16	10408,0	10408,0	0,00	0,19
17	8227,0	8227,0	0,00	0,17
18	7125,0	7125,0	0,00	0,14
19	8886,0	8886,0	0,00	0,33
20	10486,0	10486,0	0,00	0,25
21	8068,0	8068,0	0,00	0,17
22	7092,0	7092,0	0,00	0,15
23	8746,0	8746,0	0,00	0,23
24	10273,0	10273,0	0,00	0,25

Tabela 1 - Resultados da pesquisa por dispersão para o subconjunto 1 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
25	11630,0	11630,0	0,00	1,16
26	10771,0	10771,0	0,00	1,27
27	12322,0	12322,0	0,00	1,25
28	13722,0	13722,0	0,00	1,44
29	12371,0	12371,0	0,00	3,00
30	11331,0	11339,0	0,07	2,55
31	13331,0	13331,0	0,00	2,10
32	15331,0	15338,0	0,05	2,20
33	11629,0	11629,0	0,00	1,23
34	10632,0	10632,0	0,00	1,20
35	12232,0	12232,0	0,00	1,19
36	13832,0	13832,0	0,00	1,63
37	11258,0	11258,0	0,00	1,51
38	10551,0	10551,0	0,00	1,19
39	11824,0	11824,0	0,00	1,56
40	13024,0	13024,0	0,00	1,56

Tabela 2 - Resultados da pesquisa por dispersão para o subconjunto 2 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
41	6589,0	6589,0	0,00	0,38
42	5663,0	5663,0	0,00	0,70
43	5214,0	5214,0	0,00	0,24
44	7028,0	7028,0	0,00	0,10
45	6251,0	6251,0	0,00	0,14
46	5651,0	5651,0	0,00	0,31
47	6228,0	6257,0	0,47	0,08
48	5596,0	5596,0	0,00	0,14
49	5302,0	5302,0	0,00	0,47
50	8741,0	8741,0	0,00	0,48
51	7414,0	7458,0	0,59	0,22
52	9178,0	9180,0	0,02	0,18
53	8531,0	8531,0	0,00	0,15
54	8777,0	8777,0	0,00	0,15
55	7654,0	7654,0	0,00	0,15

Tabela 3 - Resultados da pesquisa por dispersão para o subconjunto 3 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
56	21103,0	21134,0	0,15	2,11
57	26039,0	26039,0	0,00	2,76
58	37239,0	37239,0	0,00	2,65
59	27282,0	27282,0	0,00	2,13
60	20534,0	20534,0	0,00	0,41
61	24454,0	24454,0	0,00	0,65
62	32643,0	32643,0	0,00	1,77
63	25105,0	25105,0	0,00	1,69
64	20530,0	20530,0	0,00	0,57
65	24445,0	24445,0	0,00	1,45
66	31415,0	31457,0	0,13	2,84
67	24848,0	24851,0	0,01	2,12
68	20538,0	20538,0	0,00	0,40
69	24532,0	24532,0	0,00	0,67
70	32321,0	32393,0	0,22	2,35
71	25540,0	25560,0	0,08	3,47

Tabela 4 - Resultados da pesquisa por dispersão para o subconjunto 4 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
1	2014,0	2014,0	0,00	0,12
2	4251,0	4251,0	0,00	0,12
3	6051,0	6051,0	0,00	0,16
4	7168,0	7272,0	1,45	0,15
5	4551,0	4551,0	0,00	0,15
6	2269,0	2269,0	0,00	0,11

Tabela 5 - Resultados da pesquisa por dispersão para o subconjunto 1 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
7	4366,0	4406,0	0,92	0,15
8	7926,0	8055,0	1,63	0,32
9	2480,0	2480,0	0,00	0,43
10	23112,0	23112,0	0,00	0,33
11	3447,0	3471,0	0,70	0,42
12	3711,0	3731,0	0,54	0,20
13	3760,0	3773,0	0,35	0,18
14	5965,0	5984,0	0,32	0,22
15	7816,0	7832,0	0,21	0,33
16	11543,0	11558,0	0,13	0,36
17	9884,0	9935,0	0,52	0,06

