Ministério da
Ciência e Tecnologia

UMA META-HEURÍSTICA HÍBRIDA COM BUSCA POR AGRUPAMENTOS APLICADA A PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

Antonio Augusto Chaves

Tese de Doutorado em Computação Aplicada, orientada pelo Prof. Dr. Luiz Antonio Nogueira Lorena, aprovada em 10 de março de 2009.

Registro do documento original:
<http://urlib.net/>

INPE
São José dos Campos
2009

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

Ministério da
Ciência e Tecnologia

UMA META-HEURÍSTICA HÍBRIDA COM BUSCA POR AGRUPAMENTOS APLICADA A PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

Antonio Augusto Chaves

Tese de Doutorado em Computação Aplicada, orientada pelo Prof. Dr. Luiz Antonio Nogueira Lorena, aprovada em 10 de março de 2009.

Registro do documento original:
<http://urlib.net/>

INPE
São José dos Campos
2009

Dados Internacionais de Catalogação na Publicação (CIP)

Chaves, Antonio Augusto.
Uma meta-heurística híbrida com busca por agrupamentos aplicada a problemas de otimização combinatória/ Antonio Augusto Chaves. – São José dos Campos: INPE, 2009.
197p. ; ()

Tese () – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2009.

1. Meta-heurísticas. 2. Otimização combinatória. 3. Busca por agrupamentos. 4. Localização de facilidades. 5. Caixeiro viajante. 6. Linha de produção. 1. Metaheuristics. 2. Combinatorial optimization. 3. Clustering search. 4. location of facilities . 5. Traveling salesman. 6. Assembly line.

CDU

ATENÇÃO! A FOLHA DE APROVAÇÃO SERÁ INCLUIDA POSTERIORMENTE.

"A alegria está na luta, na tentativa, no sofrimento envolvido. Não na vitória propriamente dita.".

Mahatma Gandhi

A meus pais ...

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por ter me guiado por este caminho.

Ao Prof. Luiz Antonio Nogueira Lorena, pela orientação, conhecimento transmitido, apoio e pelas várias oportunidades que me foram dadas para crescer na vida acadêmica.

À minha família por sempre acreditar em mim e pelo apoio incondicional. À minha namorada Thaís pelo carinho e compreensão nestes últimos meses.

Ao Prof. Cristobal Mirales pelo apoio e orientação, e por me mostrar que a Pesquisa Operacional pode ser aplicada com fins sociais.

Ao Prof. Marcone Jamilson Freitas Souza, meu orientador na graduação, por ter despertado em mim o gosto pela pesquisa e por todo o incentivo para que continuasse na vida acadêmica.

Aos membros da banca pela disposição e análise deste documento e pelas contribuições propostas.

Ao Instituto Nacional de Pesquisas Espaciais (INPE) pela oportunidade, e aos professores da CAP pelos ensinamentos transmitidos.

A todos os meus amigos de Piumhí, Ouro Preto e São José dos Campos, sem os quais eu nada seria. À república K-Zona por ser responsável pela pessoa que sou hoje.

Aos amigos do INPE que muito me ajudaram neste trabalho, com trocas de idéias e também com conversa fiada na hora do cafezinho.

A todas as secretárias que passaram pela CAP, e, principalmente, as atuais secretárias pelo apoio.

E finalmente, à CAPES e ao CNPQ pelo suporte recebido.

RESUMO

Esta tese apresenta um método híbrido, denominado Busca por Agrupamentos (CS, do inglês *Clustering Search*), que consiste em detectar dinamicamente regiões promissoras no espaço de busca baseando-se na frequência em que são amostradas nestas regiões as soluções geradas por uma meta-heurística. Um processo de agrupamento iterativo é executado em conjunto com a meta-heurística, agrupando as soluções similares e mantendo soluções que sejam representativas para os grupos de soluções. As regiões promissoras podem ser exploradas tão logo sejam descobertas, por meio de heurísticas de busca local específicas para o problema abordado. São propostas algumas aplicações do CS a diferentes problemas de Otimização Combinatória encontrados na literatura, tais como, Problema de p-Medianas Capacitado, Problema de Agrupamento Centrado Capacitado, Problema do Caixeiro Viajante com Coleta de Prêmios e o Problema de Balanceamento e Designação de Trabalhadores em Linhas de Produção. Esses problemas possuem diferentes características e particularidades, sendo assim, é possível analisar o comportamento do CS em diversas situações. Nessas abordagens são utilizadas diferentes meta-heurísticas para gerar soluções para o processo de agrupamento do CS, e também um método gerador de soluções aleatórias. Os testes computacionais mostram o potencial do CS para resolução desses problemas de otimização, colocando-o como uma alternativa para problemas que necessitem ser resolvidos de forma aproximada e em um tempo computacional competitivo. Conclusões a respeito dos componentes e parâmetros do CS também são apresentadas.

A HYDRID METAHEURISTIC WITH CLUSTERING SEARCH APPLIED TO COMBINATORIAL OPTIMIZATION PROBLEMS

ABSTRACT

This thesis presents a hybrid method, denominated Clustering Search (CS), that consists of detecting dynamically promising regions in the search space based on the frequence that are sampled in these regions the solutions originated from the metaheuristic. A iterative clustering process is executed in ensembling the metaheuristic, grouping the similar solutions and keeping solutions that are representative to the clusters. The promising regions can be explored as soon as they are discovered, by means of local search heuristics. Some applications of CS are proposed in different combinatorial optimization problems found in literature like the Capacitated p-Median Problem, Capacitated Centred Clustering Problem, Prize Collecting Traveling Salesman Problem and the Assembly Line Worker Assignment and Balancing Problem. These problems have different characteristics and particularities, therefore, it is possible to analyse the behavior of CS in several situations. In these approaches different metaheuristics are utilized to generate solutions for the clustering process of CS, and also a generator method of random solutions. The computational tests present the potential of CS for resolving these optimization problems, putting it as an alternative for the problems that demand to be solved in an approximate form and in a competitive computational time. Conclusions regarding the components and parameters of CS are also presented.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

CS	–	*Clustering Search*
ECS	–	*Evolutionary Clustering Search*
ACO	–	*Ant Colony Optimization*
GA	–	*Genetic Algorithm*
SA	–	*Simulated Annealing*
TS	–	*Tabu Search*
GRASP	–	*Greedy Randomized Adaptive Search Procedure*
VNS	–	*Variable Neighborhood Search*
VND	–	*Variable Neighborhood Descent*
ILS	–	*Iterated Local Search*
PR	–	*Path-Relinking*
CPMP	–	*Capacitated p-Median Problem*
CCCP	–	*Capacitated Centered Clustering Problem*
TSP	–	*Traveling Salesman Problem*
TSPP	–	*Traveling Salesman Problem with Profits*
PCTSP	–	*Prize Collecting Traveling Salesman Problem*
PTP	–	*Profitable Tour Problem*
QTSP	–	*Quota Traveling Salesman Problem*
ALWABP	–	*Assembly Line Worker Assigment and Balancing Problem*
SALB	–	*Simple Assembly Line Balancing Problem*
ALDP	–	*Assembly Line Design Problem*
CTD	–	Centro de Trabalho para Deficientes
OMS	–	Organização Mundial de Saúde
ONG	–	Organização não Governamental
BOX	–	*Block Order Crossover*
LCR	–	Lista de candidatos restrita

LISTA DE SÍMBOLOS

P	–	problema de otimização
S	–	espaço de busca
X	–	variáveis de decisão
Ω	–	conjunto de restrições entre variáveis
f	–	função objetivo
s	–	solução do espaço de busca
s^*	–	solução ótima global
s'	–	solução vizinha
$\hat{s}$	–	solução ótima local
C	–	cluster do CS
c	–	centro do cluster
v	–	volume do cluster
r	–	índice de ineficácia da busca local aplicada no cluster
r_{max}	–	valor máximo do índice de ineficácia
λ	–	limitante acima do qual um cluster se torna promissor
d	–	distância entre dois clusters
μ	–	tamanho da população do GA
p_c	–	probabilidade de cruzamento do GA
p_m	–	probabilidade de mutação do GA
T_0	–	temperatura inicial do SA
SA_{max}	–	número de iterações em cada temperatura do SA
T	–	temperatura corrente do SA
α	–	taxa de resfriamento do SA
N^k	–	k-ésima estrutura de vizinhança
N	–	conjunto de soluções aleatórias geradas no método para criar clusters
M	–	conjunto de soluções diversas que constituem os centros dos clusters iniciais
D	–	matriz de diversidade entre as soluções de N
D_{soma}	–	soma das diversidades entre um candidato e todas as soluções de M
s_{inicio}	–	solução inicial do PR
s_{guia}	–	solução final do PR
mv	–	movimento do PR
mv^*	–	melhor movimento em uma iteração do PR
T_j	–	soma das distâncias entre a mediana j e os pontos de demanda alocados a ela
ϖ	–	peso que reflete a penalidade imposta se a restrição de capacidade de uma mediana for violada
φ	–	distância na qual um vizinho é gerado no VNS

β	–	porcentagem da perturbação do ILS
D_j	–	demanda total atendida pela mediana j
Q_j	–	capacidade da mediana j
t_{ij}	–	distância entre a mediana j e o ponto de demanda i
c_{ij}	–	custo de deslocamento entre o vértice i e j
γ_i	–	penalidade paga se o vértice i não for visitado
ψ	–	peso que reflete a penalidade imposta se a restrição de prêmio mínimo for violada
p_i	–	prêmio coletado no vértice i
p_{min}	–	prêmio mínimo
ω	–	peso que reflete a penalidade da rede de precedência
δ	–	peso que reflete a penalidade de incompatilidade entre tarefa e trabalhador
C_{tempo}	–	maior tempo de processamento entre as estações de trabalho

1 INTRODUÇÃO

Muitas pesquisas vêm sendo realizadas durante as últimas décadas buscando métodos que resolvam o desafio de encontrar soluções ótimas para problemas de otimização de importância prática e/ou teórica. Esses problemas surgem numa infinidade de aplicações da vida real, tais como, projetos de redes de telecomunicação e de circuitos integrados, empacotamento de objetos, localização de centros ou facilidades, roteamento e escalonamento de veículos, alocação de trabalhadores ou máquinas a tarefas, planejamento de produção, sequenciamento de DNA e proteínas, etc.

Segundo Blum e Roli (2008), um problema de otimização P pode ser descrito como uma tripla (S, Ω, f), na qual:

a) S é o espaço de busca definido sobre um conjunto finito de variáveis de decisão $X_i, i = 1, \ldots, n$. Caso essas variáveis tenham domínios discretos, P é chamado problema de otimização discreta (ou problema de otimização combinatória), e em casos de domínios contínuos trata-se de um problema de otimização contínua. Problemas com variáveis mistas também existem. Ω é um conjunto de restrições entre as variáveis;

b) $f : S \rightarrow \mathbb{R}^+$ é a função objetivo que especifica um valor positivo para cada elemento (ou solução) de S.

O objetivo de um problema de otimização é encontrar a melhor solução $s^* \in S$ tal que $f(s^*) \leq f(s), \forall s \in S$ (para um problema de minimização), ou $s^* \in S$ tal que $f(s^*) \geq f(s), \forall s \in S$ (para um problema de maximização). Em problemas reais, o objetivo é frequentemente otimizar várias funções objetivos ao mesmo tempo. Essa forma de otimização é definida como otimização multiobjetivo.

A Figura 1.1 ilustra um problema de maximização com uma função objetivo f definida sobre um espaço bidimensional $S = (S_1, S_2)$. Neste gráfico distingue-se entre ótimo local e global. Um ótimo local $\hat{s} \in S$ de uma função f é um elemento com $f(\hat{s}) \geq f(s')$ para todos s' vizinhos a $\hat{s}$, ou seja, não é possível por meio de um movimento discreto encontrar uma solução vizinha melhor que a solução corrente. Um ótimo global corresponde ao maior valor da função objetivo, entre todos os ótimos locais existentes no espaço de busca. A função objetivo dos problemas de

otimização são frequentemente multimodais, ou seja, existem múltiplos ótimos locais e globais. A existência de múltiplos ótimos locais dificulta consideravelmente a busca pelo ótimo global.

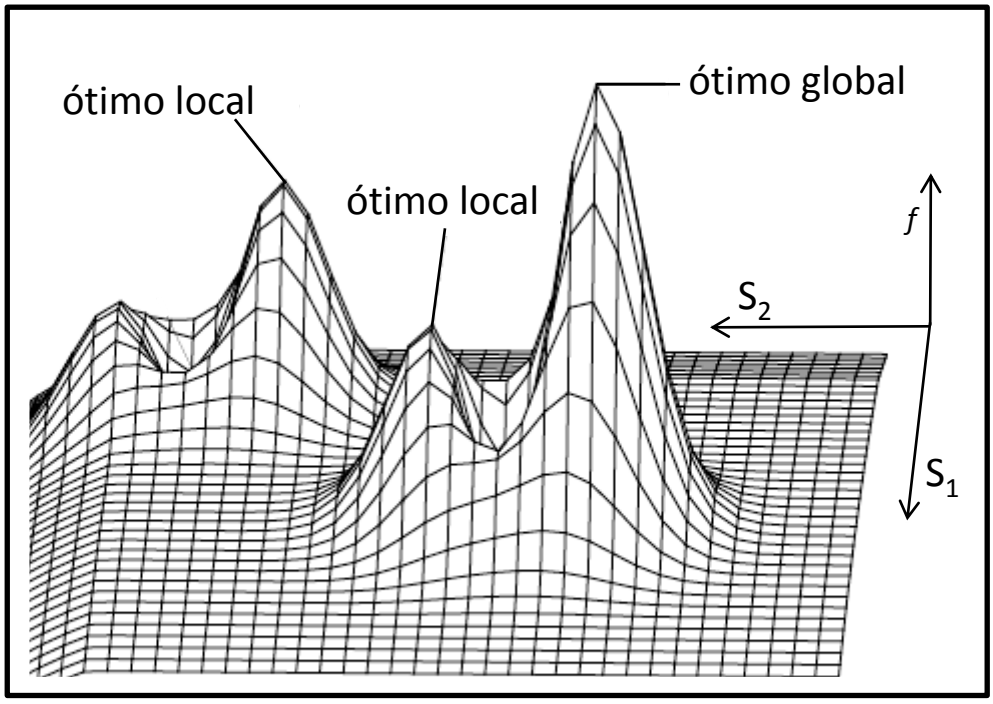

Figura 1.1 - Ótimo local e global de uma função bidimensional.

Fonte: Adaptado de Weise (2008).

Uma forma de resolver problemas de otimização combinatória seria simplesmente enumerar todas as soluções possíveis e guardar aquela de melhor valor da função objetivo. Porém, este método torna-se impraticável para problemas reais, pois, para a maioria dos problemas, o número de soluções possíveis cresce exponencialmente em função do tamanho do problema. Portanto, técnicas mais apuradas são necessárias.

Um exemplo da explosão combinatória em problemas reais pode ser visto no Problema do Caixeiro Viajante (TSP, do inglês *Traveling Salesman Problem*) (DANTZIG et al., 1954), um dos problemas mais estudados na literatura. No TSP deseja-se encontrar a melhor rota entre n vértices de um grafo completo, visitando cada vértice apenas uma vez. Considerando que as distâncias entre os vértices são simétricas, o número de rotas possíveis é $(n-1)!/2$. Para um exemplo com 20 vértices,

tem-se 6×10^{16} rotas possíveis. Desta forma, um computador que avalie uma rota em cerca de 10^{-8} segundos, gastaria aproximadamente 19 anos para encontrar a melhor rota. Mesmo considerando os rápidos avanços tecnológicos dos computadores, uma enumeração completa de todas essas rotas é inconcebível para valores elevados de n.

Os métodos de otimização procuram acrescentar um pouco de "inteligência" ao processo de enumeração, reduzindo assim o número de soluções a analisar no espaço de busca e possibilitando a resolução de problemas de dimensões mais elevadas.

Esses métodos podem ser classificados como exatos e heurísticos. Nos métodos exatos é garantido encontrar a solução ótima para um problema de otimização combinatória (NEMHAUSER; WOLSEY, 1988). No entanto, para problemas de otimização combinatória que são classificados como NP-*hard* não existe um algoritmo que consiga provar que uma solução é ótima em tempo polinomial (GAREY; JOHNSON, 1990). Por isso, métodos exatos geralmente necessitam de um tempo computacional exponencial. Isso frequentemente conduz a tempos computacionais inviáveis para o propósito prático. Nos métodos heurísticos sacrifica-se a garantia de encontrar soluções ótimas para obter "boas" soluções em um tempo computacional viável (BLUM; ROLI, 2008).

Na prática, uma solução relativamente boa pode ser considerada suficiente para muitas aplicações. Com base nesse cenário, as heurísticas surgem como ferramentas eficientes para solucionar problemas de otimização, retornando uma solução de qualidade em um tempo adequado para as necessidades da aplicação.

Em otimização, heurísticas são métodos que utilizam o conceito de *tentativa* e *erro* para encontrar soluções de boa qualidade em um tempo computacional razoável para problemas de otimização complexos, sem, no entanto, estar capacitada a garantir a otimalidade, bem como garantir quão próxima uma determinada solução está da solução ótima. As heurísticas se baseiam no conhecimento empírico sobre um determinado problema para examinar as soluções e encontrar uma boa solução.

A grande desvantagem das heurísticas reside na dificuldade de escapar de ótimos locais, o que deu origem à outra metodologia, chamada meta-heurística, que são métodos que utilizam estratégias de alto nível para explorar o espaço de busca utilizando diferentes heurísticas (KELLY, 1996). As meta-heurísticas possuem ferramentas que possibilitam escapar dos ótimos locais, permitindo a busca em

regiões mais promissoras. O grande desafio é produzir, em tempo reduzido, soluções tão próximas quanto possíveis da solução ótima.

Como alternativa para melhorar o desempenho das meta-heurísticas tem surgido nos últimos anos o interesse por meta-heurísticas híbridas. Vários tipos de integrações são possíveis, dando origem a algoritmos eficazes e eficientes. Uma combinação muito utilizada é a aplicação de heurísticas de busca local específicas para o problema abordado para intensificar a busca nas soluções das meta-heurísticas.

Entretanto, aplicar heurísticas de busca local em cada solução gerada por uma meta-heurística pode tornar o processo de busca impraticável em relação ao tempo de processamento. Sendo assim, o conceito de regiões promissoras pode ser empregado, permitindo uma aplicação mais racional dos métodos de intensificação de busca e melhorando o desempenho dos algoritmos. Principalmente, quando se tratar de heurísticas de busca local com altos custos computacionais.

A ideia de regiões promissoras pode ser associada ao emprego de estratégias capazes de identificar o potencial positivo de subespaços de busca. Soluções são identificadas segundo alguma estratégia, representando tais regiões promissoras. Estas soluções devem ser examinadas, assim que possível, por componentes heurísticos específicos (OLIVEIRA, 2004).

Neste contexto, a meta-heurística seria responsável pela exploração do espaço de busca, gerando soluções representativas de regiões promissoras. Em conjunto com a meta-heurística, métodos de busca local específicos ficam responsáveis por realizar uma intensificação da busca em tais regiões.

1.1 Objetivos e Contribuições

Esta tese se propõe a contribuir com o desenvolvimento de um método híbrido, combinando meta-heurísticas e heurísticas de busca local, para resolver problemas de otimização combinatória. A busca é intensificada somente em regiões consideradas promissoras, que são detectadas por meio de um processo de agrupamento de soluções.

Esse método foi proposto inicialmente em Oliveira (2004) e Oliveira e Lorena (2004), sendo chamado Busca Evolutiva por Agrupamentos (ECS, do inglês *Evolutionary Clustering Search*). Neste trabalho faz-se uma generalização do ECS, chamada

simplesmente Busca por Agrupamentos (CS, do inglês *Clustering Search*).

Os principais objetivos deste trabalho são generalizar o método em relação à meta-heurística utilizada e tornar simples as ideias acerca do CS, além de inserir novos atributos neste. Com isso, espera-se reduzir o esforço de implementações futuras e tornar o CS um método interessante para resolver diferentes problemas de otimização combinatória. Deseja-se também mostrar que o CS é um método robusto, competitivo e eficiente tanto em termos de qualidade de soluções quanto em tempo computacional para obtê-las.

Para tanto, o CS foi aplicado a alguns problemas de otimização combinatória encontrados em diferentes contextos, tais como, localização de facilidades, roteamento de veículos e balanceamento de linhas de produção. Sendo assim, são apresentadas três abordagens do CS.