Tabela 6 - Resultados da pesquisa por dispersão para o subconjunto 2 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
18	15607,0	15908,0	1,93	0,72
19	18683,0	18683,0	0,00	0,43
20	26561,0	26574,0	0,05	1,18
21	7295,0	7369,0	1,01	0,84
22	3271,0	3271,0	0,00	0,31
23	6036,0	6042,0	0,10	0,33
24	6327,0	6327,0	0,00	0,22
25	8947,0	8950,0	0,03	0,13

Tabela 7 - Resultados da pesquisa por dispersão para o subconjunto 3 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
26	4448,0	4458,0	0,23	1,11
27	10921,0	10974,0	0,49	0,41
28	11117,0	11117,0	0,00	0,36
29	9832,0	9850,0	0,18	0,69
30	10816,0	10864,0	0,44	1,18
31	4466,0	4513,0	1,05	0,34
32	9881,0	9881,0	0,00	0,19
33	39463,0	39581,0	0,30	1,46

Tabela 8 - Resultados da pesquisa por dispersão para o subconjunto 4 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
34	4701,0	4709,0	0,17	0,73
35	5456,0	5456,0	0,00	0,35
36	16781,0	17243,0	2,75	0,28
37	14668,0	14757,0	0,61	0,49
38	47249,0	47263,0	0,03	1,26
39	41007,0	41073,0	0,16	1,89
40	61633,0	61655,0	0,04	2,29
41	17246,0	17246,0	0,00	0,39

Tabela 9 - Resultados da pesquisa por dispersão para o subconjunto 5 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
42	7887,0	7890,0	0,04	0,70
43	5114,0	5161,0	0,92	0,71
44	36022,0	36337,0	0,87	2,99
45	17676,0	17678,0	0,01	0,45
46	48701,0	49157,0	0,94	2,30
47	66230,0	66263,0	0,05	2,45
48	58964,0	59045,0	0,14	0,85
49	79614,0	79746,0	0,17	2,25

Tabela 10 - Resultados da pesquisa por dispersão para o subconjunto 6 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
50	5937,0	5952,0	0,25	0,66
51	9060,0	9123,0	0,70	0,74
52	34652,0	34680,0	0,08	2,98
53	30038,0	30045,0	0,02	2,93
54	43853,0	43854,0	0,00	2,44
55	69610,0	69916,0	0,44	3,30
56	64474,0	64474,0	0,00	0,95
57	49791,0	49791,0	0,00	0,59

Tabela 11 - Resultados da pesquisa por dispersão para o subconjunto 7 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
cap61	932615,75	932615,75	0,00	0,06
cap62	977799,40	977799,40	0,00	0,06
cap63	1014099,61	1014099,61	0,00	0,12
cap64	1053197,44	1055801,24	0,25	0,09
cap71	932615,75	932615,75	0,00	0,04
cap72	977799,40	977799,40	0,00	0,04
cap73	1010641,45	1012476,98	0,18	0,02
cap74	1034976,97	1034976,98	0,00	0,02

Tabela 12 - Resultados da pesquisa por dispersão para o subconjunto 1 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap91	796648,44	796648,44	0,00	0,10
cap92	858109,32	858109,33	0,00	0,09
cap93	900760,11	900760,11	0,00	0,10
cap94	950608,42	950608,43	0,00	0,20
cap101	796648,44	797508,73	0,11	0,07
cap102	854704,20	854704,20	0,00	0,07
cap103	893782,11	895027,19	0,14	0,03
cap104	928941,75	928941,75	0,00	0,03

Tabela 13 - Resultados da pesquisa por dispersão para o subconjunto 2 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap121	793439,56	793439,56	0,00	0,16
cap122	854900,45	854900,45	0,00	0,16
cap123	898266,08	898266,08	0,00	0,13
cap124	950608,43	950608,43	0,00	0,19
cap131	793439,56	794299,85	0,11	0,15
cap132	851495,32	851495,33	0,00	0,12
cap133	893076,71	894095,76	0,11	0,10
cap134	928941,75	928941,75	0,00	0,06