A primeira das abordagens consiste na aplicação do CS ao Problema de p-Medianas Capacitado (CPMP, do inglês *Capacitated p-Median Problem*) (OSMAN; CHRISTOFIDES, 1994) e ao Problema de Agrupamento Centrado Capacitado (CCCP, do inglês *Capacitated Centered Clustering Problem*) (NEGREIROS; PALHANO, 2006).

A segunda abordagem consiste na aplicação do CS ao Problema do Caixeiro Viajante com Coleta de Prêmios (PCTSP, do inglês *Prize Collecting Traveling Salesman Problem*) (BALAS, 1989), que é uma generalização do TSP pertencente à classe de Problemas do Caixeiro Viajante com Lucros (*TSPP, do inglês Traveling Salesman Problems with Profits*) (FEILLET et al., 2005).

Na terceira abordagem, o CS é aplicado ao Problema de Balanceamento e Designação de Trabalhadores em Linha de Produção (ALWABP, do inglês *Assembly Line Worker Assignment and Balancing Problem*) (MIRALLES et al., 2008).

Neste trabalho, utilizam-se diferentes meta-heurísticas clássicas como geradores de soluções para o processo de agrupamento do CS, visando mostrar a flexibilidade deste método com relação à meta-heurística utilizada. Um gerador de soluções aleatórias também foi testado. Além dessa análise, um número grande de testes foi realizado para investigar a influência de cada atributo do CS em seu comportamento.

1.2 Organização da Tese

Esta tese está organizada em oito capítulos, sendo este o primeiro. No Capítulo 2 são apresentados os conceitos básicos sobre meta-heurísticas, e de forma resumida, as principais características das meta-heurísticas utilizadas neste trabalho. Uma revisão bibliográfica sobre métodos de otimização com agrupamentos de soluções também é realizada. Em seguida, no Capítulo 3 faz-se uma descrição detalhada do método CS, apresentando seus conceitos e componentes.

Os Capítulos 4, 5 e 6 apresentam as abordagens do CS aplicado ao CPMP e ao CCCP, ao PCTSP e ao ALWABP, respectivamente. Nesses capítulos também são apresentadas as definições, formulações matemáticas e revisões bibliográficas sobre os problemas abordados, além dos resultados computacionais destes.

No Capítulo 7 são apresentadas análises sobre os experimentos computacionais realizados variando os atributos do CS, e conclusões sobre a influência de cada atributo no desempenho do CS.

Por último, no Capítulo 8 encontra-se um sumário das principais contribuições e considerações a respeito deste trabalho, seguido de sugestões para futuras pesquisas.

2 META-HEURÍSTICAS E BUSCA LOCAL EM REGIÕES PROMISSORAS

2.1 Conceitos Sobre Meta-heurísticas

O trabalho de Lin e Kernighan (1973) deu origem aos estudos sobre as heurísticas modernas, as quais evoluíram e atualmente são chamadas meta-heurísticas. O termo meta-heurística, introduzido por Glover (1986), deriva da composição de duas palavras gregas, heurística (do verbo *heuristiké*) que significa "arte de encontrar, descobrir", e o prefixo meta (*metá*) que exprime a ideia de "nível superior, maior generalidade".

Na literatura existem diversas definições para meta-heurística, porém, não há um consenso entre os autores. Blum e Roli (2003) apresentam algumas destas definições e fazem um resumo das propriedades fundamentais que caracterizam as meta-heurísticas, sendo:

- meta-heurísticas são estratégias que "guiam" o processo de busca;
- o objetivo é explorar eficientemente o espaço de busca para encontrar soluções ótimas ou próximas do ótimo;
- técnicas que fazem parte das meta-heurísticas variam desde procedimentos de busca local simples até processos de aprendizado complexos;
- meta-heurísticas incorporaram mecanismos para evitar ficarem presas em ótimos locais do espaço de busca;
- meta-heurísticas não são específicas para um determinado problema;
- meta-heurísticas podem fazer uso de conhecimento específico do problema por meio de heurísticas que são controladas por estratégias superiores; e
- meta-heurísticas mais avançadas utilizam a experiência obtida durante a busca (por meio de uma memória) para guiar a busca.

Dentre os algoritmos classificados como meta-heurísticas podemos citar: Otimização por Colonia de Formigas (ACO, do inglês *Ant Colony Optimization*), Algoritmo Genético (GA, do inglês *Genetic Algorithm*), Busca Tabu (TS, do inglês *Tabu Search*), Recozimento Simulado (SA, do inglês *Simulated Annealing*), Pesquisa em Vizinhança Variável (VNS, do inglês *Variable Neighboorhod Search*), Busca Local Iterativa (ILS, do inglês *Iterated Local Search*), entre outros.

As meta-heurísticas, em geral, encontram várias dificuldades durante o processo de busca. Um dos problemas é que não é possível determinar se a melhor solução conhecida é um ótimo local ou global, e consequentemente, se a convergência é aceitável. Sendo assim, não é possível saber se o processo de otimização pode ser interrompido, se deveria tentar melhorar a solução corrente, ou se deveria examinar outras partes do espaço de busca. Isto se torna problemático se levarmos em conta que a maioria dos problemas de otimização são multimodais, e a ocorrência de vários ótimos locais dificulta o processo de busca.

Um dos grandes desafios enfrentado pelas meta-heurísticas é evitar a convergência prematura para um ótimo local, mas também não impedir que o algoritmo convirja. Diz-se que um algoritmo convergiu prematuramente para um ótimo local se este não é capaz de explorar outras partes do espaço de busca, além da área examinada até o momento, e existem outras regiões com soluções melhores que a solução corrente.

As operações para criar novas soluções a partir das soluções existentes têm um grande impacto na velocidade de convergência (SMITH, 2004). Estabelecer estas operações apropriadamente é muito importante e conduz a um problema conhecido na literatura como diversificação *versus* intensificação (*exploration* $\times$ *exploitation*) (GLOVER; LAGUNA, 1997; EIBEN; SCHIPPERS, 1998; TAN et al., 2008).

Diversificação significa encontrar novas regiões do espaço de busca que ainda não tenham sido investigadas, enquanto, intensificação significa tentar melhorar a solução corrente realizando pequenas mudanças que conduzam a novas soluções próximas à solução corrente em uma região.

Algoritmos que focam intensificação têm uma velocidade de convergência muito rápida, mas correm o risco de não encontrar a solução ótima e talvez ficarem presos em um ótimo local. Já algoritmos que realizam uma diversificação excessiva podem encontrar a solução ótima, mas levarão um tempo computacional muito alto. O

ideal para um algoritmo de otimização seria empregar operadores com características explorativas e operadores capazes de pesquisar mais detalhadamente a região de uma determinada solução.

Todas meta-heurísticas procuram balancear diversificação e intensificação. Isto é importante, de um lado para identificar rapidamente regiões no espaço de busca com soluções de alta qualidade, e de outro lado para não perder muito tempo em regiões do espaço de busca que já foram exploradas ou que não forneçam boas soluções.

Nos últimos anos, tornou-se evidente que a concentração em uma única meta-heurística é muito restritiva. Na prática, a combinação de uma meta-heurística com outras técnicas de otimização, chamada meta-heurística híbrida, pode proporcionar um comportamento mais eficiente e uma maior flexibilidade quando se trata de problemas do mundo real e em larga escala (BLUM; ROLI, 2008). Isto pode ser obtido, por exemplo, combinando a meta-heurística com métodos exatos (*branch* e *bound*, programação linear, geração de colunas, etc), com outras meta-heurísticas, ou com heurísticas de busca específicas para um problema. A motivação por trás dessas hibridizações de diferentes métodos é geralmente obter um melhor desempenho e explorar sistemas que unam as vantagens das diferentes estratégias.

De fato, a escolha de uma abordagem híbrida adequada, muitas vezes utilizando algoritmos de busca mais especializados que consideram a natureza do problema e/ou informações sobre o espaço de busca, é determinante para alcançar o melhor desempenho na resolução de problemas difíceis (RAIDL, 2006).

2.2 Meta-heurísticas Clássicas

Dentro da abordagem proposta nesta tese foram implementadas algumas meta-heurísticas clássicas, tais como, Algoritmo Genético, Recozimento Simulado, Pesquisa em Vizinhança Variável e Busca Local Iterativa. A seguir realiza-se uma revisão acerca desses métodos, apresentando suas principais características.

2.2.1 Algoritmo Genético

O Algoritmo Genético (GA) foi introduzido por Holland (1975). O GA faz parte de um escopo mais abrangente, chamado Algoritmos Evolutivos (*Evolutionary Algorithms*). Nas décadas de 60 e 70, surgiram vários pesquisadores de diferentes pontos dos Estados Unidos e Europa cujas pesquisas convergiam para a ideia de

mimetizar o mecanismo de evolução biológica para resolver problemas das mais diversas áreas, em especial a de otimização. Dessas pesquisas resultaram diferentes abordagens algorítmicas, cujos principais representantes são as estratégias evolutivas (RECHENBERG, 1973), a programação evolutiva (FOGEL et al., 1966) e os algoritmos genéticos (HOLLAND, 1975; JONG, 1975; GOLDBERG, 1989).

O GA é uma meta-heurística que se fundamenta em uma analogia com o processo de evolução natural descrito por Charles Darwin. Assim como acontece no meio ambiente, em um GA existe uma população de indivíduos que competem entre si para garantir a própria sobrevivência e para assegurar que suas características sejam passadas adiante. Os melhores indivíduos, ou seja, os mais bem adaptados ao ambiente, contam com uma maior probabilidade de "sobreviver" e, por meio de operadores genéticos, produzirem filhos cada vez mais aptos. Cada iteração do processo é chamada de geração. Após várias gerações, existe uma grande probabilidade de que a população tenha aumentado a sua qualidade média, isto é, seja composta por indivíduos mais bem adaptados ao ambiente (REEVES, 2003).

Portanto, cada solução de um problema de otimização é representada por um indivíduo, sendo definido por um cromossomo composto por genes que estão associados aos atributos da solução.

A Figura 2.1 apresenta o pseudocódigo de um GA clássico.

```
algoritmo GA ( μ, p_c, p_m )
    gere a população inicial de tamanho μ
    enquanto ( critério de parada não for satisfeito ) faça
        selecione a população de pais
        aplique o operador de recombinação com probabilidade p_c
        aplique o operador de mutação com probabilidade p_m
        avalie a população de filhos
        selecione a nova população
    fim-enquanto
fim-algoritmo
```

Figura 2.1 - Pseudocódigo da meta-heurística GA.

Fonte: Adaptado de Reeves (2003).

A população inicial pode ser gerada aleatoriamente, objetivando representar o maior número possível de regiões do espaço de busca do problema. A definição do tamanho da população do GA (μ) é uma questão muito importante. Uma população pequena pode não ser suficiente para explorar o espaço de busca de maneira efetiva, enquanto, uma população muito grande pode acarretar em um tempo de processamento inviável.

Existem várias maneiras de selecionar os indivíduos para o processo de reprodução, tais como, roda da roleta, *ranking*, torneio, entre outros. O processo de seleção deve privilegiar os melhores indivíduos. A seleção por torneio é a maneira mais comum de se escolher a população de pais. Nesta estratégia seleciona-se aleatoriamente k indivíduos da população, e o melhor destes indivíduos é escolhido para fazer parte da população de pais. Este procedimento é repetido até que sejam selecionados μ indivíduos para formar a população de pais.

A recombinação (*crossover*) é considerada por muitos autores como o principal operador do GA. Na recombinação os genes de dois cromossomos pais são combinados gerando dois cromossomos filhos, de forma que, em cada cromossomo filho haja um conjunto de genes de cada um dos cromossomos pais. Existem vários tipos de recombinação, e a escolha de qual utilizar está diretamente ligada às características do problema abordado.

Após o operador de recombinação ter gerado μ filhos é aplicado o operador de mutação, que consiste em alterar aleatoriamente uma parte dos genes de cada cromossomo, permitindo inserir na população características que ainda não existam no sistema.

Os operadores de recombinação e mutação são realizados com certa probabilidade. A recombinação é realizada geralmente com alta probabilidade ($p_c \in [0,5; 1,0]$) e a mutação com probabilidade mais baixa ($p_m \in [0,001; 0,05]$). Caso a probabilidade de recombinação não seja aceita, os filhos são criados iguais aos pais.

No GA proposto por Holland (1975) é utilizada uma abordagem geracional, na qual em cada geração começa-se com uma população de tamanho μ, da qual são selecionados μ pais para reprodução. Em seguida μ filhos são gerados aplicando os operadores de recombinação e mutação. Por fim, a população de filhos se torna a nova população corrente.

Entretanto, nessa estratégia existe o risco de perder informações referente à uma boa solução. Por essa razão Jong (1975) introduziu o conceito de elitismo, que sempre mantém na população o melhor indivíduo encontrado durante a busca. As estratégias utilizadas na seleção dos pais também podem ser utilizadas para selecionar a população sobrevivente. Uma questão importante é que se deve admitir a sobrevivência de indivíduos menos aptos, objetivando que a população não perca a característica de diversidade.

2.2.2 Recozimento Simulado

A meta-heurística Recozimento Simulado (SA) foi proposta por Kirkpatrick et al. (1983), sendo uma das primeiras meta-heurísticas relatadas na literatura. O SA procura simular o processo de recozimento físico (*physical annealing*), no qual um sólido é aquecido a uma alta temperatura (acima do seu ponto de fusão), e logo em seguida é realizado uma lenta e gradativa diminuição de sua temperatura, até que o ponto de solidificação seja atingido. Durante o recozimento, o material passa por vários estados possíveis. Se o resfriamento for suficientemente lento obtêm-se uma estrutura cristalina livre de imperfeições (estado de mínima energia).

Em analogia com um problema de otimização combinatória, o nível de energia é a função objetivo do problema, um estado do sistema é uma solução viável, a temperatura é um parâmetro de controle e o estado de mínima energia é a melhor solução para o problema.

A ideia fundamental do SA é permitir movimentos que resultem em soluções de pior qualidade que a solução corrente, objetivando escapar de ótimos locais. A probabilidade de realizar tais movimentos é diminuída durante a busca (HENDERSON et al., 2003).

O SA inicializa sua busca a partir de uma solução inicial qualquer e de uma temperatura inicial (T_0) definida a *priori*. A cada temperatura do SA é aplicado o algoritmo de Metropolis (METROPOLIS et al., 1953) até que seja atingido o equilíbrio térmico nesta temperatura, que é obtido após certo número de iterações (SA_{max}). A cada iteração, uma solução vizinha da solução corrente é gerada aleatoriamente. Se essa solução for melhor que a solução corrente, continua-se a busca a partir do vizinho gerado. Caso contrário, existe uma probabilidade de aceitar uma solução de piora, que está associada ao valor da temperatura corrente.

A Figura 2.2 apresenta o pseudocódigo do SA.

```
algoritmo SA (T0, SAmax, α )
    gere uma solução inicial s
    IterT ← 0
    T ← T0
    enquanto (critério de parada não for satisfeito ) faça
        enquanto ( IterT < SAmax ) faça
            IterT ← IterT + 1
            gere um vizinho s' aleatoriamente ( s' ∈ N(s) )
            calcule Δ = f (s') – f (s)
            se (Δ ≤ 0) então
                s ← s'
            senão
                s ← s' com probabilidade  e -Δ/T
        fim-enquanto
        T ← α x T
        IterT ← 0
    fim-enquanto
fim-algoritmo
```

Figura 2.2 - Pseudocódigo da meta-heurística SA.

Fonte: Adaptado de Henderson et al. (2003).

A probabilidade de aceitar uma solução que seja pior que a solução corrente é dada por $e^{-\Delta/T}$, sendo Δ a variação do valor da função objetivo ao mover-se da solução corrente para a solução vizinha, e T é a temperatura corrente. Para valores altos de temperatura há uma grande probabilidade de se aceitar uma solução de pior custo, e esta probabilidade é praticamente nula em temperaturas baixas.

A temperatura é gradativamente diminuída por uma taxa de resfriamento α após SA_{max} iterações, por exemplo, $T_k = \alpha \times T_{k-1}$ e $\alpha \in [0, 1]$. Sendo assim, o SA possui um comportamento aleatório no início do processo de busca e converge para um método de refinamento à medida que a temperatura se aproxima de zero, pois

a probabilidade de aceitar movimentos de piora diminui. Portanto, o critério de resfriamento é o responsável por determinar o balanceamento entre diversificação e intensificação no SA.

A fase intermediária do SA é a fase mais importante. Sendo assim, uma técnica muito utilizada é o reaquecimento da temperatura, permitindo que o processo de busca seja executado por mais algumas iterações antes que a temperatura se aproxime de zero (por exemplo, pode-se aumentar a temperatura a 30% da temperatura inicial, quando esta atingir o valor de 0,1 pela primeira vez).

O critério de parada do SA é quando a temperatura chega a um valor próximo de zero (por exemplo, $T = 0,001$) e nenhuma solução de piora é mais aceita, situação que evidencia o encontro de um ótimo local. Outro critério que pode ser utilizado é o número máximo de iterações.

2.2.3 Pesquisa em Vizinhança Variável

A meta-heurística Pesquisa em Vizinhança Variável (VNS) foi proposta por Mladenovic e Hansen (1997). O VNS combina busca local com mudanças sistemáticas de vizinhança, procurando escapar de ótimos locais. Ao contrário de outros métodos de busca local, o VNS não segue uma trajetória, mas explora vizinhanças cada vez mais distantes da solução corrente, gerando vizinhos aleatoriamente e movendo-se para a nova solução se e somente se uma melhora for produzida. Além disso, um método de busca local é aplicado repetidamente para transformar a solução vizinha em um ótimo local. Portanto, o VNS é um método que combina mudanças de vizinhança estocátisca e determinística.

O VNS parte de um conjunto pré-definido de estruturas de vizinhança $N^k, (k = \{1, \ldots, k_{max}\}$, sendo $N^k(s)$ o conjunto de soluções da $k-$ésima vizinhança da solução corrente s. O VNS inicializa com uma solução inicial qualquer e a cada iteração é gerado aleatoriamente um vizinho, s', dentro da vizinhança N^k da solução corrente ($s' \in N^k(s)$). Aplica-se então um método de busca local no vizinho gerado, encontrando uma solução que representa um ótimo local ($\hat{s}$). Se o ótimo local for melhor que a solução corrente ($f(\hat{s}) < f(s)$) a busca continua a partir deste, recomeçando da primeira estrutura de vizinhança. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança, N^{k+1} (HANSEN; MLADENOVIC, 2003a).

O critério de parada do VNS pode ser, por exemplo, o tempo máximo de processamento, o número máximo de iterações, ou o número máximo de iterações consecutivas sem melhora da solução corrente.

O pseudocódigo do VNS é apresentado na Figura 2.3.

```
algoritmo VNS
    gere uma solução inicial s
    seja k_max o número de estruturas diferentes de vizinhança
    enquanto ( critério de parada não for satisfeito ) faça
        k ← 1
        enquanto ( k ≤ k_max ) faça
            gere um vizinho s' aleatoriamente da k-ésima vizinhança de s (s' ∈ N^k(s))
            aplique uma busca local em s' obtendo um ótimo local ( ŝ )
            se ( f(ŝ) < f(s) ) então
                s ← ŝ
                k ← 1
            senão
                k ← k + 1
        fim-enquanto
    fim-enquanto
fim-algoritmo
```

Figura 2.3 - Pseudocódigo da meta-heurística VNS.

Fonte: Adaptado de Hansen e Mladenovic (2003b).

2.2.4 Busca Local Iterativa

A meta-heurística Busca Local Iterativa (ILS) foi proposta por Stützle (1999) e Lourenço et al. (2003). O ILS procura focar a busca não no espaço completo de soluções, mas em um subespaço S^* definido por soluções que são ótimos locais de determinado procedimento de otimização.

A ideia do ILS tem uma longa história (BAXTER, 1981), e seu estudo por muitos autores levou a vários métodos com nomes diferentes, como *iterated descent* (BAUM, 1986), *large-step Markov chains* (MARTIN et al., 1991), *iterated Lin-Kernighan* (JOHNSON, 1990) e *chained local optimization* (MARTIN; OTTO, 1993).

O ILS consiste em um processo iterativo, no qual uma solução é perturbada, gerando novas soluções de partida para um método de busca local. A perturbação tem uma importância muito grande. Uma perturbação pequena pode fazer com que o algoritmo não consiga escapar da região de atração de um ótimo local já pesquisado. Porém, com uma perturbação muito grande o algoritmo tem um reinício aleatório.

A Figura 2.4 apresenta o pseudocódigo do ILS.

```
algoritmo ILS
    gere uma solução inicial s_0
    ŝ ← BuscaLocal (s_0)
    enquanto ( critério de parada não for satisfeito ) faça
        s' ← Perturbação (ŝ, histórico)
        ŝ' ← BuscaLocal (s')
        ŝ ← CritériodeAceitação (ŝ, ŝ', histórico)
    fim-enquanto
fim-algoritmo
```

Figura 2.4 - Pseudocódigo da meta-heurística ILS.

Fonte: Adaptado de Lourenço et al. (2003).

O algoritmo ILS inicializa gerando uma solução inicial qualquer para o problema (s_0). Em seguida, é aplicada uma heurística de busca local específica obtendo uma solução ótima local ($\hat{s}$). A partir dessa solução inicializa-se um processo iterativo até que algum critério de parada seja satisfeito. A cada iteração realiza-se um procedimento de perturbação aleatória da solução $\hat{s}$ produzindo um novo ponto de partida para a busca local (s'). A heurística de busca local é então aplicada sobre s' encontrando a solução ótima local desta região ($\hat{s}'$). Um critério de aceitação é aplicado para decidir a partir de qual solução a busca continuará.

A perturbação precisa ser aleatória ou semi-determinística para evitar ciclagem. A principal característica da perturbação é a sua intensidade, isto é, a quantidade de mudanças realizadas na solução corrente. Essa intensidade pode ser fixa ou variável, porém, experimentalmente ficou evidenciado que uma intensidade variável é mais

eficiente (BLUM; ROLI, 2008). A perturbação deve ser forte o suficiente para permitir escapar do ótimo local corrente e explorar diferentes regiões. Ao mesmo tempo, precisa ser fraca o suficiente para guardar características do ótimo local corrente.

O método de busca local escolhido tem uma forte influência na qualidade da solução final e na velocidade de convergência do ILS. Normalmente utiliza-se uma heurística de busca local específica para o problema, mas também pode ser aplicado outra técnica de otimização.

O mecanismo de perturbação junto com a busca local definem as possíveis transições entre uma solução corrente $\hat{s}$ e uma solução "vizinha" $\hat{s}'$ ambas em S^*. O critério de aceitação irá determinar se $\hat{s}'$ é aceita ou não como a nova solução corrente. Uma forte intensificação é conseguida se somente soluções melhores são aceitas. Por outro lado, uma forma de favorecer a diversificação é sempre aceitar a nova solução. Para realizar um balanceamento entre intensificação e diversificação várias escolhas intermediárias podem ser utilizadas, como adotar uma probabilidade de aceitar um ótimo local que piore a solução corrente.

A maior complexidade do ILS está na dependência do *histórico*. Se não houver tal dependência, o ILS não possui memória: a perturbação e o critério de aceitação não dependem das soluções visitadas anteriormente no processo de busca, e o aceite ou não de $\hat{s}'$ é uma regra fixa. A maioria dos trabalhos utiliza o ILS dessa maneira, embora Stüzle (1998) mostre que o uso de memória melhora o desempenho do ILS (LOURENçO et al., 2003).

O método ILS se encerra após certo número de iterações sem melhora ou quando um tempo limite ou número de iterações for atingido.

2.3 Detecção de Regiões Promissoras e Algoritmos de Agrupamento

Muitos trabalhos recentes sobre meta-heurísticas híbridas referem-se à meta-heurísticas que empregam heurísticas de busca local, específicas para o problema tratado, objetivando acelerar o processo de busca dessas meta-heurísticas.

Heurísticas de busca local possuem um custo computacional muito alto, pois em geral, necessitam recalcular o valor da função objetivo a cada solução pesquisada. Sendo assim, a utilização indiscriminada destas heurísticas pode levar a um crescimento significativo da complexidade dos algoritmos.

Várias estratégias podem ser adotadas para limitar a aplicação de busca local. Um critério muito utilizado é o elitismo, que visa a aplicação de busca local apenas em um percentual das melhores soluções obtidas pelas meta-heurísticas. Outro critério é o probabilístico, no qual uma parte aleatória das soluções é melhorada por meio de busca local. Em ambos os casos podem ocorrer problemas relacionados à convergência. No elitismo, a busca pode ficar concentrada em poucas regiões, enquanto, regiões realmente promissoras podem nunca ser melhoradas no probabilístico.

Um critério interessante é a aplicação de busca local somente em regiões consideradas promissoras. Uma maneira de detectar essas regiões é por meio da quantidade de soluções geradas em uma região do espaço de busca. Considerando que as meta-heurísticas, em sua grande maioria, privilegiam as soluções de maior qualidade e estas influenciam no processo de busca, é esperado que se tenha uma maior amostragem de soluções próximas às regiões destas soluções. Sendo assim, é possível estabelecer que regiões nas quais muitas soluções sejam geradas tendem a ser regiões promissoras, isto é, regiões que contenham as melhores soluções.

Diversos benefícios podem ser obtidos intensificando a busca apenas em regiões supostamente promissoras, tais como, redução do número de buscas locais sem prejudicar o desempenho do algoritmo, provável pesquisa das principais regiões do espaço de busca, e aplicação de busca local em soluções com boa qualidade que necessitam de poucas alterações para torná-las ainda melhor e, consequentemente, um baixo custo computacional.

Técnicas de agrupamento podem ser utilizadas para dividir o espaço de busca em regiões. Um algoritmo de agrupamento é um método não supervisionado de classificação de padrões em grupos. Com agrupamento, um conjunto de dados é particionado em subconjuntos (grupos), tal que, os elementos em cada subconjunto compartilham características comuns. Proximidade ou medidas de distância são geralmente utilizadas como base para agrupar os elementos (JAIN et al., 1999).

No campo da otimização, considerando que muitos problemas possuem um conjunto de soluções muito grande, os algoritmos de agrupamento podem ser utilizados para reduzir esse conjunto sem perder sua característica de diversidade, agrupando soluções similares em grupos.

Na literatura existem alguns algoritmos de otimização que levam em consideração os conceitos de agrupamento de soluções e detecção de regiões promissoras, objetivando intensificar a busca sobre algumas regiões do espaço de busca.

A ideia de agrupamento, introduzida por Becker e Lago (1970), foi inspirada pela intenção de reduzir o esforço computacional desperdiçado ao se explorar a região de um mesmo ótimo local repetidamente. Idealmente, uma única busca local deveria ser realizada na região de atração[1] de um ótimo local, ou melhor, uma única busca local deveria ser inicializada a partir da região de atração dos poucos ótimos locais com valores de função objetivo muito bons. Técnicas de agrupamento podem ser utilizadas para limitar o número de buscas locais que conduzem à exploração de um ótimo local já pesquisado (SCHOEN, 2002).

O método de agrupamento (BECKER; LAGO, 1970; TÖRN, 1973) se baseia no fato de que soluções são geradas no espaço de busca de tal forma que há uma maior concentração de soluções dentro da região de atração de um ótimo local. Sendo assim, algum procedimento estatístico pode ser aplicado para particionar as soluções em grupos, que são caracterizados pelo fato das distâncias entre as soluções do mesmo grupo serem significativamente menores que distâncias entre as soluções pertencentes a grupos diferentes. Uma vez que os grupos tenham sido identificados, uma única solução representativa de cada grupo é escolhida e uma busca local é inicializada. Essa sequência de amostragem das soluções, agrupamento e busca local é repetida até algum critério de parada ser satisfeito. As soluções nas quais foram aplicadas busca local e seus respectivos ótimos locais são armazenadas separadamente. Portanto, a busca local só é executada em uma solução na qual esta ainda não tenha sido realizada.

Törn e Viitanen (1994) apresentam uma modificação do método de agrupamento, sendo o centro do grupo identificado diretamente sem nenhuma técnica de agrupamento. Para cada solução gerada na amostragem encontra-se um número pré-definido de vizinhos mais próximos, sendo esses vizinhos classificados como agrupados a essa solução. Após analisar todas as soluções é formado um grafo no qual os vértices são as soluções e as arestas representam os vizinhos de cada solução. A busca local é intensificada sobre aquelas soluções que não possuírem nenhum vizinho com um valor de função objetivo melhor, ou seja, nos vértices que são ótimos locais

[1] Região de atração de um ótimo local $\hat{s}$ é o conjunto de pontos em S a partir dos quais um procedimento de busca local converge para $\hat{s}$.

no grafo de soluções agrupadas.

Jelasity et al. (2001) apresentam uma técnica baseada no conceito de espécies para acelerar e paralelizar métodos de busca. Uma espécie pode ser definida como um conjunto de soluções que são similares de acordo com alguma medida de distância. A ideia é dividir a população em espécies que representem diferentes regiões de atração do espaço de busca. Cada espécie evolui em torno de uma solução representativa dessa espécie, independentemente das demais. Ao longo do processo de busca espécies são criadas, excluídas e otimizadas. Um método de otimização específico precisa ser implementado para intensificar a busca nas regiões delimitadas pelas espécies. O número de espécies é reduzido na medida em que a busca prossegue. Esse método pode ser paralelizado alocando cada espécie em um processador diferente.

Em Chelouah e Siarry (2003) uma nova estratégia foi proposta para combinar diversificação e intensificação. Na fase de diversificação é utilizado um algoritmo genético (GA), o qual parte de uma solução inicial e por meio dos operadores genéticos (cruzamento e mutação) tende a evoluir para as melhores regiões do espaço de busca. O GA é interrompido quando a diversidade da população estiver baixa, indicando que a maioria dos indivíduos está na mesma região. Essa região é considerada promissora e a fase de intensificação é executada dentro dessa região utilizando um método de busca local. A busca local parte do melhor indivíduo encontrado pelo GA. Para uma maior eficiência do método, é necessário definir cuidadosamente a população inicial e os operadores genéticos, levando em consideração o tamanho e a distribuição da população em todo o espaço de busca.

Oliveira e Lorena (2004) apresentam um algoritmo evolutivo híbrido que consiste num processo de agrupamento iterativo executado simultaneamente com um algoritmo evolutivo, detectando regiões promissoras no espaço de busca. Essas regiões são exploradas tão logo sejam descobertas, por meio de heurísticas de busca local. Uma região se torna promissora quando a quantidade de indivíduos gerados nesta região atingir certo limitante.

Martinez-Estudillo et al. (2005) também propuseram um algoritmo evolutivo híbrido combinando um processo de agrupamento e procedimentos de busca local. Nesse algoritmo é selecionado um subconjunto dos melhores indivíduos e executa-se uma análise dos indivíduos para agrupa-los em grupos similares. A busca é intensificada somente nos melhores indivíduos de cada grupo. Há duas versões diferentes

dependendo do estagio no qual é realizada a busca local e o processo de agrupamento. Na primeira versão uma intensificação é aplicada uma única vez sobre a população final do algoritmo evolutivo. A segunda versão analisa a população do algoritmo evolutivo a cada geração, agrupando os indivíduos em grupos e intensificando a busca nos melhores indivíduos.

Wei e Zhao (2005) apresentam uma nova arquitetura de algoritmos híbridos, a qual utiliza técnicas para gerar nichos e métodos de busca dentro de um algoritmo genético. Um ecossistema natural possui diferentes subsistemas (nichos), e tais nichos podem evoluir em paralelo, convergindo para diferentes pontos ótimos no espaço de busca. Nesta nova arquitetura, uma busca é realizada nos nichos para localizar regiões promissoras dentro do espaço de busca. Uma outra busca é utilizada para descobrir um ótimo dentro de uma região promissora. Este método diminui a probabilidade de uma convergência prematura e melhora a capacidade de intensificação dos algoritmos genéticos.

Melo et al. (2007) propuseram uma abordagem diferente, encontrando regiões promissoras antes de utilizar um método de otimização. Partindo de uma amostragem uniforme de soluções, iterativamente eliminam-se regiões do espaço de busca onde o algoritmo garante, com alta probabilidade, que não existe um ótimo global. Para determinar regiões não promissoras utiliza-se um método estatístico de Regressão Linear Múltipla aplicado sobre o conjunto de soluções. O objetivo é realizar um pré-processamento reduzindo o espaço de busca e indicando regiões com grande probabilidade de encontrar o ótimo global. Com isso, aumenta-se a eficiência de um método de busca aplicado a partir dessas regiões promissoras.

Nesta tese aproveitam-se as ideias dos algoritmos de agrupamento e do critério de detecção de regiões promissoras baseado em amostragem para desenvolver um método híbrido, no qual uma meta-heurística é responsável por explorar o espaço de busca e um processo de agrupamento de soluções é executado iterativamente descobrindo regiões promissoras, nas quais ocorre uma intensificação da busca por meio de heurísticas de busca local.

3 MÉTODO HÍBRIDO COM BUSCA POR AGRUPAMENTOS

O algoritmo de Busca Evolutiva por Agrupamentos (ECS, do inglês, *Evolutionary Clustering Search*) surgiu do trabalho de Oliveira (2004), que aplicou este método à problemas de minimização de funções numéricas sem restrições e a um problema de sequenciamento de padrões chamado Problema de Leiaute de Matriz-Porta (*Gate Matrix Layout Problem*).

O ECS se baseia no fato de que, ao longo do processo de busca de um algoritmo evolutivo, há uma maior concentração de indivíduos em regiões nas quais estão os indivíduos melhor avaliados. Sendo assim, um processo de agrupamento é executado iterativamente com um algoritmo evolutivo, contabilizando o número de indivíduos gerados em regiões do espaço de busca e identificando grupos de indivíduos com similaridades que mereçam especial interesse. A ideia é detectar regiões promissoras por mérito de frequência, baseado na quantidade de soluções geradas em cada região de busca. Uma intensificação da busca é realizada nas regiões promissoras tão logo sejam detectadas (OLIVEIRA, 2004).

Neste trabalho são propostas modificações para o ECS, objetivando principalmente generalizar o algoritmo responsável por gerar soluções para o processo de agrupamento e clarear as idéias acerca do método. Por causa desta generalização, o nome do método foi simplificado para Busca por Agrupamentos (CS, do inglês *Clustering Search*). Essa e outras modificações propostas para deixar o método mais eficiente e simples de ser implementado serão explicadas neste capítulo.

3.1 Definição Formal do CS

As meta-heurísticas híbridas têm obtido um grande sucesso na literatura quando aplicadas para resolver os mais diversos problemas de otimização combinatória. Nesse sentido, encontrar uma maneira interessante de combinar meta-heurísticas com outras técnicas de otimização tornou-se um desafio importante, para que problemas com dimensões cada vez maiores possam ser resolvidos de forma satisfatória e em um tempo aceitável.

O objetivo deste trabalho é fornecer, por meio do CS, um método híbrido que combine adequadamente meta-heurísticas e métodos de busca local, no qual a busca é intensificada somente em áreas que tenham um grande potencial de melhoria da solução. O CS acrescenta uma "inteligência" para auxiliar a escolha de em

quais soluções aplicar busca local, ao invés de escolher aleatoriamente ou aplicar busca local em todas as soluções. Assim, espera-se uma melhoria no processo de convergência associado a uma diminuição no esforço computacional em virtude do emprego mais racional dos métodos de busca local.

O CS procura dividir o espaço de busca e localizar regiões promissoras por meio do enquadramento dessas em *clusters*. Para os fins do CS, um cluster pode ser definido por três atributos $C = (c, v, r)$. O centro c_i é uma solução que representa o cluster C_i, identificando a sua localização dentro do espaço de busca. Ao invés de armazenar todas as soluções agrupadas no cluster, apenas parte das informações destas soluções são inseridas no centro do cluster. O volume v_i é a quantidade de soluções agrupadas no cluster C_i. Um cluster se torna promissor quando o volume atingir certo limitante λ. O índice de ineficácia r_i é uma variável de controle para identificar se a busca local está ou não melhorando o centro do cluster C_i. O valor de r_i indica o número de vezes consecutivas que a busca local foi aplicada no cluster C_i e não melhorou a solução. Este atributo evita que a busca local fique sendo executada por mais de r_{max} vezes em regiões ruins ou regiões que já tenham sido suficientemente exploradas.

Para que o algoritmo possa agrupar soluções em clusters é necessário definir alguma forma de medir a distância entre duas soluções. Sendo assim, uma função de medida de distância $d(i, j)$ é definida, a *priori*, para calcular a distância entre duas soluções como um número positivo, o qual é maior dependendo de quão mais distante estão as duas soluções.

Várias métricas de distância podem ser utilizadas, tais como, distância euclidiana, distância de Manhattan, distância de Hamming, distância de Levenshtein, entre outras. Neste trabalho utiliza-se a distância de Hamming[1], uma vez que esta métrica possui um bom custo-benefício em termos de qualidade e tempo computacional para calcular a distância entre duas soluções de um problema de otimização combinatória.

A distância de Hamming (HAMMING, 1950) entre duas soluções s_1 e s_2 é definida como o número de posições nas quais s_1 e s_2 são diferentes. A Figura 3.1 apresenta um exemplo do cálculo da distância de Hamming entre duas soluções com representação binária. A distância precisa ser definida para cada problema abordado levando-se em consideração a representação adotada e suas características.

[1]Outros tipos de distância não foram testados neste trabalho.

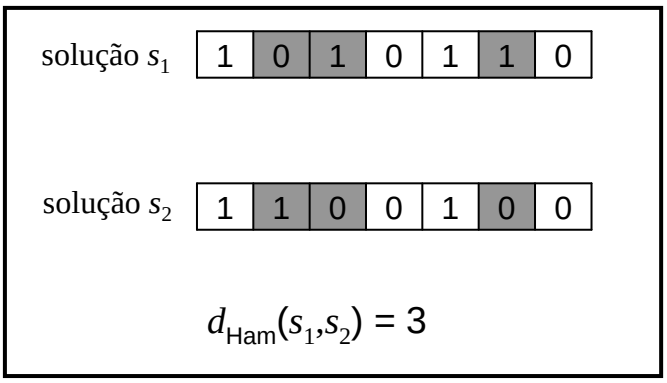

Figura 3.1 - Exemplo do cálculo da distância de Hamming.

O CS é um método iterativo que possui três componentes principais: uma meta-heurística, um processo de agrupamento e um método de busca local. A cada iteração do CS, uma solução s_k é gerada pela meta-heurística e enviada para o processo de agrupamento. Essa solução é então agrupada no cluster mais similar C_j, que é o cluster mais próximo à solução s_k. E, o centro deste cluster (c_j) é atualizado com informações contidas na nova solução agrupada por meio do processo de assimilação, fazendo com que o centro se desloque no espaço de busca. Em seguida é analisado o volume v_j do cluster. Caso esse volume tenha atingido um limitante λ, definido a *priori*, esse cluster pode estar em uma região de busca promissora. Porém, se o método de busca local não tiver obtido sucesso nas últimas r_{max} aplicações neste cluster promissor (índice de ineficácia $r_j \geq r_{max}$) é aplicada uma perturbação aleatória no centro c_j, objetivando escapar desta região do espaço de busca. Por outro lado, se r_j for menor que r_{max}, uma busca local é aplicada no centro c_j intensificando a busca na vizinhança deste cluster. A busca obtém sucesso em um cluster quando encontra uma solução que seja a melhor obtida neste cluster até o momento (c_j^*). Encerrado o processo de agrupamento, retorna-se para a meta-heurística que irá gerar outra solução.

O critério de parada do CS pode ser definido pelo número máximo de iterações do processo de agrupamento ou utilizar o critério de parada da meta-heurística escolhida.

A estratégia híbrida do CS para um problema de minimização pode ser descrita pelo fluxograma ilustrado na Figura 3.2, e o pseudocódigo do CS é apresentado na Figura 3.3. A seguir é descrito detalhadamente cada passo do CS.