Tabela 14 - Resultados da pesquisa por dispersão para o subconjunto 3 de Beasley

Problema	Z^*	Z	Gap	T(s)
capa1	19240822,45	19244318,68	0,02	237,17
capa2	18438046,45	18455395,78	0,09	124,43
capa3	17765201,95	17765201,95	0,00	75,58
capa4	17160439,01	17160612,23	0,00	85,44
capb1	13656379,58	13658606,54	0,02	127,80
capb2	13361927,45	13367492,54	0,04	126,64
capb3	13198556,43	13229733,89	0,24	138,46
capb4	13082516,50	13110199,94	0,21	164,20
capc1	11646596,97	11647979,03	0,01	129,18
capc2	11570340,29	11570437,68	0,00	126,66
capc3	11518743,74	11519924,61	0,01	233,77
capc4	11505767,39	11505861,86	0,00	60,46

Tabela 15 - Resultados da pesquisa por dispersão para o subconjunto 4 de Beasley

C. Resultados Computacionais - Dual RAMP

Problema	Z^*	Z	Gap	T(s)
1	8848,0	8848,0	0,00	0,02
2	7913,0	7913,0	0,00	0,17
3	9314,0	9314,0	0,00	0,03
4	10714,0	10714,0	0,00	0,26
5	8838,0	8838,0	0,00	0,02
6	7777,0	7777,0	0,00	0,02
7	9488,0	9488,0	0,00	0,19
8	11088,0	11088,0	0,00	0,25
9	8462,0	8462,0	0,00	0,16
10	7617,0	7617,0	0,00	0,19
11	8932,0	8932,0	0,00	0,02
12	10132,0	10132,0	0,00	0,17
13	8252,0	8252,0	0,00	0,04
14	7137,0	7137,0	0,00	0,06
15	8808,0	8808,0	0,00	0,04
16	10408,0	10408,0	0,00	0,38
17	8227,0	8227,0	0,00	0,06
18	7125,0	7125,0	0,00	0,06
19	8886,0	8886,0	0,00	0,43
20	10486,0	10486,0	0,00	0,43
21	8068,0	8068,0	0,00	0,12
22	7092,0	7092,0	0,00	0,12
23	8746,0	8746,0	0,00	0,12
24	10273,0	10273,0	0,00	0,40

Tabela 1 - Resultados do Dual RAMP para o subconjunto 1 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
25	11630,0	11630,0	0,00	1,88
26	10771,0	10771,0	0,00	1,78
27	12322,0	12324,0	0,02	1,98
28	13722,0	13722,0	0,00	1,98
29	12371,0	12371,0	0,00	2,20
30	11331,0	11345,0	0,12	2,16
31	13331,0	13345,0	0,11	2,29
32	15331,0	15343,0	0,08	2,29
33	11629,0	11629,0	0,00	1,98
34	10632,0	10632,0	0,00	1,38
35	12232,0	12232,0	0,00	2,02
36	13832,0	13832,0	0,00	2,05
37	11258,0	11258,0	0,00	1,17
38	10551,0	10551,0	0,00	1,17
39	11824,0	11824,0	0,00	1,15
40	13024,0	13024,0	0,00	1,14

Tabela 2 - Resultados do Dual RAMP para o subconjunto 2 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
41	6589,0	6589,0	0,00	0,32
42	5663,0	5663,0	0,00	0,54
43	5214,0	5214,0	0,00	0,11
44	7028,0	7028,0	0,00	0,06
45	6251,0	6251,0	0,00	0,11
46	5651,0	5651,0	0,00	0,64
47	6228,0	6228,0	0,00	0,04
48	5596,0	5596,0	0,00	0,43
49	5302,0	5302,0	0,00	0,23
50	8741,0	8741,0	0,00	0,45
51	7414,0	7453,0	0,53	0,69
52	9178,0	9178,0	0,00	0,14
53	8531,0	8531,0	0,00	0,11
54	8777,0	8777,0	0,00	0,05
55	7654,0	7654,0	0,00	0,61