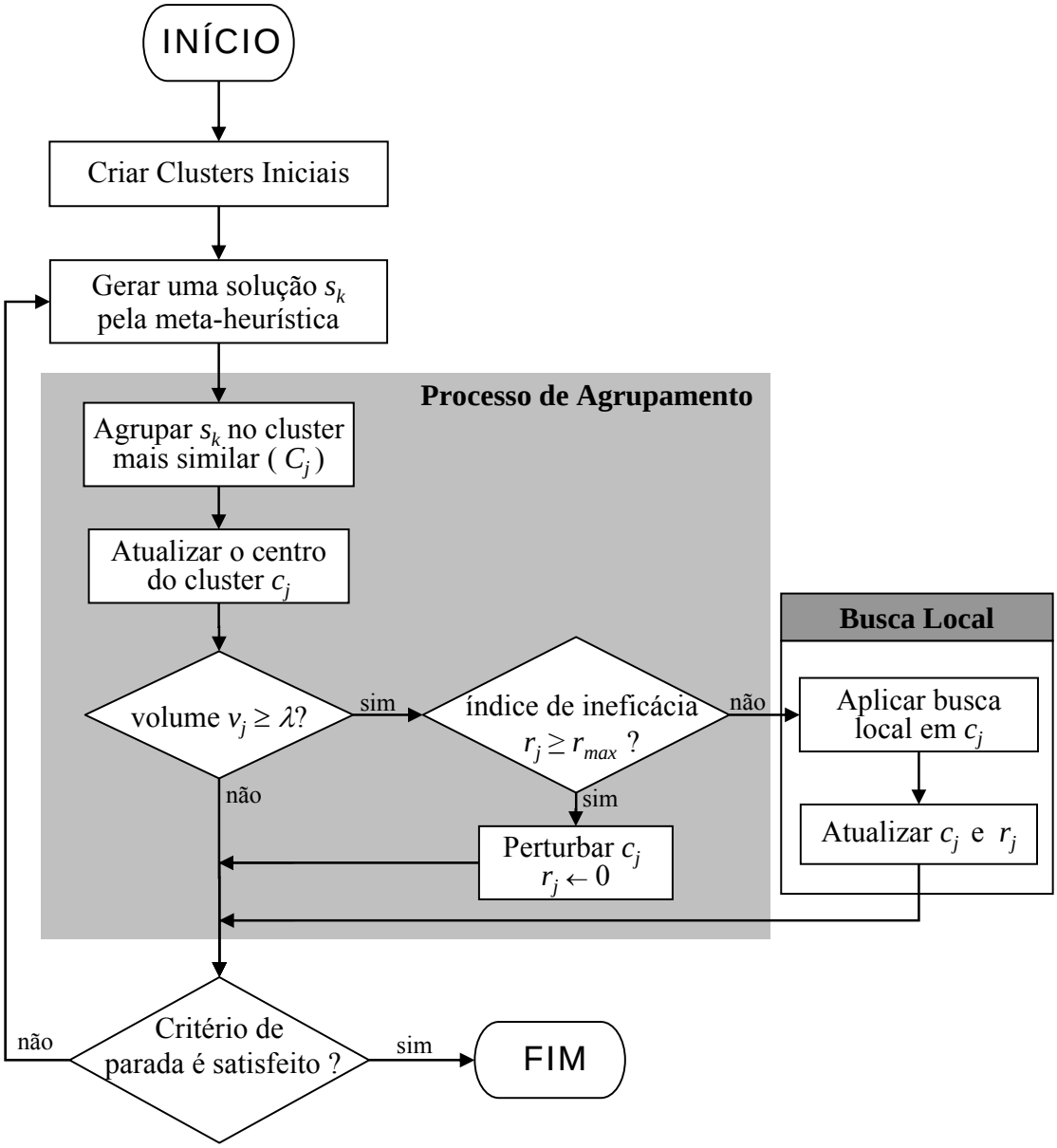

Figura 3.2 - Fluxograma do método CS.

algoritmo CS
crie os clusters iniciais
{ meta-heurística }
enquanto (critério de parada não for satisfeito) **faça**
gere uma solução (s_k) pela meta-heurística
{ processo de agrupamento }
encontre o cluster mais similar a s_k ($C_j \mid d(s_k,c_j) \equiv min \{ d(s_k,c) \}$)
agrupe s_k no cluster mais similar C_j ($v_j \leftarrow v_j + 1$)
atualize o centro do *cluster* ($c_j \leftarrow$ assimilação (c_j, s_k))
se ($v_j \geq \lambda$) **então**
reduza o volume $v_j \leftarrow 0$
se ($r_j \geq r_{max}$) **então**
aplique uma perturbação aleatória em c_j
zere o índice de ineficácia $r_j \leftarrow 0$
senão
{ heurística de busca local }
encontre o melhor vizinho ($\hat{c}_j$) de c_j
se ($f(\hat{c}_j) < f(c_j)$) **então**
atualize o centro $c_j \leftarrow \hat{c}_j$
se ($f(\hat{c}_j) < f(c_j^*)$) **então**
zere o índice de ineficácia $r_j \leftarrow 0$
atualize o melhor centro $c_j^* \leftarrow \hat{c}_j$
senão
aumente o índice de ineficácia $r_j \leftarrow r_j + 1$
fim-se
fim-se
fim-enquanto
fim-algoritmo

Figura 3.3 - Pseudocódigo do método CS.

3.2 Criar os Clusters Iniciais

Primeiramente, é necessário definir o número de clusters do CS, ou seja, a quantidade de regiões na qual o espaço de busca será dividido. Uma maneira de determinar o número de clusters é empiricamente. Um número pequeno de clusters pode fazer com que a busca fique presa em poucas regiões do espaço de busca, enquanto, um número de clusters muito grande pode tornar o processo de agrupamento muito caro computacionalmente.

Os clusters iniciais são criados contendo uma solução que representa sua localização no espaço de busca, chamada centro do cluster (c_i). Inicialmente, os valores de volume v_i e índice de ineficácia r_i são os mesmos para todos os clusters, sendo v_i definido com valor unitário ($v_i = 1$) e r_i com valor nulo ($r_i = 0$).

Os centros dos clusters devem ser gerados de forma a representar diferentes regiões do espaço de busca. Para tal, neste trabalho é proposto um método de criação dos centros dos clusters iniciais baseado na diversidade máxima entre eles. A medida de diversidade entre dois clusters pode ser representada pela distância entre os dois centros desses clusters.

Seja N um conjunto de n soluções aleatórias, $d(i, j)$ a distância entre as soluções i e j de N, e M um conjunto diverso com m soluções ($m < n$). O método para criar os centros dos clusters iniciais consiste em selecionar m soluções de N, de forma a maximizar a soma das distâncias entre essas soluções.

O problema de diversidade máxima é classificado como *NP-hard* (KUO et al., 1993) e sua resolução de forma ótima pode conduzir a um tempo computacional muito alto. Porém, devido ao fato dos clusters se deslocarem no espaço de busca ao longo das iterações do CS, é suficiente ter uma boa solução inicial para esse problema. Sendo assim, neste trabalho foi utilizado um método construtivo, proposto por Silva et al. (2006), para obter uma boa configuração inicial dos clusters, representando de maneira satisfatória o espaço de busca.

Inicialmente é necessário montar uma matriz simétrica $D[n \times n]$ que armazena as distâncias entre cada uma das soluções de N, preenchida da seguinte maneira: $d(i, j) = d(j, i)$ e $d(i, i) = 0, \forall (i, j) \in N$. O valor da diversidade de M é determinado pelo somatório das distâncias entre as soluções selecionadas para esse conjunto

$$\left(\sum_{i,j \in M, i<j} d(i,j) \right).$$

O método construtivo se baseia na heurística de inserção mais distante. A primeira solução do conjunto M (m_1) é escolhida aleatoriamente entre todas as soluções pertencentes a N. A segunda solução a ser selecionada deve ser a solução $j \in N - m_1$ que forneça a maior diversidade entre m_1 e j. A partir da terceira solução é necessário calcular $D_{soma}(i)$, $\forall i \in N - M$, que é a soma das diversidades entre o candidato i e todas as soluções que pertencem ao conjunto parcial M. A solução i que tiver o maior valor de D_{soma} é selecionada para ser incluída no conjunto M. O processo é repetido até m soluções serem selecionadas. Cada solução do conjunto M será o centro de um cluster do CS. Esse método é executado diversas vezes, e a melhor configuração obtida é escolhida para ser os clusters iniciais.

A Figura 3.4 apresenta o pseudocódigo do método para criar os clusters iniciais.

```
algoritmo CriarClusters (n, m)
    gere um conjunto de n soluções aleatórias ( N )
    M = ∅
    escolha aleatoriamente a primeira solução m1 ∈ N
    M ← M ∪ {m1}
    para cada ( j ∈ N – m1) faça
        calcule d ( j, m1 )
    fim-para
    m2 = max {d ( k, m1), k ∈ N – m1) }
    M ← M ∪ {m2}
    para ( c = 3 até m ) faça
        calcule Dsoma(i ), ∀i ∈ N – M
        mc = max { Dsoma (k), k ∈ N – M }
        M ← M ∪ {mc}
    fim-para
fim-algoritmo
```

Figura 3.4 - Pseudocódigo do método para criar os clusters iniciais.

Fonte: Adaptado de Silva et al. (2006).

Neste trabalho utilizou-se $|N| = 200$ e $|M| = 20$, ou seja, em todas as abordagens do CS são geradas 200 soluções aleatoriamente e selecionam-se 20 soluções que sejam diversas entre si para representarem os clusters. Alguns testes variando o número de clusters foram realizados, objetivando analisar a influência que o número de clusters tem no comportamento do CS. Estes testes serão apresentados no Capítulo 7.

3.3 Gerar Soluções para o Processo de Agrupamento

Uma meta-heurística trabalha como um gerador de soluções para o processo de agrupamento. Pode-se utilizar qualquer meta-heurística. Porém, é importante que esta seja capaz de gerar um grande número de soluções diferentes, possibilitando uma análise ampla do espaço de busca. Em outras palavras, a meta-heurística precisa levar em consideração principalmente o critério de diversificação.

As meta-heurísticas utilizadas neste trabalho foram Recozimento Simulado (SA), Algoritmo Genético (GA), Busca Local Iterativa (ILS) e Pesquisa em Vizinhança Variável (VNS) (ver Seção 2.2). Outras meta-heurísticas também poderiam ter sido utilizadas, mas optou-se por essas principalmente pelo fato de possuírem bons mecanismos de diversificação, e também porque são capazes de produzir soluções rapidamente.

A meta-heurística é executada independentemente dos demais componentes, sendo assim, as soluções armazenadas nos centros dos clusters não causam interferência no processo de busca da meta-heurística.

O gerador de soluções não precisa ser necessariamente uma meta-heurística, pode-se utilizar também um método de construção de soluções aleatórias. Porém, esse método não leva em consideração informações sobre o espaço de busca do problema e, portanto, tende a demorar mais tempo para convergir para boas soluções. Neste trabalho foram realizados testes substituindo a meta-heurística por um método aleatório.

Para realizar uma comparação entre os diferentes métodos de gerar soluções foi definido um número fixo de soluções que devem ser enviadas para o processo de agrupamento. Assim, é possível realizar uma análise da qualidade das soluções do CS e do tempo necessário para obtê-las, independentemente do método utilizado. Neste trabalho foram enviadas 10000 soluções para o processo de agrupamento.

Nas meta-heurísticas VNS e ILS e no método aleatório as soluções geradas a cada iteração são enviadas para o processo de agrupamento do CS. Neste caso, o critério de parada destes métodos foi estabelecido como o número de iterações igual a 10000.

A implementação do VNS e do ILS objetivou uma maior diversificação desses métodos. Portanto, no VNS utilizou-se estruturas de vizinhança distantes da solução corrente e variando de acordo com o tamanho do problema. A perturbação do ILS também é baseada no tamanho do problema tratado. Além disso, em ambas as meta-heurísticas foi implementado um critério de aceitação probabilístico, aceitando soluções que piorem o valor da função objetivo com uma probabilidade de 5%.

No SA há o problema dos vizinhos gerados serem muito similares às soluções correntes, sendo assim, não é possível enviar cada solução vizinha para o processo de agrupamento. Ao invés disso, um intervalo precisa ser definido, permitindo uma diversificação da solução. Este intervalo está relacionado à quantidade de soluções geradas pelo SA durante o processo de busca. Portanto, só é possível estabelecê-lo após ter definido os demais parâmetros do SA. Para haver uma maior diversificação do SA foi implementado uma fase de reaquecimento, sendo a temperatura corrente aquecida a 30% da temperatura inicial quando esta atingir o valor de 0,01 pela primeira vez.

O GA poderia enviar todos os filhos para o processo de agrupamento. Porém, para respeitar o limite de 10000 soluções, seria necessário utilizar uma população pequena ou senão poucas gerações. Mas essas configurações podem impedir que a população evolua adequadamente. Portanto, optou-se por enviar somente uma porcentagem dos filhos para o processo de agrupamento. Essa amostragem de filhos também depende dos parâmetros do GA e, portanto, precisa ser definida após a calibragem dos mesmos.

3.4 Processo de Agrupamento de Soluções

O processo de agrupamento trabalha como um classificador, conservando no sistema somente informações relevantes e direcionando a busca para regiões supostamente promissoras. O objetivo é reunir soluções similares dentro de clusters, mantendo uma solução no centro do cluster que seja representativa para as demais soluções. Para evitar um esforço computacional extra, o agrupamento é desenvolvido como um processo iterativo, no qual os clusters são progressivamente alimentados por soluções

geradas em cada iteração da meta-heurística.

A medida de distância definida inicialmente é utilizada para calcular a similaridade entre uma solução e os clusters. As soluções geradas pela meta-heurística são enviadas para o processo de agrupamento que procura agrupá-las no cluster mais similar. O cluster que possuir a menor distância entre o centro (c_j) e a solução gerada pela meta-heurística (s_k) é considerado o cluster mais similar (C_j). Caso aconteça de dois ou mais clusters terem a mesma distância para a solução s_k, o cluster mais similar é escolhido aleatoriamente entre estes.

As soluções geradas pela meta-heurística não são armazenadas nos clusters. Ao invés disso, apenas algumas características dessas soluções são inseridas no centro do cluster considerado mais similar e o volume do cluster é aumentado em uma unidade ($v_j = v_j + 1$).

A "inserção" de uma nova solução em um cluster deve causar uma perturbação em seu centro. Tal perturbação é chamada de assimilação e consiste basicamente em atualizar o centro com informações da nova solução.

Segundo Oliveira (2004) pode-se utilizar três formas diferentes de assimilação:

a) **assimilação simples**: insere uma porcentagem, definida a *priori*, das características da solução s_k no centro c_j;

b) **assimilação por recombinação**: realiza um cruzamento entre o centro c_j e a solução s_k gerando um novo centro com características de ambas soluções;

c) **assimilação por caminho**: analisa um conjunto de soluções que são geradas no caminho que interconecta o centro c_j e a solução s_k.

A assimilação simples e a assimilação por recombinação geram uma única solução que será o novo centro do cluster. Na assimilação por caminho podem ser geradas várias soluções e o centro do cluster será a melhor solução encontrada. Apesar do custo computacional, a assimilação por caminho tem mostrado ser a melhor técnica de assimilação para o CS. O fato de deslocar o centro para a melhor solução faz com

que esse seja um bom ponto de partida para a busca local, desta forma, necessita-se de um número pequeno de modificações para encontrar um ótimo local.

Na assimilação por caminho é utilizado o método Reconexão por Caminhos (PR, do inglês *Path-Relinking*) (GLOVER, 1996), que realiza movimentos exploratórios na trajetória que interconecta duas soluções. Assim sendo, o processo de assimilação é responsável por intensificar e diversificar a busca dentro de um cluster, pois o centro será deslocado para a melhor solução avaliada nessa trajetória mesmo que ela seja pior que o centro corrente.

O PR inicializa a partir de duas soluções. A primeira é o centro do cluster mais similar c_j (s_{inicio}). A segunda é a solução gerada pela meta-heurística s_k (s_{guia}). O método inicializa calculando a diferença simétrica entre as duas soluções $\Delta(s_{inicio}, s_{guia})$, que é o conjunto de movimentos necessários para alcançar s_{guia} a partir de s_{inicio}. Um caminho de soluções é gerado, conectando s_{inicio} e s_{guia}. A cada passo, o método examina todos os movimentos $mv \in \Delta(s_{inicio}, s_{guia})$ da solução corrente s e seleciona o movimento que resultar na solução de melhor custo, aplicando o melhor movimento (mv^*) na solução s ($s \oplus mv^*$). O conjunto de movimentos possíveis é atualizado. O método termina quando a solução s_{guia} for alcançada ou quando uma porcentagem do caminho for analisada. A melhor solução neste caminho será o novo centro do cluster. A Figura 3.5 ilustra o pseudocódigo do PR e a Figura 3.6 mostra um exemplo de uma aplicação do PR.

algoritmo Reconexão por Caminho (s_{inicio}, s_{guia})
  calcule a diferença simétrica Δ (s_{inicio}, s_{guia})
  $s \leftarrow s_{inicio}$
  enquanto (critério de parada não for satisfeito) **faça**
    $mv^* = \min\{ f(s \oplus mv): mv \in \Delta(s_{initial}, s_{guide})\}$
    $\Delta(s \oplus mv^*, s_{guide}) = \Delta(s, s_{guide}) \backslash \{ mv^* \}$
    $s = s \oplus mv^*$
  fim-enquanto
fim-algoritmo

Figura 3.5 - Pseudocódigo do método Reconexão por Caminho.

Fonte: Adaptado de Resende e Ribeiro (2005).

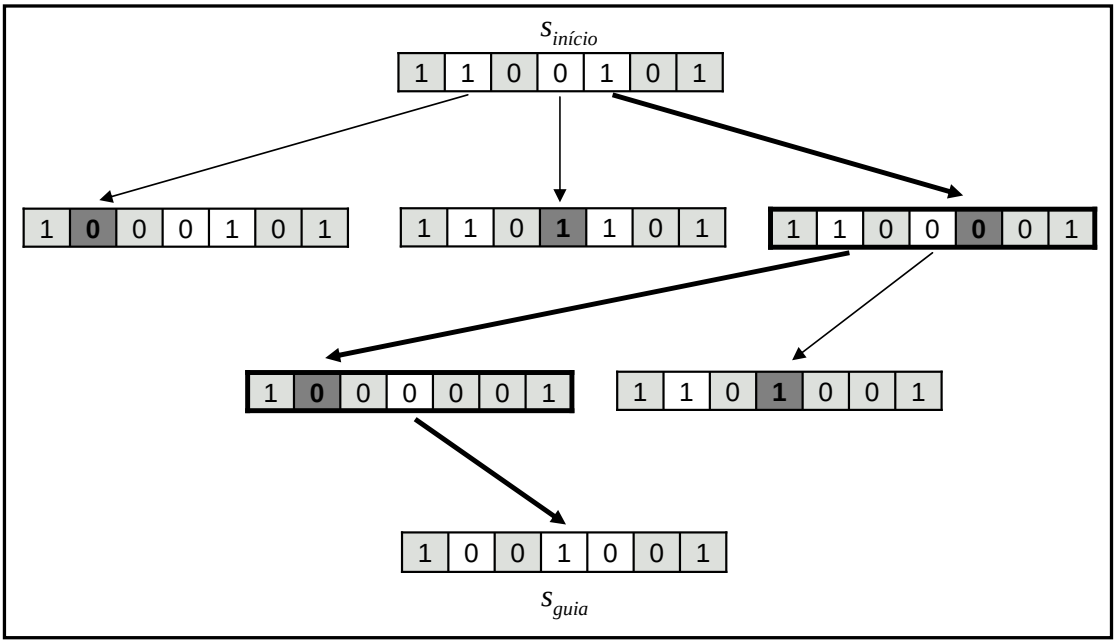

Figura 3.6 - Exemplo de uma aplicação do método Reconexão por Caminho.

Na atualização do centro não é interessante perder toda a informação referente à localização do cluster, ou seja, mover o centro para soluções muito distantes do centro atual. Por causa disso, o métoto PR deve ser trucado, analisando somente uma porcentagem fixa, definida a *priori*, do caminho entre o centro c_j e a solução s_k. Essa porcentagem geralmente varia entre 10% e 50% do caminho, e precisa ser definida para cada problema abordado. O centro é deslocado para a melhor solução encontrada no caminho percorrido.

Após realizar a assimilação é preciso conduzir uma análise do cluster ativado, verificando se este pode ser considerado promissor. Um cluster se torna promissor quando o volume v_j atinge certo limitante λ e este cluster está em uma boa região de busca. Esta última condição é alcançada se a heurística de busca local tiver obtido sucesso no centro c_j recentemente. O índice de ineficácia r_j provê essa memória, sendo sempre atualizado após a busca local ser executada. Para que a heurística tenha sucesso em um cluster é necessário que a solução encontrada por ela seja a melhor solução obtida até o momento dentro deste cluster (c_j^*). Neste trabalho foi definido que o componente de busca local pode ficar no máximo cinco iterações consecutivas sem obter sucesso em um centro j ($r_{max} = 5$).

A busca é intensificada no centro de um cluster considerado promissor, ou seja, $v_j \geq \lambda$ e $r_j < r_{max}$, aplicando heurísticas de busca local específicas para o problema abordado. O centro do cluster é atualizado se a solução encontrada pela heurística for melhor que o centro atual. Já o índice de ineficácia é zerado ou aumentado em uma unidade, dependendo se a heurística melhora ou não a melhor solução encontrada neste cluster.

Por outro lado, se o volume de um cluster atingir λ ($v_j \geq \lambda$) e o índice de ineficácia for alto ($r_j \geq r_{max}$), indicando que esta é uma região ruim ou que já foi suficientemente explorada pela heurística, aplica-se uma perturbação aleatória no centro c_j. Essa perturbação visa permitir que o cluster escape dessa região e pesquise outras áreas do espaço de busca. A perturbação é realizada aplicando movimentos aleatórios em um percentual da solução.

Essa perturbação também se faz necessária em razão do CS utilizar um número fixo de clusters. Para evitar que informações sobre o espaço de busca sejam perdidas por meio da eliminação de um cluster, optou-se por permitir uma maior movimentação dos clusters no espaço de busca, sempre atualizando o centro com características contidas na nova solução agrupada e perturbando o centro quando o volume do cluster for igual a λ e o índice de ineficácia igual a r_{max}.

Sempre que o volume de um cluster atingir λ, independentemente de ser aplicado busca local ou perturbação, é preciso reduzir o volume ($v_j \leftarrow 0$). Isso impede que um cluster seja novamente considerado promissor já na próxima iteração em que uma solução for agrupada ao mesmo.

3.5 Heurística de Busca Local

O componente de busca local é um módulo de busca que provê a exploração de uma suposta região promissora, delimitada pelo cluster. Este processo acontece após ser descoberto um cluster promissor e uma intensificação da busca é aplicada no centro desse cluster. Neste trabalho utilizam-se heurísticas de busca local, analisando a vizinhança próxima ao centro do cluster. Entretanto, outras técnicas de otimização também poderiam ser empregadas neste componente.

Em problemas de otimização, as heurísticas de busca local (ou heurísticas de refinamento) constituem uma família de técnicas baseadas na noção de vizinhança. Essas heurísticas partem de uma solução inicial e caminham, a cada iteração, de

vizinho para vizinho de acordo com a definição de vizinhança adotada.

Essa definição de vizinhança é crucial em uma heurística de busca local. A partir de uma solução s do espaço de busca deve ser sempre possível atingir qualquer outra solução em um número finito de passos, utilizando um determinado tipo ou tipos de movimentos.

Considerando que um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a outra estrutura de vizinhança, Mladenovic e Hansen (1997) propuseram o método de Descida em Vizinhança Variável (VND, do inglês *Variable Neighborhood Descent*) que consiste em explorar o espaço de busca por meio de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O VND pode ser utilizado para intensificar a busca em uma região promissora, permitindo analisar um número maior de soluções ao redor do cluster. Desta forma, a utilização de diferentes heurísticas estabelece diferentes vizinhanças e, consequentemente, diferentes visões do mesmo espaço de busca. A Figura 3.7 apresenta o pseudocódigo do VND.

```
algoritmo VND ( s )
    seja k_max o número de estruturas diferentes de vizinhança
    k ← 1
    enquanto ( k ≤ k_max ) faça
        encontre o melhor vizinho s' de s ( s' ∈ N^k(s) )
        se ( f(s') < f(s) ) então
            s ← s'
            k ← 1
        senão
            k ← k + 1
    fim-enquanto
fim-algoritmo
```

Figura 3.7 - Pseudocódigo do método VND.

Fonte: Adaptado de Hansen e Mladenovic (2003a).

4 PROBLEMA DE P-MEDIANAS CAPACITADO

4.1 Introdução

Problemas de localização de facilidades possuem várias aplicações práticas em setores públicos e privados, tais como, a localização de escolas, postos de saúde, corpo de bombeiros, ambulâncias, viaturas de polícia, pontos de ônibus, aplicações que envolvem a escolha de locais para instalação de fábricas, torres de transmissão, lojas de franquias, depósitos, etc. Esses problemas tratam a questão de onde localizar facilidades, considerando clientes que devem ser servidos, de forma a otimizar algum critério. A maioria dos problemas de localização de facilidades é considerada de difícil solução.

Um dos mais conhecidos e estudados problemas de localização de facilidades é o problema de p-medianas (HAKIMI, 1964; HAKIMI, 1965). Esse problema consiste em localizar p facilidades (medianas) em um dado espaço (por exemplo, espaço Euclidiano), satisfazendo n pontos de demanda de tal forma que a soma total das distâncias entre cada ponto de demanda e sua mediana mais próxima seja minimizada. Esse problema é classificado como NP-*hard* (GAREY; JOHNSON, 1990).

O problema de p-medianas pode ser generalizado definindo-se uma capacidade fixa para cada mediana candidata e uma demanda para cada ponto de atendimento, sendo que, a soma das demandas de todos os pontos alocados a uma mediana não pode exceder sua capacidade. Essa generalização é conhecida como Problema de p-Medianas Capacitado (CPMP, do inglês *Capacitated p-Median Problem*) (MULVEY; BECK, 1984). A restrição de capacidade acrescenta uma fator realístico ao problema, uma vez que, muitos dos problemas encontrados na prática possuem uma limitação de atendimento por parte das facilidades aos pontos de demanda.

O Problema de Agrupamento Centrado Capacitado (CCCP, do inglês *Capacitated Centered Clustering Problem*), proposto por Negreiros e Palhano (2006), é similar ao CPMP. A diferença desse problema está no fato de existir, ao invés de medianas, um centróide para cada agrupamento de pontos de demanda. Os centróides são calculados a partir da média das coordenadas dos pontos de demanda alocados no agrupamento.

O CPMP e o CCCP foram resolvidos de forma aproximada por meio do CS. Neste trabalho foram testadas diferentes meta-heurísticas para gerar soluções para

o processo de agrupamento e também um gerador de soluções aleatórias. Todas as versões do CS obtiveram bons resultados. Neste capítulo são apresentadas essas abordagens.

4.2 Definição e Formulação Matemática

O CPMP é um problema de particionamento de conjunto de n clientes (pontos), no qual cada cliente possui uma demanda, em p agrupamentos disjuntos, de maneira que a dissimilaridade (tempo ou distância) total dentro de cada agrupamento seja minimizada e a restrição de capacidade de cada agrupamento seja respeitada. A dissimilaridade total de cada agrupamento é calculada como a soma das dissimilaridades existentes entre cada ponto de demanda e sua mediana atribuída ao agrupamento.

O modelo matemático do CPMP é apresentado a seguir.

$$z = \min \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ij} \tag{4.1}$$

sujeito a:

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in N \tag{4.2}$$

$$\sum_{j \in N} x_{jj} = p \tag{4.3}$$

$$\sum_{i \in N} q_i x_{ij} \leq Q_j x_{jj} \qquad \forall j \in N \tag{4.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in N, \forall j \in N \tag{4.5}$$

no qual,

- $N = \{1,...,n\}$ é o conjunto de índices dos pontos de demanda a serem alocados e também das possíveis medianas;
- p é o número de medianas que serão localizadas;
- q_i é a demanda de cada ponto;
- Q_j é a capacidade total de cada possível mediana;
- $[t_{ij}]_{n \times n}$ é a matriz de tempos (distâncias) entre os pontos;
- $[x_{ij}]_{n \times n}$ é a matriz de alocações;
- $x_{ij} = \begin{cases} 1, & \text{se o ponto } i \text{ é alocado para a mediana } j, \\ 0, & \text{caso contrário.} \end{cases}$
- $x_{jj} = \begin{cases} 1, & \text{se a mediana } j \text{ é selecionada,} \\ 0, & \text{caso contrário.} \end{cases}$

A função objetivo 4.1 corresponde a soma das distâncias entre as medianas e seus pontos alocados. As restrições 4.2 asseguram que cada ponto é alocado para exatamente uma mediana. A restrição 4.3 define o número de medianas localizadas. As restrições 4.4 impõem que a capacidade total de cada mediana precisa ser respeitada, e as restrições 4.5 mantém a integralidade das variáveis x_{ij}.

4.3 Revisão Bibliográfica

O CPMP aparece na literatura também com os nomes Problema de Agrupamento Capacitado (*Capacitated Clustering Problem*), Problema de Localização de Depósitos Capacitado (*Capacitated Warehouse Location Problem*), Problema de Agrupamento da Soma das Estrelas (*Sum-of-Stars Clustering Problem*), entre outros. Vários métodos de solução têm sido propostos para esses problemas.

Osman e Christofides (1994) utilizam um método híbrido combinando Recozimento Simulado (SA) e Busca Tabu (TS) para resolver o CPMP, mostrando que a abordagem híbrida obteve melhores resultados que o SA e o TS aplicados separadamente. Os autores também apresentam uma revisão bibliográfica sobre problemas similares ao CPMP e um conjunto de instâncias para este.

Ceselli (2003) propõe duas abordagens de solução exata para o CPMP. A primeira apresenta um algoritmo *Branch* e *Bound*, utilizando técnicas de relaxação Lagrangiana e o método de otimização do subgradiente. A segunda abordagem apresenta um algoritmo *Branch* e *Price*, testando a eficácia de técnicas baseadas na geração de colunas para o CPMP.

Lorena e Senne (2003) exploram heurísticas de busca local, baseadas no método de Localização-Alocação, procurando melhorar as soluções obtidas pela relaxação Lagrangiana/Surrogate. Essa abordagem foi aplicada a um conjunto de instâncias clássicas da literatura e a um conjunto de instâncias com dados reais coletados da região central da cidade de São José dos Campos. Lorena e Senne (2004) apresentam uma aplicação de geração de colunas para o CPMP, explorando uma nova alternativa para estabilizar a geração de colunas quando aplicada a problemas de localização.

Diaz e Fernandez (2006) apresentam alguns algoritmos para o CPMP. Inicialmente é proposta uma meta-heurística GRASP para obter um conjunto inicial de soluções de referência. Em seguida é aplicado um método *Scatter Search*, no qual a cada iteração as soluções geradas pelo GRASP são combinadas entre si para gerar novas soluções, as quais podem substituir alguma solução do conjunto de referência. Outra abordagem apresentada é a aplicação da heurística path-relinking (PR) em pares de soluções do conjunto de referência obtido pelo GRASP, executando uma série de passos para transformar uma solução em outra. Uma terceira abordagem que combina PR com *Scatter Search* também é analisada. Scheuerer e Wendolsky (2006) também apresentam um método híbrido combinando *Scatter Search* e PR.

Fleszar e Hindi (2008) resolvem o CPMP em duas etapas. A primeira etapa é resolver o problema de localização das medianas e a segunda é resolver o problema de atribuição dos clientes para as medianas selecionadas. Na primeira etapa é utilizado a meta-heurística Pesquisa em Vizinhança Variável (VNS) para gerar diferentes conjuntos de medianas e na segunda etapa é utilizado o *software* CPLEX para resolver o problema de atribuição generalizado, definindo a alocação dos clientes às medianas.

Boccia et al. (2008) propuseram um algoritmo de planos de corte baseado nos cortes de Fenchel (BOYD, 1993), conseguindo resolver algumas instâncias para as quais ainda não se conhecia a solução ótima. A formulação fortalecida com os cortes de Fenchel foi resolvida pelo *software* CPLEX.

4.4 Representação do Problema

Uma solução do CPMP é representada por dois vetores de valores inteiros. O primeiro vetor contém n posições, sendo que a posição i representa a mediana na qual está alocado o ponto de demanda i. O segundo vetor contém p posições, sendo que cada posição j representa uma mediana selecionada. A Figura 4.1 ilustra um exemplo de uma solução com 10 pontos de demanda $(n_1, n_2, ..., n_{10})$ e 3 medianas (p_1, p_2 e p_3).

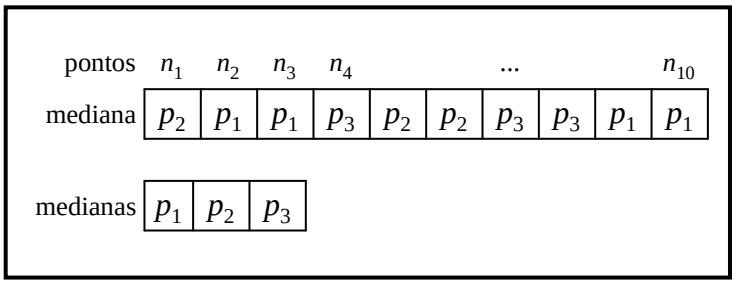

Figura 4.1 - Exemplo da representação de uma solução do CPMP.

Esta representação foi escolhida porque permite a definição de tipos simples de movimentos (conforme especificado na Seção 4.5), possibilitando uma navegação eficaz pelo espaço de busca. Além disso, essa representação torna mais fácil o cálculo da função objetivo, a qual está baseada na alocação ponto/mediana.

4.5 Estruturas de Vizinhança

Quatro movimentos são propostos possibilitando que qualquer solução do espaço de busca possa ser alcançada com um número finito de operações. Os movimentos são:

- m_1: trocar uma mediana por um dos pontos alocados a ela;
- m_2: trocar uma mediana por um ponto qualquer;
- m_3: trocar um ponto de uma mediana com um ponto de outra mediana;
- m_4: transferir um ponto de uma mediana para outra mediana.

Observe que os movimentos m_3 e m_4 podem produzir inviabilidade, isto é, ao permutar ou inserir um ponto de demanda em uma mediana permite-se que sejam

geradas soluções que extrapolem a capacidade das medianas, sendo a função objetivo dessas soluções penalizada.

Após realizar a troca de medianas nos movimentos m_1 e m_2, um método para realocar os pontos precisa ser aplicado. Esse método será explicado na Subseção 4.8.3.

4.6 Função Objetivo

A função objetivo do CPMP é construída calculando as distâncias entre os pontos de demanda e suas medianas. Uma penalidade é acrescentada caso a capacidade da mediana seja excedida. Portanto, uma solução é avaliada com base na seguinte função objetivo, a qual deve ser minimizada:

$$f(s) = \sum_{j=1}^{p} (T_j + \varpi * f_j) \tag{4.6}$$

sendo T_j a soma das distâncias entre a mediana j e os pontos de demanda alocados a ela, e ϖ o peso que reflete a penalidade imposta se a restrição de capacidade da mediana j for violada. O componente f_j assume valor 1 se a capacidade da mediana j for extrapolada, e 0 caso contrário. Neste trabalho utilizou-se $\varpi = 10^5$.

4.7 Medida de Distância entre duas Soluções

A medida de distância entre duas soluções do CPMP é o número de pontos de demanda atribuídos para medianas diferentes nas soluções. Sendo assim, a distância entre duas soluções será maior quando houver muitos pontos alocados em medianas diferentes, indicando que estas soluções não são similares.

Essa medida de distância permite analisar a similaridade entre duas soluções com relação à alocação dos pontos de demanda e também com relação às medianas selecionadas, possibilitando um agrupamento adequado das soluções.

4.8 CS aplicado ao CPMP

Uma abordagem do CS para o CPMP é descrita a seguir, apresentando detalhes do método. Cinco versões do CS foram implementadas variando somente o componente gerador de soluções para o processo de agrupamento.

Primeiramente é preciso definir um método para gerar soluções viáveis. Uma solução é gerada de forma parcialmente aleatória. O método começa determinando todas as medianas e pontos de demanda disponíveis. A seguir, selecionam-se aleatoriamente p medianas. Para cada ponto de demanda i é verificado qual a mediana mais próxima a i (menor distância) que não exceda a capacidade. O ponto i é alocado nessa mediana e retirado da lista de pontos disponíveis. A Figura 4.2 apresenta o pseudocódigo desse método.

---

algoritmo GerarSolucaoViavel
  seja N o conjunto de pontos de demanda disponíveis
  seja P o conjunto de medianas disponíveis
  para (j de 1 até p) **faça**
    selecione aleatoriamente uma mediana $p_j \in P$
    $P \leftarrow P - \{p_j\}$
  fim-para
  para (i de 1 até n) **faça**
    selecione aleatoriamente um ponto de demanda $k \in N$
    $j = \min\left\{t_{kj} \mid q_k \leq Q_j\right\}$
    aloque o ponto de demanda k na mediana j e atualize Q_j
    $N \leftarrow N - \{k\}$
  fim-para
fim-algoritmo

Figura 4.2 - Pseudocódigo do método para gerar uma solução viável do CPMP.

4.8.1 Meta-heurísticas

No componente gerador de soluções foram implementadas quatro meta-heurísticas clássicas e um algoritmo de geração de soluções aleatórias. Este foi baseado no método de gerar soluções apresentado anteriormente, sendo geradas 10000 soluções que são enviadas iterativamente para o processo de agrupamento do CS. Todas as meta-heurísticas fazem uso das estruturas de vizinhança definidas, exceto o Algoritmo Genético que possui operadores próprios para evoluir a população. Os detalhes de cada meta-heurística são descritos a seguir.

4.8.1.1 Algoritmo Genético

O Algoritmo Genético (GA) repete as operações de seleção, cruzamento e mutação por um dado número de gerações (ver Subseção 2.2.1). A população inicial é definida gerando μ indivíduos aleatoriamente.

O GA emprega o operador de seleção por torneio, selecionando μ pais para participarem do processo de evolução. A cada iteração três indivíduos da população corrente são escolhidos aleatoriamente, sendo o indivíduo com melhor valor de função objetivo selecionado para fazer parte da população de pais.

O operador de cruzamento combina dois pais gerando dois filhos. Um cruzamento uniforme (HOLLAND, 1975) foi implementado para o CPMP, sendo aplicado sobre o vetor das medianas. Inicialmente, as medianas dos pais são analisadas definindo quais medianas podem ser trocadas, isto é, as medianas diferentes entre os dois pais. Em seguida, para cada mediana diferente é verificado a probabilidade do filho herdar a mediana de um pai ou do outro, com 50% de chance para cada pai. O segundo filho é criado com o mapeamento inverso. A Figura 4.3 exemplifica esse operador.

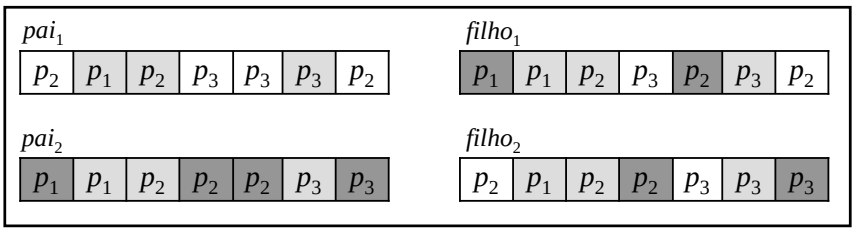

Figura 4.3 - Exemplo do cruzamento uniforme para o CPMP.

A mutação empregada neste trabalho altera aleatoriamente a alocação de um ponto de demanda. Para cada ponto, se a probabilidade de mutação for aceita, é escolhida uma nova mediana qualquer e o ponto é alocado a esta mediana. A população de filhos após a aplicação da mutação se torna a nova população.

Os operadores de cruzamento e mutação ocorrem com certa probabilidade definida como $p_c = 0,95$ e $p_m = 0,05$. O tamanho da população (μ) é de 100 indivíduos e o GA é executado por 1000 gerações. Sendo assim, a cada geração uma amostragem aleatória de 10% dos filhos é enviada para o processo de agrupamento.

4.8.1.2 Recozimento Simulado

Partindo de uma solução inicial, gerada aleatoriamente, o algoritmo segue os passos do Recozimento Simulado (SA) tradicional (ver Subseção 2.2.2), explorando a vizinhança da solução corrente por meio dos movimentos de uma das estruturas de vizinhança definidas, a qual é escolhida aleatoriamente a cada iteração.

Os parâmetros de controle do SA são a taxa de resfriamento α, o número de iterações para cada temperatura (SA_{max}) e a temperatura inicial (T_0). Neste trabalho foram utilizados $\alpha = 0,95$, $SA_{max} = 2000$ e $T_0 = 10^6$. A cada 200 vizinhos gerados, a solução corrente do SA é enviada para o processo de agrupamento.

4.8.1.3 Pesquisa em Vizinhança Variável

A meta-heurística Pesquisa em Vizinhança Variável (VNS) (ver Subseção 2.2.3) parte de uma solução inicial aleatória e explora vizinhanças cada vez mais distantes durante o processo de busca.