Tabela 3 - Resultados do Dual RAMP para o subconjunto 3 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
56	21103,0	21138,0	0,17	1,41
57	26039,0	26054,0	0,06	1,91
58	37239,0	37329,0	0,24	2,03
59	27282,0	27308,0	0,10	2,02
60	20534,0	20534,0	0,00	0,36
61	24454,0	24454,0	0,00	0,30
62	32643,0	32673,0	0,09	1,89
63	25105,0	25105,0	0,00	1,07
64	20530,0	20530,0	0,00	0,56
65	24445,0	24445,0	0,00	0,45
66	31415,0	31420,0	0,02	1,91
67	24848,0	24873,0	0,10	1,35
68	20538,0	20538,0	0,00	0,34
69	24532,0	24532,0	0,00	0,37
70	32321,0	32352,0	0,10	1,50
71	25540,0	25540,0	0,00	0,61

Tabela 4 - Resultados do Dual RAMP para o subconjunto 4 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
1	2014,0	2014,0	0,00	0,20
2	4251,0	4251,0	0,00	0,19
3	6051,0	6073,0	0,36	0,21
4	7168,0	7276,0	1,51	0,20
5	4551,0	4567,0	0,35	0,21
6	2269,0	2269,0	0,00	0,19

Tabela 5 - Resultados do Dual RAMP para o subconjunto 1 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
7	4366,0	4393,0	0,62	0,37
8	7926,0	8172,0	3,10	0,34
9	2480,0	2480,0	0,00	0,32
10	23112,0	23114,0	0,01	0,35
11	3447,0	3471,0	0,70	0,33
12	3711,0	3716,0	0,14	0,31
13	3760,0	3773,0	0,35	0,32
14	5965,0	6129,0	2,75	0,36
15	7816,0	7828,0	0,15	0,34
16	11543,0	11555,0	0,10	0,34
17	9884,0	9935,0	0,52	0,34

Tabela 6 - Resultados do Dual RAMP para o subconjunto 2 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
18	15607,0	15805,0	1,27	0,58
19	18683,0	18806,0	0,66	0,54
20	26561,0	26775,0	0,81	0,59
21	7295,0	7414,0	1,63	0,56
22	3271,0	3271,0	0,00	0,51
23	6036,0	6051,0	0,25	0,55
24	6327,0	6352,0	0,40	0,53
25	8947,0	8947,0	0,00	0,51

Tabela 7 - Resultados do Dual RAMP para o subconjunto 3 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
26	4448,0	4465,0	0,38	0,57
27	10921,0	10977,0	0,51	0,68
28	11117,0	11138,0	0,19	0,64
29	9832,0	9896,0	0,65	0,64
30	10816,0	11001,0	1,71	0,67
31	4466,0	4513,0	1,05	0,59
32	9881,0	10051,0	1,72	0,57
33	39463,0	39605,0	0,36	0,64

Tabela 8 - Resultados do Dual RAMP para o subconjunto 4 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
34	4701,0	4718,0	0,36	0,90
35	5456,0	5456,0	0,00	0,80
36	16781,0	17243,0	2,75	1,04
37	14668,0	14796,0	0,87	0,96
38	47249,0	47316,0	0,14	1,11
39	41007,0	41098,0	0,22	1,13
40	61633,0	62160,0	0,86	1,12
41	17246,0	17246,0	0,00	1,08

Tabela 9 - Resultados do Dual RAMP para o subconjunto 5 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
42	7887,0	7913,0	0,33	1,17
43	5114,0	5192,0	1,53	0,96
44	36022,0	36805,0	2,17	1,31
45	17676,0	17687,0	0,06	1,21
46	48701,0	49481,0	1,60	1,31
47	66230,0	66716,0	0,73	1,30
48	58964,0	59065,0	0,17	1,27
49	79614,0	79990,0	0,47	1,24

Tabela 10 - Resultados do Dual RAMP para o subconjunto 6 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
50	5937,0	5983,0	0,78	1,21
51	9060,0	9130,0	0,77	1,30
52	34652,0	34981,0	0,95	1,48
53	30038,0	30269,0	0,77	1,47
54	43853,0	44013,0	0,37	1,60
55	69610,0	70226,0	0,89	1,44
56	64474,0	64483,0	0,01	1,23
57	49791,0	49792,0	0,00	1,35

Tabela 11 - Resultados do Dual RAMP para o subconjunto 7 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
cap61	932615,75	932615,75	0,00	0,06
cap62	977799,40	977799,40	0,00	0,06
cap63	1014099,61	1014099,61	0,00	0,28
cap64	1053197,44	1053197,44	0,00	0,33
cap71	932615,75	932615,75	0,00	0,04
cap72	977799,40	977799,40	0,00	0,04
cap73	1010641,45	1010641,45	0,00	0,03
cap74	1034976,97	1034976,98	0,00	0,03

Tabela 12 - Resultados do Dual RAMP para o subconjunto 1 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap91	796648,44	796648,44	0,00	0,09
cap92	858109,32	858109,33	0,00	0,10
cap93	900760,11	900760,11	0,00	0,45
cap94	950608,42	950608,43	0,00	0,48
cap101	796648,44	796648,44	0,00	0,09
cap102	854704,20	854704,20	0,00	0,06
cap103	893782,11	893782,11	0,00	0,07
cap104	928941,75	928941,75	0,00	0,04

Tabela 13 - Resultados do Dual RAMP para o subconjunto 2 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap121	793439,56	793439,56	0,00	0,16
cap122	854900,45	854900,45	0,00	0,16
cap123	898266,08	898266,08	0,00	0,83
cap124	950608,43	950608,43	0,00	0,86
cap131	793439,56	793439,56	0,00	0,16
cap132	851495,32	851495,33	0,00	0,15
cap133	893076,71	893076,71	0,00	0,10
cap134	928941,75	928941,75	0,00	0,08

Tabela 14 – Resultados do Dual RAMP para o subconjunto 3 de Beasley

Problema	Z^*	Z	Gap	T(s)
capa1	19240822,45	19281453,53	0,21	26,62
capa2	18438046,45	18441462,00	0,02	13,77
capa3	17765201,95	17765201,95	0,00	13,70
capa4	17160439,01	17161219,58	0,01	8,11
capb1	13656379,58	13690040,81	0,25	17,15
capb2	13361927,45	13396980,64	0,26	27,06
capb3	13198556,43	13234714,53	0,27	23,58
capb4	13082516,50	13103435,25	0,16	21,94
capc1	11646596,97	11655773,49	0,08	14,89
capc2	11570340,29	11573433,94	0,03	21,64
capc3	11518743,74	11534089,24	0,13	14,97
capc4	11505767,39	11505861,86	0,00	14,47

Tabela 15 - Resultados do Dual RAMP para o subconjunto 4 de Beasley

D. Resultados Computacionais - PD-RAMP

Problema	Z^*	Z	Gap	T(s)
1	8848,0	8848,0	0,00	0,03
2	7913,0	7913,0	0,00	0,07
3	9314,0	9314,0	0,00	0,03
4	10714,0	10714,0	0,00	0,04
5	8838,0	8838,0	0,00	0,03
6	7777,0	7777,0	0,00	0,03
7	9488,0	9488,0	0,00	0,04
8	11088,0	11088,0	0,00	0,03
9	8462,0	8462,0	0,00	0,03
10	7617,0	7617,0	0,00	0,08
11	8932,0	8932,0	0,00	0,06
12	10132,0	10132,0	0,00	0,03
13	8252,0	8252,0	0,00	0,07
14	7137,0	7137,0	0,00	0,07
15	8808,0	8808,0	0,00	0,09
16	10408,0	10408,0	0,00	0,07
17	8227,0	8227,0	0,00	0,16
18	7125,0	7125,0	0,00	0,08
19	8886,0	8886,0	0,00	0,22
20	10486,0	10486,0	0,00	0,27
21	8068,0	8068,0	0,00	0,17
22	7092,0	7092,0	0,00	0,13
23	8746,0	8746,0	0,00	0,23
24	10273,0	10273,0	0,00	0,27

Tabela 1 - Resultados do PD-RAMP para o subconjunto 1 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
25	11630,0	11630,0	0,00	1,03
26	10771,0	10773,0	0,02	1,02
27	12322,0	12371,0	0,40	1,12
28	13722,0	13722,0	0,00	1,02
29	12371,0	12371,0	0,00	1,15
30	11331,0	11394,0	0,56	1,14
31	13331,0	13331,0	0,00	1,48
32	15331,0	15331,0	0,00	1,68
33	11629,0	11629,0	0,00	1,24
34	10632,0	10632,0	0,00	1,14
35	12232,0	12232,0	0,00	1,13
36	13832,0	13832,0	0,00	1,06
37	11258,0	11258,0	0,00	1,33
38	10551,0	10551,0	0,00	1,26
39	11824,0	11824,0	0,00	1,16
40	13024,0	13024,0	0,00	1,16