No VNS utiliza-se cinco estruturas de vizinhança aninhadas ($k_{max} = 5$), nas quais k medianas são aleatoriamente trocadas por um ponto de demanda qualquer (movimento m_2). Cada estrutura de vizinhança gera um vizinho a uma distância φ da solução corrente, sendo que, a distância está relacionada ao número de medianas do problema. Então, o número de medianas trocadas em cada vizinhança é dado por $k = \varphi \times p$. Neste trabalho foi definido $\varphi = \{0,10; 0,15; 0,20; 0,25; 0,30\}$.

O método de busca local utiliza a heurística Troca e Transferência (LORENA; SENNE, 2003), que será explicada na Subseção 4.8.3, sendo baseada em mudanças da alocação dos pontos de demanda.

4.8.1.4 Busca Local Iterativa

Na Busca Local Iterativa (ILS) (ver Subseção 2.2.4) uma solução inicial é gerada aleatoriamente. A cada iteração, uma perturbação é aplicada na solução corrente por meio da troca de algumas medianas por um ponto de demanda qualquer (movimento m_2). O número de trocas da perturbação é uma porcentagem β do número de medianas do problema. O valor de β varia aleatoriamente de 25% a 75%.

A heurística Troca e Transferência (LORENA; SENNE, 2003) (ver Subseção 4.8.3) foi utilizada para obter um ótimo local da solução gerada.

4.8.2 Processo de Agrupamento

O processo de agrupamento executa uma classificação de cada solução enviada pelo gerador de soluções, agrupando a solução s_k no cluster mais similar, isto é, o cluster com a menor distância entre o centro e a solução s_k.

A solução s_k e o centro c_j mais próximo participam da assimilação por meio do método Reconexão por Caminho (PR), utilizando as novas informações contidas na solução s_k para causar um deslocamento na localização do centro.

Para analisar o caminho que conecta as duas soluções, um movimento do PR consiste em trocar uma mediana do centro c_j por uma mediana da solução s_k e realocar os pontos de demanda por meio do método Realocar (ver Subseção 4.8.3). Apenas 50% do caminho é analisado, evitando que o centro seja deslocado para uma solução muito distante. A melhor solução encontrada ao final do processo passa a ser o novo centro c_j. A Figura 4.4 ilustra um exemplo do PR aplicado ao CPMP.

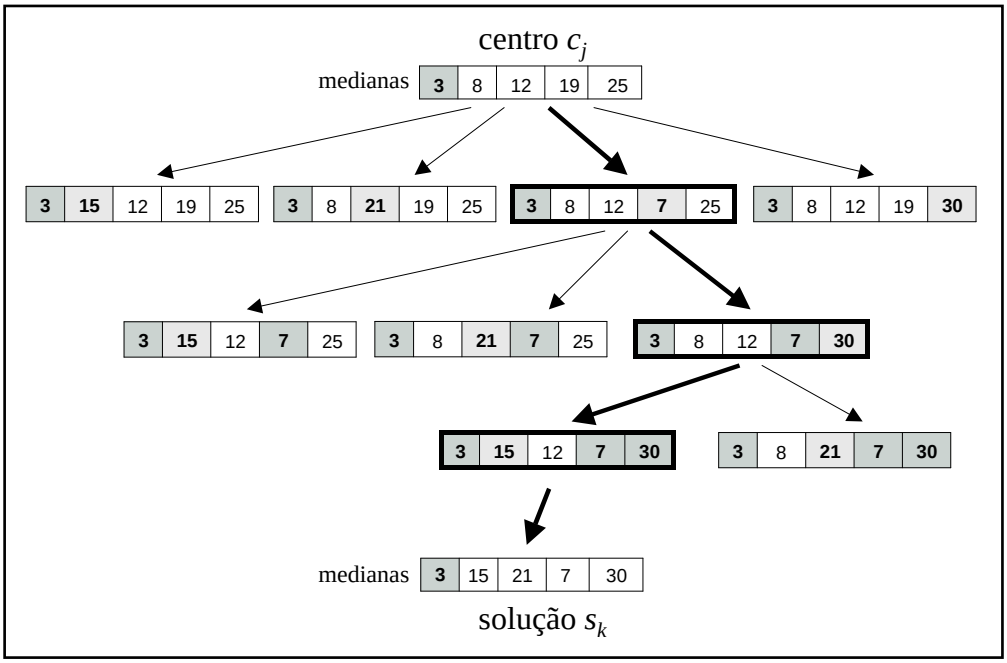

Figura 4.4 - Assimilação por caminho aplicada ao CPMP.

O volume do cluster, v_j, é verificado após o centro c_j ser atualizado. Um cluster pode ser considerado promissor quando atinge certo limitante λ. Neste trabalho foi

utilizado $\lambda = 15$. O valor de λ tem influência direta sobre o número de vezes que a busca local é realizada, quanto maior o valor de λ menor será o número de chamadas ao método de busca local.

Quando v_j atingir λ e a localização do cluster não for uma boa região de busca (índice de ineficácia $r_j \geq 5$), aplica-se uma perturbação no centro c_j. Essa perturbação é realizada trocando a alocação de 30% dos pontos de demanda aleatoriamente. Por outro lado, se $v_j \geq \lambda$ e $r_j < 5$, a heurística de busca local é aplicada no centro c_j.

4.8.3 Heurísticas de Busca Local

A busca local é acionada sempre que um cluster for considerado promissor, intensificando a busca na região do centro c_j. Neste trabalho foi implementado o método de Descida em Vizinhança Variável (VND) (ver Seção 3.5), no qual são utilizadas duas heurísticas de descida propostas por Lorena e Senne (2003): Localização-Alocação (*Location-Allocation Heuristic*) e Troca e Transferência (*Interchange-Transfer Heuristic*). Se uma solução melhor for encontrada retorna-se para a primeira heurística e continua-se a busca a partir da nova solução. O VND se encerra quando nenhuma melhora na solução corrente puder ser obtida por meio das heurísticas.

A heurística Localização-Alocação começa a busca a partir do centro c_j. A solução pode ser melhorada pesquisando uma nova mediana dentro de cada agrupamento, trocando a mediana corrente por um ponto alocado à ela e realocando os pontos de demanda. Todas as trocas possíveis são analisadas e, sempre que uma troca fornecer uma solução com melhor valor de função objetivo, esta é realizada e continua-se a busca a partir da nova solução. A Figura 4.5 apresenta o pseudocódigo da heurística Localização-Alocação.

Após trocar uma mediana j por um ponto de demanda i é preciso realocar os pontos de demanda. O método Realocar utilizado neste trabalho consiste de três passos. Primeiramente é preciso desalocar todos os pontos atendidos pela mediana j. Em seguida, cada ponto de demanda desalocado é atribuído à mediana mais próxima que não exceda a capacidade. Para os pontos pertencentes às outras medianas é verificado se há economia em transferir este ponto para a nova mediana. A Figura 4.6 apresenta o pseudocódigo do método Realocar, sendo p_k a mediana que atende o ponto de demanda k.

```
algoritmo Localizacao-Alocacao ( s )
    para ( j  de 1 até p ) faça
        para ( i de 1 até n) faça
            s' ← s
            se ( ponto i é atendido pela mediana j ) então
                s' ← trocar a mediana j pelo ponto i
                realoque os pontos de s'
                se ( f (s') < f (s) ) então
                    s ← s'
            fim-se
        fim-para
    fim-para
fim-algoritmo
```

Figura 4.5 - Pseudocódigo da heurística Localização-Alocação.

Fonte: Adaptado de Lorena e Senne (2003).

```
algoritmo Realocar ( s, j, i )
    para ( k de 1 até n ) faça
        se ( ponto k é atendido pela mediana j ) então
            desaloque k
    fim-para
    para ( k de 1 até n ) faça
        escolha um ponto b aleatoriamente
        se ( ponto b estiver desalocado ) então
            aloque b na mediana mais próxima que não exceda a capacidade
        senão
            se ( t_{bp_b} − t_{bi} > 0 ) então
                transfira o ponto b para a mediana i, caso não exceda a capacidade Q_i
    fim-para
fim-algoritmo
```

Figura 4.6 - Pseudocódigo do método Realocar.

A heurística Troca e Transferência consiste em dois movimentos: trocar a alocação de dois pontos de demanda atribuídos à medianas diferentes, e transferir um ponto de demanda de uma mediana para outra. Todos os movimentos possíveis são analisados, e sempre que houver um movimento que melhore a solução corrente este é realizado, continuando a busca a partir da nova solução. A Figura 4.7 apresenta o pseudocódigo dessa heurística, sendo D_j a demanda total atendida por cada mediana j.

```
algoritmo Troca e Transferencia ( s )
    para ( j de 1 até p ) faça
        para ( i de 1 até p, i ≠ j ) faça
            para ( cada ponto k da mediana j e l da mediana i ) faça
                se ( t_kj − t_ki + t_li − t_lj < 0  e
                     D_j − q_k + q_l ≤ Q_j  e
                     D_i − q_l + q_k ≤ Q_i ) então
                        troque k com l
            fim-para
            para ( cada ponto k da mediana i ) faça
                se ( t_ki − t_kj < 0  e
                     D_j + q_k ≤ Q_j ) então
                        transfira k da mediana i para a mediana j
            fim-para
        fim-para
    fim-para
fim-algoritmo
```

Figura 4.7 - Pseudocódigo da heurística Troca e Transferência.

Fonte: Adaptado de Lorena e Senne (2003).

4.8.4 Resultados Computacionais

O CS para o CPMP foi codificado em C++ e os testes computacionais executados em um PC com processador Pentium 4, 2.6 GHz e memória de 1 GB. Dois conjuntos de instâncias foram usados nos testes: um conjunto clássico introduzido por Osman e Christofides (1994), que contém 20 instâncias nomeadas *p1* a *p20*, e um conjunto de instâncias com dados reais coletados na cidade de São José dos

Campos proposto por Lorena e Senne (2003), que contém 6 instâncias nomeadas *sjc*. Essas instâncias podem ser encontradas no site da *OR-Library*. A Tabela 4.1 apresenta as características e o valor da solução ótima dessas instâncias. Boccia et al. (2008) provaram as soluções ótimas para essas instâncias, e para a instância *sjc4b* mostraram que o ótimo ainda não era conhecido.

Tabela 4.1 - CPMP: Características das instâncias testadas.

instância	n	p	Q	**solução ótima**
p1	50	5	120	713
p2	50	5	120	740
p3	50	5	120	751
p4	50	5	120	651
p5	50	5	120	664
p6	50	5	120	778
p7	50	5	120	787
p8	50	5	120	820
p9	50	5	120	715
p10	50	5	120	829
p11	100	10	120	1006
p12	100	10	120	966
p13	100	10	120	1026
p14	100	10	120	982
p15	100	10	120	1091
p16	100	10	120	954
p17	100	10	120	1034
p18	100	10	120	1043
p19	100	10	120	1031
p20	100	10	120	1005
sjc1	100	10	720	17288,99
sjc2	200	15	840	33270,94
sjc3a	300	25	740	45335,16
sjc3b	300	30	740	40635,90
sjc4a	402	30	840	61925,51
sjc4b	402	40	840	52458,00

As Tabelas 4.2 - 4.6 apresentam os resultados do CS aplicado ao CPMP, utilizando diferentes métodos geradores de soluções. Cada tabela mostra os resultados do CS e do gerador de soluções sem o processo de agrupamento. Os valores em negrito indicam as instâncias nas quais foi encontrada a solução ótima. As colunas das tabelas representam:

- **ótimo**: solução ótima global da instância;
- **sol***: melhor solução encontrada pelo método;
- **desvio**: erro relativo entre a média das soluções ($sol_{média}$) e a melhor solução encontrada pelo método (sol^*)(ver Equação 4.7);
- **gap**: erro relativo entre a melhor solução encontrada pelo método (sol^*) e a melhor solução conhecida (sol_{melhor}) (ver Equação 4.8);
- **T***: tempo médio gasto pelo método para encontrar a melhor solução (em segundos);
- **T**: tempo médio de execução do método (em segundos);
- **delta**: erro relativo entre a melhor solução encontrada pelo CS (sol^*_{CS}) e a melhor solução encontrada pelo método gerador de soluções (sol^*_{MH}) (ver Equação 4.9).

$$\text{desvio} = \left(\frac{sol_{média} - sol^*}{sol^*} \right) \times 100 \tag{4.7}$$

$$\text{gap} = \left(\frac{sol^* - sol_{melhor}}{sol_{melhor}} \right) \times 100 \tag{4.8}$$

$$\text{delta} = \left(\frac{sol^*_{MH} - sol^*_{CS}}{sol^*_{CS}} \right) \times 100 \tag{4.9}$$

Para avaliar a robustez do CS, cada instância foi executada 20 vezes. Assim, é possível ter uma noção do comportamento geral do método, apesar de não poder afirmar que esse comportamento sempre acontecerá.

Observando as Tabelas 4.2 - 4.6 nota-se que o CS encontrou a solução ótima em pelo ao menos 24 das 26 instâncias independentemente do método utilizado para gerar soluções. Para a instância *sjc4a* apenas o CS com o GA e com o SA como geradores de soluções foi capaz de encontrar a solução ótima. Para a instância *sjc4b*, apesar do CS não ter encontrado a solução ótima, foram obtidas soluções muito próximas

à solução ótima. Por meio do desvio é possível perceber que o CS se mostrou um método robusto, uma vez que, todas as versões do CS apresentaram desvios próximos a zero.

Os tempos de execução do CS foram competitivos, e o tempo gasto para encontrar a melhor solução é aproximadamente 50% do tempo total de execução. À exceção do CS que utiliza o método gerador de soluções aleatórias, no qual o tempo gasto para obter a melhor solução é cerca de 75% do tempo total. Esses dados mostram que a utilização de uma meta-heurística como gerador de soluções possibilita uma melhor convergência do CS, uma vez que, estas levam em consideração informações sobre o espaço de busca do problema.

Outra observação que pode ser feita sobre os dados dessas tabelas diz respeito à melhora produzida pela utilização do processo de agrupamento. Por meio da coluna *delta*, pode-se notar que o CS obtém uma melhora significativa, principalmente para as instâncias *sjc*, em relação às soluções encontradas pelas meta-heurísticas.

Entre as meta-heurísticas implementadas, a que obteve os melhores resultados sem o processo de agrupamento foi o SA, enquanto, GA e ILS obtiveram os piores resultados. Entretanto, as versões do CS utilizando GA e ILS como geradores de soluções, além da versão do CS com método aleatório, encontraram soluções muito boas neste trabalho. Essa questão mostra que o CS não é dependente da qualidade da solução gerada para o processo de agrupamento, sendo mais importante que o método gerador de soluções seja capaz de gerar uma grande quantidade de soluções diferentes.

Na Tabela 4.7 são apresentados os resultados de três abordagens para o CPMP que possuem as melhores performances entre os trabalhos encontrados na literatura. A primeira é uma meta-heurística híbrida combinando *Scatter Search* com Recombinação por Caminho (PR) (SCHEUERER; WENDOLSKY, 2006), a segunda abordagem consiste na meta-heurística VNS combinada com o CPLEX (FLESZAR; HINDI, 2008), e a terceira abordagem apresenta um algoritmo com planos de corte (BOCCIA et al., 2008). Os resultados do CS são competitivos se comparados com as duas primeiras abordagens, tanto em termos de qualidade da solução quanto em tempo computacional. Entretanto, Boccia et al. (2008) conseguiram uma formulação matemática muito eficiente, na qual o CPLEX precisa de alguns minutos para obter a solução ótima para todas as instâncias.

Tabela 4.2 - CPMP: Resultados Computacionais do CS-GA

		CS-GA					GA				
	ótimo	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
p1	713	**713**	0,01	0,00	1,11	3,57	**713**	1,14	0,00	1,29	0,00
p2	740	**740**	0,00	0,00	0,70	3,48	**740**	0,00	0,00	1,29	0,00
p3	751	**751**	0,00	0,00	1,21	4,59	765	0,00	1,86	1,33	1,86
p4	651	**651**	0,00	0,00	0,79	3,84	652	0,00	0,15	1,34	0,15
p5	664	**664**	0,00	0,00	0,71	4,60	**664**	0,00	0,00	1,30	0,00
p6	778	**778**	0,00	0,00	0,73	4,86	787	0,00	1,16	1,38	1,16
p7	787	**787**	0,00	0,00	0,84	5,43	826	2,82	4,96	1,42	4,96
p8	820	**820**	0,00	0,00	1,84	5,70	831	0,45	1,34	1,43	1,34
p9	715	**715**	0,00	0,00	0,76	4,94	716	0,28	0,14	1,36	0,14
p10	829	**829**	0,29	0,00	1,52	6,23	867	1,51	4,58	1,45	4,58
p11	1006	**1006**	0,00	0,00	6,42	23,32	**1006**	0,36	0,00	3,04	0,00
p12	966	**966**	0,00	0,00	1,23	20,08	970	0,30	0,41	2,63	0,41
p13	1026	**1026**	0,00	0,00	0,80	15,91	**1026**	0,00	0,00	2,61	0,00
p14	982	**982**	0,17	0,00	8,53	19,85	991	0,00	0,92	2,67	0,92
p15	1091	**1091**	0,14	0,00	10,71	24,43	1096	0,45	0,46	2,77	0,46
p16	954	**954**	0,00	0,00	5,62	19,39	957	0,26	0,31	2,69	0,31
p17	1034	**1034**	0,00	0,00	4,70	19,79	**1034**	0,21	0,00	2,81	0,00
p18	1043	**1043**	0,58	0,00	16,48	37,92	1077	0,55	3,26	3,54	3,26
p19	1031	**1031**	0,00	0,00	9,86	27,21	1043	0,06	1,16	3,09	1,16
p20	1005	**1005**	0,28	0,00	7,95	29,05	1024	0,53	1,89	3,03	1,89
sjc1	17288,99	**17288,99**	0,24	0,00	7,21	19,25	17426,41	0,30	0,79	2,63	0,79
sjc2	33270,94	**33270,94**	0,03	0,00	27,90	75,66	33360,08	0,04	0,27	5,90	0,27
sjc3a	45335,16	**45335,16**	0,01	0,00	640,30	1339,75	47894,89	1,30	5,65	68,93	5,65
sjc3b	40635,90	**40635,90**	0,02	0,00	1524,59	2704,10	43431,25	1,25	6,88	134,92	6,88
sjc4a	61925,51	**61925,51**	0,07	0,00	2793,71	4441,70	68263,07	1,15	10,23	223,33	10,23
sjc4b	52458,00	52468,87	0,04	0,02	3964,17	7539,40	58424,84	0,37	11,37	573,69	11,35
média			**0,07**	**0,001**	**347,71**	**630,92**		**0,51**	**2,22**	**40,46**	**2,22**

Tabela 4.3 - CPMP: Resultados Computacionais do CS-SA

		CS-SA					SA				
	ótimo	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
p1	713	**713**	0,00	0,00	0,98	7,97	**713**	0,00	0,00	5,79	0,00
p2	740	**740**	0,00	0,00	0,80	8,15	**740**	0,00	0,00	5,89	0,00
p3	751	**751**	0,00	0,00	0,94	8,01	**751**	0,00	0,00	5,93	0,00
p4	651	**651**	0,00	0,00	0,84	7,88	**651**	0,00	0,00	5,83	0,00
p5	664	**664**	0,00	0,00	1,11	8,06	**664**	0,00	0,00	5,83	0,00
p6	778	**778**	0,00	0,00	0,83	8,07	**778**	0,00	0,00	5,90	0,00
p7	787	**787**	0,00	0,00	1,46	8,13	**787**	0,05	0,00	5,92	0,00
p8	820	**820**	0,00	0,00	1,31	8,14	**820**	0,11	0,00	5,86	0,00
p9	715	**715**	0,00	0,00	1,02	8,18	**715**	0,06	0,00	5,92	0,00
p10	829	**829**	0,12	0,00	4,75	8,25	**829**	0,12	0,00	5,95	0,00
p11	1006	**1006**	0,04	0,00	9,10	33,96	**1006**	0,75	0,00	12,05	0,00
p12	966	**966**	0,00	0,00	1,90	34,86	**966**	0,28	0,00	11,87	0,00
p13	1026	**1026**	0,00	0,00	1,43	36,27	**1026**	0,06	0,00	12,17	0,00
p14	982	**982**	0,11	0,00	7,64	34,51	985	0,74	0,31	12,08	0,31
p15	1091	**1091**	0,00	0,00	12,39	33,97	1093	0,90	0,18	12,01	0,18
p16	954	**954**	0,00	0,00	5,45	34,60	**954**	0,38	0,00	12,18	0,00
p17	1034	**1034**	0,08	0,00	11,62	33,64	**1034**	1,49	0,00	12,04	0,00
p18	1043	**1043**	0,15	0,00	10,15	34,07	1043	0,74	0,00	12,04	0,00
p19	1031	**1031**	0,10	0,00	15,34	34,02	1034	0,82	0,29	12,10	0,29
p20	1005	**1005**	0,29	0,00	18,33	34,48	1008	1,16	0,30	12,08	0,30
sjc1	17288,99	**17288,99**	0,00	0,00	11,47	24,65	**17288,99**	1,10	0,00	11,78	0,00
sjc2	33270,94	**33270,94**	0,05	0,00	45,78	102,49	33388,52	0,54	0,35	25,91	0,35
sjc3a	45335,16	**45335,16**	0,02	0,00	410,64	668,38	46011,60	1,56	1,49	53,06	1,49
sjc3b	40635,90	**40635,90**	0,03	0,00	662,60	1087,36	41514,86	1,19	2,16	61,11	2,16
sjc4a	61925,51	**61925,51**	0,12	0,00	1003,76	1758,19	63826,17	1,69	3,07	81,36	3,07
sjc4b	52458,00	52472,40	0,13	0,03	2829,96	4722,04	54012,21	1,18	2,96	102,02	2,93
média			**0,05**	**0,001**	**195,06**	**338,01**		**0,57**	**0,43**	**19,80**	**0,42**

Tabela 4.4 - CPMP: Resultados Computacionais do CS-VNS

		CS-VNS					VNS				
	ótimo	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
p1	713	**713**	0,00	0,00	0,95	5,44	**713**	0,32	0,00	4,06	0,00
p2	740	**740**	0,00	0,00	0,72	5,43	**740**	0,00	0,00	4,13	0,00
p3	751	**751**	0,00	0,00	1,19	5,43	**751**	0,13	0,00	3,95	0,00
p4	651	**651**	0,00	0,00	0,74	5,43	**651**	0,05	0,00	4,05	0,00
p5	664	**664**	0,00	0,00	1,40	5,43	**664**	0,24	0,00	3,92	0,00
p6	778	**778**	0,00	0,00	0,79	5,43	**778**	0,31	0,00	3,76	0,00
p7	787	**787**	0,00	0,00	1,11	5,43	**787**	0,28	0,00	3,77	0,00
p8	820	**820**	0,00	0,00	1,49	5,43	822	1,00	0,24	3,67	0,24
p9	715	**715**	0,00	0,00	0,92	5,43	**715**	0,66	0,00	3,74	0,00
p10	829	**829**	0,33	0,00	1,35	6,48	837	2,51	0,97	4,53	0,97
p11	1006	**1006**	0,17	0,00	3,21	12,07	1009	1,07	0,30	3,85	0,30
p12	966	**966**	0,00	0,00	1,54	12,46	967	0,82	0,10	3,93	0,10
p13	1026	**1026**	0,00	0,00	1,10	12,57	**1026**	0,95	0,00	3,79	0,00
p14	982	**982**	0,30	0,00	6,56	12,93	990	0,92	0,81	3,92	0,81
p15	1091	**1091**	0,05	0,00	5,99	13,67	1096	2,28	0,46	3,97	0,46
p16	954	**954**	0,02	0,00	5,43	13,04	955	0,69	0,10	3,94	0,10
p17	1034	**1034**	0,27	0,00	7,27	12,55	1043	2,08	0,87	3,98	0,87
p18	1043	**1043**	0,42	0,00	7,31	13,15	1049	0,99	0,58	4,02	0,58
p19	1031	**1031**	0,32	0,00	7,22	14,68	1032	1,65	0,10	4,22	0,10
p20	1005	**1005**	0,78	0,00	6,05	14,88	1025	1,91	1,99	3,70	1,99
sjc1	17288,99	**17288,99**	0,15	0,00	4,56	14,87	17470,79	1,59	1,05	3,97	1,05
sjc2	33270,94	**33270,94**	0,14	0,00	34,65	76,63	33656,97	1,71	1,16	10,30	1,16
sjc3a	45335,16	**45335,16**	0,21	0,00	157,33	511,35	47201,76	1,65	4,12	20,33	4,12
sjc3b	40635,90	**40635,90**	0,34	0,00	305,33	736,25	42495,69	0,88	4,58	17,95	4,58
sjc4a	61925,51	62000,63	0,38	0,12	737,24	1343,40	65115,14	1,34	5,15	33,09	5,02
sjc4b	52458,00	52480,85	0,31	0,04	977,68	3440,68	54833,10	1,47	4,53	30,30	4,48
média			**0,16**	**0,01**	**87,66**	**242,71**		**1,06**	**1,04**	**7,49**	**1,04**

Tabela 4.5 - CPMP: Resultados Computacionais do CS-ILS

		CS-ILS					ILS				
	ótimo	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
p1	713	**713**	0,00	0,00	0,89	3,53	722	1,51	1,26	1,52	1,26
p2	740	**740**	0,00	0,00	0,72	3,80	741	1,00	0,14	1,50	0,14
p3	751	**751**	0,00	0,00	0,81	3,53	757	0,63	0,80	1,55	0,80
p4	651	**651**	0,00	0,00	0,73	3,46	652	0,51	0,15	1,56	0,15
p5	664	**664**	0,00	0,00	1,03	3,65	669	1,46	0,75	1,59	0,75
p6	778	**778**	0,00	0,00	0,74	3,74	784	0,92	0,77	1,54	0,77
p7	787	**787**	0,00	0,00	1,16	3,74	791	1,50	0,51	1,53	0,51
p8	820	**820**	0,00	0,00	1,03	3,64	824	1,08	0,49	1,52	0,49
p9	715	**715**	0,00	0,00	0,85	3,67	717	0,81	0,28	1,54	0,28
p10	829	**829**	0,33	0,00	1,52	5,42	851	1,02	2,65	3,28	2,65
p11	1006	**1006**	0,07	0,00	3,62	23,89	1057	2,40	5,07	3,26	5,07
p12	966	**966**	0,00	0,00	1,52	25,55	991	2,05	2,59	3,25	2,59
p13	1026	**1026**	0,00	0,00	1,13	26,26	1084	2,06	5,65	3,20	5,65
p14	982	**982**	0,08	0,00	4,58	24,46	1039	1,28	5,80	3,61	5,80
p15	1091	**1091**	0,01	0,00	7,29	24,41	1156	1,44	5,96	3,22	5,96
p16	954	**954**	0,00	0,00	2,44	25,27	988	2,26	3,56	3,63	3,56
p17	1034	**1034**	0,11	0,00	9,34	23,90	1089	1,33	5,32	3,73	5,32
p18	1043	**1043**	0,15	0,00	5,51	24,40	1080	1,91	3,55	3,61	3,55
p19	1031	**1031**	0,16	0,00	8,78	24,40	1079	1,44	4,66	3,47	4,66
p20	1005	**1005**	0,21	0,00	18,12	24,41	1081	1,52	7,56	3,67	7,56
sjc1	17288,99	**17288,99**	0,00	0,00	8,93	23,69	17759,05	1,08	2,72	3,68	2,72
sjc2	33270,94	**33270,94**	0,00	0,00	48,35	154,21	34797,26	1,18	4,59	8,84	4,59
sjc3a	45335,16	**45335,16**	0,01	0,00	660,99	1351,55	48421,43	1,00	6,81	24,34	6,81
sjc3b	40635,90	**40635,90**	0,00	0,00	1149,80	2227,91	43617,34	0,88	7,34	22,64	7,34
sjc4a	61925,51	61967,46	0,11	0,07	1457,62	3461,51	66842,64	1,33	7,94	42,36	7,87
sjc4b	52458,00	52468,87	0,07	0,02	5109,49	8261,45	56905,72	1,13	8,48	41,86	8,46
média			**0,05**	**0,003**	**327,19**	**606,36**		**1,34**	**3,67**	**7,52**	**3,67**

Tabela 4.6 - CPMP: Resultados Computacionais do CS-Aleatório

		CS-Aleatório					Aleatório				
	ótimo	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
p1	713	**713**	0,00	0,00	0,93	6,15	731	3,42	2,52	3,93	2,52
p2	740	**740**	0,00	0,00	0,79	6,15	751	2,62	1,49	3,81	1,49
p3	751	**751**	0,00	0,00	0,94	6,15	778	3,23	3,60	4,02	3,60
p4	651	**651**	0,00	0,00	0,82	6,15	658	3,22	1,08	4,09	1,08
p5	664	**664**	0,00	0,00	1,21	6,15	685	3,96	3,16	3,91	3,16
p6	778	**778**	0,00	0,00	0,79	5,61	794	3,32	2,06	3,26	2,06
p7	787	**787**	0,00	0,00	1,35	6,16	819	2,38	4,07	3,83	4,07
p8	820	**820**	0,00	0,00	1,42	6,17	852	2,38	3,90	3,98	3,90
p9	715	**715**	0,00	0,00	1,01	6,15	735	2,22	2,80	3,91	2,80
p10	829	**829**	0,29	0,00	1,47	6,16	852	3,70	2,77	3,81	2,77
p11	1006	**1006**	0,02	0,00	9,82	26,00	1174	3,32	16,70	2,24	16,70
p12	966	**966**	0,00	0,00	1,42	26,65	1100	2,61	13,87	1,95	13,87
p13	1026	**1026**	0,00	0,00	1,18	27,57	1171	3,42	14,13	1,78	14,13
p14	982	**982**	0,10	0,00	5,62	25,70	1160	1,89	18,13	2,24	18,13
p15	1091	**1091**	0,01	0,00	9,13	25,54	1272	2,68	16,59	1,78	16,59
p16	954	**954**	0,00	0,00	3,66	26,05	1062	5,18	11,32	1,80	11,32
p17	1034	**1034**	0,15	0,00	10,14	24,80	1163	4,87	12,48	1,73	12,48
p18	1043	**1043**	0,14	0,00	5,44	25,69	1184	3,59	13,52	2,33	13,52
p19	1031	**1031**	0,10	0,00	11,92	25,74	1174	3,25	13,87	2,26	13,87
p20	1005	**1005**	0,31	0,00	10,25	25,57	1195	4,11	18,91	2,25	18,91
sjc1	17288,99	**17288,99**	0,00	0,00	6,55	26,06	19665,47	1,50	13,75	2,23	13,75
sjc2	33270,94	**33270,94**	0,00	0,00	81,98	174,77	38313,28	1,61	15,16	3,60	15,16
sjc3a	45335,16	**45335,16**	0,01	0,00	896,63	1547,12	53809,21	2,72	18,69	6,48	18,69
sjc3b	40635,90	**40635,90**	0,00	0,00	1988,20	2427,89	49007,25	1,43	20,60	6,99	20,60
sjc4a	61925,51	61928,91	0,09	0,01	3034,01	4259,75	76015,64	1,95	22,75	8,71	22,63
sjc4b	52458,00	52472,40	0,05	0,03	7152,45	9024,07	64155,66	1,07	22,30	10,65	22,75
média			**0,05**	**0,001**	**509,20**	**683,84**		**2,91**	**11,16**	**3,75**	**11,16**

Tabela 4.7 - CPMP: Melhores Resultados Computacionais da Literatura

		Scheuerer e Wendolsky (2006)			Fleszar e Hindi (2008)			Boccia et al. (2008)		
	ótimo	**sol***	**gap**	**T**	**sol***	**gap**	**T**	**sol***	**gap**	**T**
p1	713	**713**	0,00	6,00	**713**	0,00	0,17	**713**	0,00	3,20
p2	740	**740**	0,00	6,00	**740**	0,00	0,05	**740**	0,00	0,20
p3	751	**751**	0,00	6,00	**751**	0,00	0,19	**751**	0,00	1,00
p4	651	**651**	0,00	6,00	**651**	0,00	0,11	**651**	0,00	0,40
p5	664	**664**	0,00	6,00	**664**	0,00	0,27	**664**	0,00	0,50
p6	778	**778**	0,00	6,00	**778**	0,00	0,11	**778**	0,00	0,50
p7	787	**787**	0,00	6,00	**787**	0,00	0,31	**787**	0,00	9,40
p8	820	**820**	0,00	6,00	**820**	0,00	0,92	**820**	0,00	94,60
p9	715	**715**	0,00	6,00	**715**	0,00	0,13	**715**	0,00	5,00
p10	829	**829**	0,00	6,00	**829**	0,00	0,75	**829**	0,00	19,60
p11	1006	**1006**	0,00	60,00	**1006**	0,00	7,91	**1006**	0,00	29,20
p12	966	**966**	0,00	60,00	**966**	0,00	4,81	**966**	0,00	18,60
p13	1026	**1026**	0,00	60,00	**1026**	0,00	2,17	**1026**	0,00	9,10
p14	982	**982**	0,00	60,00	**982**	0,00	10,33	**982**	0,00	196,20
p15	1091	**1091**	0,00	60,00	**1091**	0,00	10,23	**1091**	0,00	77,70
p16	954	**954**	0,00	60,00	**954**	0,00	4,20	**954**	0,00	25,00
p17	1034	**1034**	0,00	60,00	**1034**	0,00	5,50	**1034**	0,00	41,20
p18	1043	**1043**	0,00	60,00	**1043**	0,00	9,06	**1043**	0,00	40,30
p19	1031	**1031**	0,00	60,00	**1031**	0,00	8,64	**1031**	0,00	42,60
p20	1005	**1005**	0,00	60,00	**1005**	0,00	27,34	**1005**	0,00	3259,10
sjc1	17288,99	**17288,99**	0,00	60,00	**17288,99**	0,00	50,50	**17288,99**	0,00	37,60
sjc2	33270,94	33293,40	0,07	600,00	**33270,94**	0,00	44,08	**33270,94**	0,00	127,90
sjc3a	45335,16	45338,02	0,01	2307,00	**45335,16**	0,00	8580,30	**45335,16**	0,00	495,10
sjc3b	40635,90	**40635,90**	0,00	2308,00	**40635,90**	0,00	2292,86	**40635,90**	0,00	72,20
sjc4a	61925,51	**61925,52**	0,00	6109,00	**61925,51**	0,00	4221,47	**61925,51**	0,00	1209,50
sjc4b	52458,00	52531,46	0,14	6106,00	52469,96	0,02	3471,44	**52458,00**	0,00	669,70
média			**0,01**	**698,08**		**0,001**	**721,30**		**0,00**	**249,44**

4.9 Problema de Agrupamento Centrado Capacitado

O Problema de Agrupamento Centrado Capacitado (CCCP) é uma generalização do CPMP, que consiste em definir um conjunto de p agrupamentos com capacidade limitada e mínima dissimilaridade entre os agrupamentos formados. Cada agrupamento possui um centróide localizado no centro geométrico de seus pontos e cobre todas as demandas de um conjunto de n pontos. A principal diferença entre o CCCP e o CPMP é que no CCCP os agrupamentos são centrados na média das coordenadas dos seus pontos, enquanto, no CPMP os agrupamentos são centrados por suas medianas. A Figura 4.8 ilustra um exemplo de uma solução para esses dois problemas.

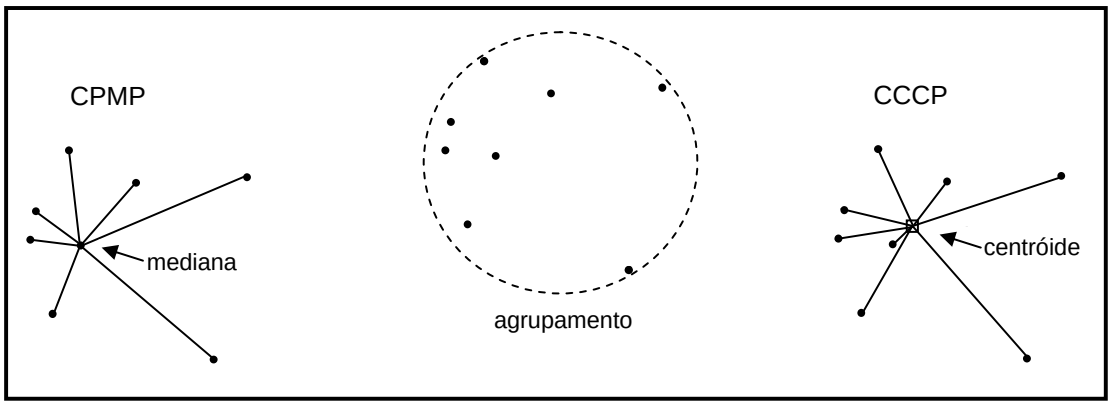

Figura 4.8 - Exemplo de uma solução dos problemas CPMP e CCCP.

Fonte: Adaptado de Pereira e Senne (2008).

A abordagem do CS para o CCCP é a mesma aplicada ao CPMP. A única diferença está no cálculo da função objetivo, no qual é preciso definir quais são os centróides antes de calcular a distância entre estes e os pontos de demanda alocados ao agrupamento.

4.9.1 Formulação Matemática

O modelo matemático do CCCP, proposto por Negreiros e Palhano (2006), é apresentado a seguir.

$$(p-CCCP) \quad \min \quad \sum_{i\in I}\sum_{j\in J} \|a_i - \bar{x}_j\|^2 y_{ij} \tag{4.10}$$

sujeito a:

$$\sum_{j\in J} y_{ij} = 1 \qquad \forall i \in I \tag{4.11}$$

$$\sum_{i\in I} y_{ij} = n_j \qquad \forall j \in J \tag{4.12}$$

$$\sum_{i\in I} a_i y_{ij} \leq n_j \bar{x}_j \qquad \forall j \in J \tag{4.13}$$

$$\sum_{i\in I} q_i y_{ij} \leq Q_j \qquad \forall j \in J \tag{4.14}$$

$$\bar{x}_j \in \Re^l,\ n_j \in N,\ y_{ij} \in \{0,\ 1\} \qquad \forall i \in I, \forall j \in J \tag{4.15}$$

no qual,

- I é o conjunto de pontos de demanda;
- J é o conjunto de agrupamentos, com $|J| = p$;
- a_i é a posição geométrica do ponto i;
- $\bar{x_j}$ é a posição geométrica do centróide do agrupamento j;
- $y_{ij} = \begin{cases} 1, & \text{se o ponto } i \text{ é alocado ao agrupamento } j, \\ 0, & \text{caso contrário.} \end{cases}$
- n_j é o número de pontos no cluster j;
- q_i é a demanda do ponto i;
- Q_j é a capacidade de cada agrupamento j.

Nesse modelo, a função objetivo 4.10 minimiza a distância total entre cada ponto e o centróide do agrupamento ao qual está alocado. Observe que a posição geométrica do centróide depende dos pontos que compõem o agrupamento, sendo assim, a posição do centróide não é um parâmetro conhecido a *priori*. As restrições 4.11 impõem que cada ponto seja alocado para exatamente um agrupamento. As restrições 4.12 fornecem o número de pontos em cada agrupamento. As restrições 4.13 localizam o centróide de cada agrupamento no seu centro geométrico. As restrições 4.14 impõem que a capacidade total do agrupamento deve ser respeitada. As restrições 4.15 definem as variáveis de decisão e o número máximo de pontos por agrupamento.

O CCCP é classificado como NP-*hard*. Além disso, a não linearidade da função objetivo dificulta o emprego de métodos exatos para resolução do CCCP. Por este motivo, esse problema é geralmente resolvido por meio de abordagens heurísticas.

4.9.2 Revisão Bibliográfica

O CCCP foi recentemente introduzido por Negreiros e Palhano (2006) e ainda não foi exaustivamente estudado como outros problemas clássicos de localização. Os autores apresentam um algoritmo heurístico de duas fases para o CCCP. A primeira fase utiliza o algoritmo de Forgy (FORGY, 1965) para construir uma solução inicial, e a segunda fase é um refinamento por meio da meta-heurística VNS. Uma aplicação prática do CCCP voltada para o desenvolvimento de zonas de coleta de lixo na cidade de Fortaleza também é reportada.

Palhano et al. (2008) apresentam um método construtivo polinomial chamado *Constrained KMedians*. Outros dois métodos são propostos para distribuir as cargas entre os agrupamentos, melhorando a exploração da vizinhança de cada agrupamento. Esses métodos foram aplicados para definir áreas de cobertura para o controle da doença da dengue por agentes sanitários em Fortaleza.