Tabela 2 - Resultados do PD-RAMP para o subconjunto 2 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
41	6589,0	6589,0	0,00	0,28
42	5663,0	5663,0	0,00	0,95
43	5214,0	5214,0	0,00	0,49
44	7028,0	7028,0	0,00	0,06
45	6251,0	6251,0	0,00	0,28
46	5651,0	5651,0	0,00	0,33
47	6228,0	6228,0	0,00	0,07
48	5596,0	5596,0	0,00	0,24
49	5302,0	5302,0	0,00	0,47
50	8741,0	8788,0	0,54	0,11
51	7414,0	7453,0	0,53	0,36
52	9178,0	9180,0	0,02	0,09
53	8531,0	8531,0	0,00	0,15
54	8777,0	8777,0	0,00	0,06
55	7654,0	7685,0	0,41	0,18

Tabela 3 - Resultados do PD-RAMP para o subconjunto 3 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
56	21103,0	21179,0	0,36	0,73
57	26039,0	26039,0	0,00	0,93
58	37239,0	37239,0	0,00	1,39
59	27282,0	27304,0	0,08	0,73
60	20534,0	20534,0	0,00	0,56
61	24454,0	24454,0	0,00	0,39
62	32643,0	32643,0	0,00	0,86
63	25105,0	25105,0	0,00	0,67
64	20530,0	20530,0	0,00	0,59
65	24445,0	24445,0	0,00	0,49
66	31415,0	31415,0	0,00	1,55
67	24848,0	24852,0	0,02	0,72
68	20538,0	20538,0	0,00	0,45
69	24532,0	24532,0	0,00	0,41
70	32321,0	32393,0	0,22	1,01
71	25540,0	25560,0	0,08	0,74

Tabela 4 - Resultados do PD-RAMP para o subconjunto 4 de Holmberg *et al.*

Problema	Z^*	Z	Gap	T(s)
1	2014,0	2014,0	0,00	0,16
2	4251,0	4251,0	0,00	0,27
3	6051,0	6051,0	0,00	0,37
4	7168,0	7272,0	1,45	0,27
5	4551,0	4567,0	0,35	0,17
6	2269,0	2275,0	0,26	0,09

Tabela 5 - Resultados do PD-RAMP para o subconjunto 1 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
7	4366,0	4416,0	1,15	0,17
8	7926,0	8172,0	3,10	0,39
9	2480,0	2480,0	0,00	0,59
10	23112,0	23114,0	0,01	0,38
11	3447,0	3471,0	0,70	0,61
12	3711,0	3716,0	0,14	0,29
13	3760,0	3773,0	0,35	0,27
14	5965,0	6129,0	2,75	0,27
15	7816,0	7831,0	0,19	0,31
16	11543,0	11558,0	0,13	0,47
17	9884,0	9935,0	0,52	0,06

Tabela 6 - Resultados do PD-RAMP para o subconjunto 2 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
18	15607,0	15633,0	0,17	1,35
19	18683,0	18684,0	0,01	0,36
20	26561,0	26743,0	0,69	1,54
21	7295,0	7298,0	0,04	1,32
22	3271,0	3274,0	0,09	0,27
23	6036,0	6051,0	0,25	0,39
24	6327,0	6328,0	0,02	0,44
25	8947,0	8950,0	0,03	0,08

Tabela 7 - Resultados do PD-RAMP para o subconjunto 3 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
26	4448,0	4467,0	0,43	0,57
27	10921,0	10974,0	0,49	0,65
28	11117,0	11128,0	0,10	0,18
29	9832,0	9834,0	0,02	0,57
30	10816,0	10985,0	1,56	1,13
31	4466,0	4519,0	1,19	0,40
32	9881,0	10039,0	1,60	0,18
33	39463,0	39605,0	0,36	2,00

Tabela 8 - Resultados do PD-RAMP para o subconjunto 4 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
34	4701,0	4712,0	0,23	0,42
35	5456,0	5456,0	0,00	0,40
36	16781,0	17243,0	2,75	0,17
37	14668,0	14677,0	0,06	0,62
38	47249,0	47255,0	0,01	1,59
39	41007,0	41066,0	0,14	1,69
40	61633,0	61697,0	0,10	1,95
41	17246,0	17246,0	0,00	0,17

Tabela 9 - Resultados do PD-RAMP para o subconjunto 5 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
42	7887,0	7892,0	0,06	0,76
43	5114,0	5192,0	1,53	0,29
44	36022,0	36540,0	1,44	3,34
45	17676,0	17676,0	0,00	0,37
46	48701,0	49318,0	1,27	1,32
47	66230,0	66259,0	0,04	1,94
48	58964,0	59052,0	0,15	0,81
49	79614,0	80329,0	0,90	2,67

Tabela 10 - Resultados do PD-RAMP para o subconjunto 6 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
50	5937,0	5950,0	0,22	0,83
51	9060,0	9123,0	0,70	0,54
52	34652,0	34729,0	0,22	2,14
53	30038,0	30049,0	0,04	2,56
54	43853,0	43878,0	0,06	1,87
55	69610,0	70996,0	1,99	1,37
56	64474,0	64482,0	0,01	0,58
57	49791,0	49792,0	0,00	0,34

Tabela 11 - Resultados do PD-RAMP para o subconjunto 7 de Delmaire *et al.*

Problema	Z^*	Z	Gap	T(s)
cap61	932615,75	932615,75	0,00	0,05
cap62	977799,40	977799,40	0,00	0,06
cap63	1014099,61	1014099,61	0,00	0,05
cap64	1053197,44	1055801,24	0,25	0,05
cap71	932615,75	932615,75	0,00	0,04
cap72	977799,40	977799,40	0,00	0,04
cap73	1010641,45	1010641,45	0,00	0,02
cap74	1034976,97	1034976,98	0,00	0,02

Tabela 12 - Resultados do PD-RAMP para o subconjunto 1 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap91	796648,44	796648,44	0,00	0,10
cap92	858109,32	858109,33	0,00	0,10
cap93	900760,11	900760,11	0,00	0,18
cap94	950608,42	950608,43	0,00	0,16
cap101	796648,44	796648,44	0,00	0,07
cap102	854704,20	854704,20	0,00	0,06
cap103	893782,11	893782,11	0,00	0,03
cap104	928941,75	928941,75	0,00	0,03

Tabela 13 - Resultados do PD-RAMP para o subconjunto 2 de Beasley

Problema	Z^*	Z	Gap	T(s)
cap121	793439,56	793439,56	0,00	0,16
cap122	854900,45	854900,45	0,00	0,20
cap123	898266,08	898266,08	0,00	0,29
cap124	950608,43	950644,44	0,00	0,22
cap131	793439,56	793439,56	0,00	0,15
cap132	851495,32	851495,33	0,00	0,11
cap133	893076,71	893076,71	0,00	0,19
cap134	928941,75	928941,75	0,00	0,06

Tabela 14 - Resultados do PD-RAMP para o subconjunto 3 de Beasley

Problema	Z^*	Z	Gap	T(s)
capa1	19240822,45	19246455,34	0,03	56,41
capa2	18438046,45	18438890,63	0,01	59,14
capa3	17765201,95	17765201,95	0,00	26,25
capa4	17160439,01	17160704,09	0,00	10,30
capb1	13656379,58	13661697,60	0,04	42,91
capb2	13361927,45	13376639,29	0,11	64,28
capb3	13198556,43	13227539,95	0,22	44,33
capb4	13082516,50	13145940,59	0,49	65,08
capc1	11646596,97	11647500,28	0,01	23,32
capc2	11570340,29	11570690,98	0,00	22,67
capc3	11518743,74	11524374,53	0,05	22,22
capc4	11505767,39	11505861,86	0,00	17,30

Tabela 15 - Resultados do PD-RAMP para o subconjunto 4 de Beasley