Pereira e Senne (2008) adaptaram um método Geração de Colunas, originalmente proposto para o CPMP, para resolver instâncias do CCCP. Esse algoritmo utiliza a relaxação lagrangiana/surrogate para estabilizar o processo de geração de colunas.

Além do CPMP, outro problema similar ao CCCP é o Problema de Localização-Alocação Contínuo Capacitado (*capacitated continuous location-allocation problem*) (BRIMBERG et al., 2008), também referenciado como *capacitated multisource Weber problem* (WESOLOWSKY, 1993). Esse problema requer a geração de um dado número

m de facilidades capacitadas no espaço contínuo e a alocação de n clientes ou pontos fixos para cada uma. As facilidades podem estar localizadas em qualquer ponto do espaço contínuo. O objetivo é satisfazer a demanda dos clientes e minimizar o custo total de transporte (ou serviço).

4.9.3 CS Aplicado ao CCCP

Como mencionado anteriormente, a única diferença entre o CPMP e o CCCP está no cálculo da função objetivo. Sendo assim, é possível utilizar a mesma abordagem do CS aplicada ao CPMP para resolver de forma aproximada o CCCP.

No CCCP não é possível predefinir uma matriz de distâncias, já que a posição geométrica do centróide depende dos pontos que compõem o agrupamento. Sendo assim, no cálculo da função objetivo do CCCP é preciso primeiramente determinar a localização dos centróides, calculando a média das coordenadas dos pontos de demanda atribuídos a cada agrupamento. E então, calcula-se a distância euclidiana entre os centróides e os pontos de demanda alocados a cada agrupamento.

Essa característica faz com que a avaliação da função objetivo tenha um custo computacional muito alto. Em razão disso, é inviável realizar um número grande de chamadas a esta função objetivo. Nesta abordagem do CS optou-se por aplicar a função objetivo do CCCP apenas no componente de busca local do CS, e no demais componentes do CS é calculada a função objetivo do CPMP.

Uma solução do CCCP é representada da mesma maneira que uma solução do CPMP (ver Figura 4.1), a qual é representada por dois vetores de valores inteiros. O primeiro vetor representa a alocação dos pontos de demanda nos agrupamentos. O segundo vetor representa as medianas selecionadas. Neste caso, o segundo vetor só é utilizado pelos componentes que resolvem o CPMP.

As estruturas de vizinhança definidas na Seção 4.5 também são utilizadas nesta abordagem, inclusive pelas heurísticas Localização-Allocação e Troca e Transferência implementadas no método VND do componente de busca local. Porém, nessas heurísticas calcula-se a função objetivo do CCCP.

Desta forma, nota-se que a abordagem do CS aplicado ao CCCP reutiliza praticamente todo o código do CS aplicado ao CPMP. As únicas modificações são a inserção da nova função para calcular a função objetivo do CCCP e a chamada a

esta nova função nas heurísticas do componente de busca local.

4.9.4 Resultados Computacionais

Os testes computacionais foram executados em um PC com processador Pentium 4, 2.6 GHz e memória de 1 GB. Quatro conjuntos de problemas são usados nestes testes: dois conjuntos com dados coletados na cidade de São José dos Campos (LORENA; SENNE, 2003), que contêm seis instâncias *sjc* e cinco instâncias *p3038*; e dois conjuntos de instâncias introduzidos por Negreiros e Palhano (2006), que contêm sete instâncias nomeadas *ta* e sete instâncias nomeadas *doni*, as quais são baseadas em clientes de uma distribuidora de comida que opera na área metropolitana de Fortaleza. Essas instâncias podem ser encontradas no site da OR-Library e no site `http://www.lac.inpe.br/~lorena/instancias.html`. A Tabela 4.8 apresenta as características e a melhores soluções conhecidas dessas instâncias. As instâncias nas quais o CS conseguiu encontrar soluções melhores que as conhecidas na literatura até então foram marcadas nesta tabela com o número 1.

Nas instâncias *p3038* e *doni* são feitas algumas modificações nos parâmetros do CS para manter o tempo computacional competitivo. O número de soluções enviadas para o processo de agrupamento foi diminuído para 5000 soluções. Por causa do grande número de medianas das instâncias *p3038*, a assimilação do centro por meio do PR analisa somente 10% do caminho entre o centro do cluster e a solução agrupada. E o método Localização-Alocação é simplificado nas instâncias *doni*, pesquisando apenas 50% das trocas possíveis entre uma mediana e seus pontos de demanda alocados.

As Tabelas 4.9 - 4.13 apresentam os resultados do CS aplicado o CCCP. Essas tabelas possuem as mesmas colunas das tabelas de resultados apresentadas para o CPMP (ver Subseção 4.8.4). As colunas são sol^* (melhor solução encontrada pelo método), *desvio* (erro relativo entre a solução média e a melhor solução encontrada pelo método), *gap* (erro relativo entre a melhor solução encontrada pelo método e a melhor solução conhecida), T^* (tempo médio, em segundos, gasto pelo método para encontrar a melhor solução), T (tempo médio, em segundos, de execução do método) e *delta* (erro relativo entre a melhor solução do CS e a melhor solução do método gerador de soluções). A única diferença é que para este problema não se conhece o ótimo global das instâncias. Portanto, a coluna *ótimo* foi substituída pela coluna *melhor*, representando a melhor solução conhecida para a instância. Novamente,

cada instância foi executada 20 vezes para avaliar a robustez do CS. Os valores em negrito indicam as instâncias para as quais foi obtida a melhor solução conhecida.

Tabela 4.8 - CCCP: Características das instâncias testadas.

instância	n	p	Q	**melhor**
ta25	25	5	6	1251,44
ta50 [1]	50	5	11	4474,52
ta60	60	5	13	5356,58
ta70 [1]	70	5	17	6240,67
ta80	80	7	12	5515,46
ta90	90	4	23	8899,05
ta100 [1]	100	6	17	8102,04
sjc1 [1]	100	10	720	17359,75
sjc2 [1]	200	15	840	33181,65
sja3a [1]	300	25	740	45354,38
sjc3b [1]	300	30	740	40661,94
sjc4a [1]	402	30	840	61931,60
sjc4b [1]	402	40	840	52227,11
p3038-600 [1]	3038	600	321	128203,40
p3038-700 [1]	3038	700	273	116051,88
p3038-800 [1]	3038	800	238	106961,20
p3038-900 [1]	3038	900	216	99756,59
p3038-1000 [1]	3038	1000	191	92706,38
doni1	1000	6	200	3021,41
doni2	2000	6	400	6080,70
doni3 [1]	3000	8	400	8438,96
doni4 [1]	4000	10	400	10952,27
doni5 [1]	5000	12	450	11172,92
doni6 [1]	10000	23	450	15722,67
doni7 [1]	13221	30	450	18596,74

A Tabela 4.14 apresenta os resultados de três trabalhos encontrados na literatura sobre o CCCP. Negreiros e Palhano (2006) utilizam uma abordagem heurística resolvendo o CCCP por meio do VNS, Palhano et al. (2008) apresentam um método construtivo polinomial para o CCCP Pereira e Senne (2008) resolvem o CCCP por meio do método Geração de Colunas.

[1]Instâncias nas quais a melhor solução conhecida foi obtida pelo CS.

Observando os *gaps* e os tempos de execução do CS nas Tabelas 4.9 - 4.13, nota-se que todas as versões do CS tiveram um comportamento semelhante, tanto em relação à qualidade da solução como em tempo de execução. O CS obteve uma solução melhor que a solução conhecida na literatura em 19 das 25 instâncias testadas, em duas instâncias encontrou a mesma solução da literatura e obteve soluções piores em quatro instâncias. Novamente, o CS se mostrou um método robusto tendo desvios médios pequenos entre a melhor solução encontrada e a solução média.

Todos os CS's encontraram as mesmas soluções para todas as instâncias *ta* em todas as execuções ($desvio = 0$), tendo obtido a melhor solução conhecida em cinco instâncias. Para as instâncias *sjc*, o CS obteve soluções melhores que as conhecidas na literatura até o momento. Algumas versões do CS não encontraram a melhor solução em algumas instâncias *sjc*, porém, o *gap* obtido é relativamente pequeno.

Os resultados obtidos pelo CS para as instâncias *p3038* foram muito superiores aos apresentados em Negreiros e Palhano (2006), melhorando consideravelmente os valores de função objetivo dessas instâncias. Entretanto, os tempos computacionais do CS foram altos. Observa-se também uma melhora muito grande entre as soluções do CS e as soluções das meta-heurísticas, evidenciando que para essas instâncias uma solução do CPMP não é adequada para o CCCP.

O CS falhou em encontrar a melhor solução conhecida nas duas menores instâncias *doni*, mas obtêm soluções melhores que as da literatura nas outras cinco instâncias. Os tempos computacionais são razoáveis para essas instâncias considerando o tamanho das mesmas.

Novamente, pode-se notar que as versões do CS que utilizam uma meta-heurística para gerar soluções convergem mais rapidamente para uma boa solução que o CS com um gerador de soluções aleatórias.

Uma comparação entre o CS e as meta-heurísticas não pode ser analisada, pois as meta-heurísticas não resolvem diretamente o CCCP. Ao invés disso, estas resolvem o CPMP, e apenas as melhores soluções encontradas pelas meta-heurísticas são calculadas utilizando a função objetivo do CCCP. Por causa disso, algumas soluções obtidas pelas meta-heurísticas foram piores que as obtidas pelo método aleatório, mostrando que uma boa solução do CPMP pode ser horrível para o CCCP.

Tabela 4.9 - CCCP: Resultados Computacionais do CS-GA.

		CS-GA					GA				
problema	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
ta25	1251,44	**1251,45**	0,00	0,00	0,68	2,16	1273,46	0,00	1,76	1,10	1,76
ta50	4474,52	**4474,52**	0,00	0,00	0,99	5,52	**4474,52**	0,93	0,00	1,41	0,00
ta60	5356,58	**5356,58**	0,00	0,00	1,05	9,13	**5356,58**	0,00	0,00	1,60	0,00
ta70	6240,67	**6240,67**	0,00	0,00	0,77	8,78	6267,89	0,00	0,44	1,69	0,44
ta80	5515,46	5730,28	0,00	3,89	2,59	22,77	5775,69	0,26	4,72	2,21	0,79
ta90	8899,05	9069,85	0,00	1,92	1,26	22,11	9133,35	0,13	2,63	2,13	0,70
ta100	8102,04	**8102,04**	0,00	0,00	11,24	48,59	8189,44	0,35	1,08	3,10	1,08
sjc1	17359,75	**17359,75**	0,02	0,00	8,17	38,82	17363,47	0,05	0,02	3,10	0,02
sjc2	33181,65	**33181,65**	0,00	0,00	40,37	179,42	33324,04	0,52	0,43	8,26	0,43
sja3a	45354,38	45358,23	0,06	0,01	509,66	1206,72	46682,60	1,28	2,93	42,55	2,92
sjc3b	40661,94	**40661,94**	0,02	0,00	771,83	1433,80	42320,00	1,43	4,08	50,24	4,08
sjc4a	61931,60	**61931,60**	0,04	0,00	1092,97	3025,72	65978,89	1,41	6,54	97,70	6,54
sjc4b	52227,11	**52227,60**	0,06	0,00	1965,82	3995,20	55753,27	1,39	6,75	136,04	6,75
p3038-600	128203,40	128419,95	0,23	0,17	6137,67	9736,80	191210,20	2,01	49,15	2341,14	48,89
p3038-700	116051,88	116325,05	0,31	0,24	6848,52	11658,11	178078,89	2,13	53,45	2972,71	53,09
p3038-800	106961,20	107764,69	0,32	0,75	8335,36	13194,76	173381,82	2,09	62,10	3423,22	60,89
p3038-900	99756,59	99968,15	0,44	0,21	11726,17	15341,74	166990,29	2,00	67,40	4153,87	67,04
p3038-1000	92706,38	**92706,38**	1,22	0,00	10747,13	17128,41	164465,64	2,60	77,40	4721,74	77,40
doni1	3021,41	3027,63	0,44	0,21	86,38	127,04	3122,02	0,08	3,33	26,43	3,12
doni2	6080,70	6373,26	0,02	4,81	309,77	687,48	6394,96	0,09	5,17	88,32	0,34
doni3	8438,96	8507,42	0,64	0,81	624,12	928,41	8945,88	0,05	6,01	175,43	5,15
doni4	10952,27	**10952,27**	0,38	0,00	1069,07	2389,58	11130,16	0,14	1,62	320,09	1,62
doni5	11172,92	11209,99	0,11	0,33	2175,04	3624,40	11341,52	0,09	1,51	548,00	1,17
doni6	15722,67	**15722,67**	0,34	0,00	6174,83	10316,55	19226,96	3,14	22,29	3371,47	22,29
doni7	18596,74	**18596,74**	0,82	0,00	15860,55	26913,91	29915,77	2,64	60,87	7008,31	60,87
médias			**0,22**	**0,53**	**2980,08**	**4881,84**		**0,99**	**17,67**	**1180,08**	**17,10**

Tabela 4.10 - CCCP: Resultados Computacionais do CS-SA.

		CS-SA					SA				
problema	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
ta25	1251,44	**1251,45**	0,00	0,00	0,73	4,63	1273,46	0,00	1,76	4,11	1,76
ta50	4474,52	**4474,52**	0,00	0,00	0,89	7,93	4478,15	1,22	0,08	5,86	0,08
ta60	5356,58	**5356,58**	0,00	0,00	1,72	10,05	**5356,58**	0,23	0,00	6,37	0,00
ta70	6240,67	**6240,67**	0,00	0,00	1,08	11,33	6267,89	0,00	0,44	7,33	0,44
ta80	5515,46	5730,28	0,00	3,89	3,59	17,72	5772,96	0,24	4,67	8,74	0,74
ta90	8899,05	9069,85	0,00	1,92	0,99	15,64	**9069,85**	0,00	1,92	8,46	0,00
ta100	8102,04	**8102,04**	0,00	0,00	5,67	23,00	8151,86	0,11	0,61	9,31	0,61
sjc1	17359,75	**17359,75**	0,01	0,00	5,53	24,96	17359,75	0,76	0,00	11,55	0,00
sjc2	33181,65	**33181,65**	0,00	0,00	37,75	119,81	33496,66	1,09	0,95	25,26	0,95
sja3a	45354,38	**45354,38**	0,08	0,00	258,35	486,20	45981,52	2,50	1,38	51,68	1,38
sjc3b	40661,94	40663,44	0,06	0,004	209,16	540,90	41425,00	1,36	1,88	58,62	1,87
sjc4a	61931,60	**61931,60**	0,08	0,00	473,86	1249,87	64380,66	1,49	3,95	79,33	3,95
sjc4b	52227,11	52244,59	0,10	0,03	617,73	1645,37	53676,79	1,48	2,78	98,11	2,74
p3038-600	128203,40	128974,03	0,40	0,60	5459,54	9991,66	195708,77	1,93	52,65	2291,33	51,74
p3038-700	116051,88	116158,34	0,69	0,09	9205,09	11675,07	178834,68	2,47	54,10	2655,73	53,96
p3038-800	106961,20	**106961,20**	1,12	0,00	9210,10	13368,57	167938,49	2,39	57,01	3020,61	57,01
p3038-900	99756,59	**99756,59**	0,75	0,00	11389,33	15049,56	158751,83	2,72	59,14	3385,33	59,14
p3038-1000	92706,38	93148,68	0,53	0,48	14184,79	18698,52	153315,46	2,34	65,38	6454,56	64,59
doni1	3021,41	3033,50	0,21	0,40	34,39	76,22	3345,62	1,77	10,73	24,17	10,29
doni2	6080,70	6382,00	0,06	4,95	112,12	242,60	6660,87	4,09	9,54	47,29	4,37
doni3	8438,96	8485,64	0,67	0,55	203,42	543,19	9600,81	6,14	13,77	79,00	13,14
doni4	10952,27	11160,44	1,04	1,90	249,45	1011,08	13022,73	5,54	18,90	119,77	16,69
doni5	11172,92	11191,41	0,24	0,17	763,65	1617,99	13127,74	8,95	17,50	159,34	17,30
doni6	15722,67	15814,56	0,56	0,58	3326,34	8705,53	22941,98	2,02	45,92	558,46	45,07
doni7	18596,74	19038,91	1,00	2,38	10193,80	18926,76	28474,71	4,21	53,12	1021,77	49,56
médias			**0,30**	**0,72**	**2637,96**	**4162,57**		**2,20**	**19,13**	**807,68**	**18,30**

Tabela 4.11 - CCCP: Resultados Computacionais do CS-VNS.

		CS-VNS					VNS				
problema	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
ta25	1251,44	**1251,45**	0,00	0,00	0,70	6,64	1273,46	0,00	1,76	6,12	1,76
ta50	4474,52	**4474,52**	0,00	0,00	0,87	6,67	4478,15	0,89	0,08	4,44	0,08
ta60	5356,58	**5356,58**	0,00	0,00	1,22	6,70	**5356,58**	0,35	0,00	3,34	0,00
ta70	6240,67	**6240,67**	0,00	0,00	0,93	6,97	**6240,67**	0,38	0,00	2,71	0,00
ta80	5515,46	5730,28	0,00	3,89	5,27	10,98	5754,20	0,74	4,33	3,05	0,42
ta90	8899,05	9069,85	0,00	1,92	0,98	11,35	**9069,85**	0,36	1,92	3,15	0,00
ta100	8102,04	**8102,04**	0,00	0,00	5,10	16,66	8153,64	0,65	0,64	3,60	0,64
sjc1	17359,75	**17359,75**	0,01	0,00	4,61	16,20	17434,31	1,20	0,43	3,87	0,43
sjc2	33181,65	**33181,65**	0,00	0,00	25,60	92,74	33414,02	1,62	0,70	9,22	0,70
sja3a	45354,38	45370,94	0,18	0,04	159,75	379,09	46484,15	0,99	2,49	16,99	2,45
sjc3b	40661,94	40696,13	0,22	0,08	235,95	434,92	41678,10	0,94	2,50	15,62	2,41
sjc4a	61931,60	61973,99	0,10	0,07	547,97	1019,04	64161,07	0,65	3,60	28,58	3,53
sjc4b	52227,11	52301,16	0,17	0,14	279,52	1253,95	54252,81	0,62	3,88	26,17	3,73
p3038-600	128203,40	**128203,40**	0,51	0,00	4147,34	9405,32	195578,60	2,65	52,55	790,14	52,55
p3038-700	116051,88	116129,45	0,60	0,07	6984,41	10949,37	175380,88	3,23	51,12	834,35	51,02
p3038-800	106961,20	107209,82	0,66	0,23	5885,53	12433,83	165111,04	2,84	54,37	872,41	54,01
p3038-900	99756,59	100133,85	0,65	0,38	8071,89	13965,54	161104,30	2,12	61,50	903,93	60,89
p3038-1000	92706,38	92850,88	1,04	0,16	7430,95	15411,06	153022,17	2,66	65,06	925,60	64,80
doni1	3021,41	3028,82	0,24	0,25	69,50	106,88	3048,38	2,00	0,89	48,76	0,65
doni2	6080,70	6376,47	0,05	4,86	141,85	310,52	6403,54	0,89	5,31	187,36	0,42
doni3	8438,96	8470,18	0,45	0,37	457,25	760,89	8470,18	2,22	0,37	423,64	0,00
doni4	10952,27	11095,35	0,63	1,31	872,55	1266,82	11110,21	1,17	1,44	718,80	0,13
doni5	11172,92	11183,23	0,34	0,09	1217,95	2020,33	11305,92	1,98	1,19	1210,10	1,10
doni6	15722,67	15825,06	0,77	0,65	5559,99	10491,32	16187,74	2,62	2,96	5066,76	2,29
doni7	18596,74	18623,59	2,38	0,14	14163,11	20122,55	19538,69	4,14	5,07	9757,34	4,91
médias			**0,36**	**0,59**	**2250,83**	**4020,25**		**1,52**	**12,97**	**874,64**	**12,36**

Tabela 4.12 - CCCP: Resultados Computacionais do CS-ILS.

		CS-ILS					ILS				
problema	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
ta25	1251,44	**1251,45**	0,00	0,00	0,73	6,65	**1273,46**	0,00	1,76	6,06	1,76
ta50	4474,52	**4474,52**	0,00	0,00	0,91	6,67	4478,15	0,72	0,08	4,16	0,08
ta60	5356,58	**5356,58**	0,00	0,00	1,55	6,73	5356,58	0,96	0,00	2,83	0,00
ta70	6240,67	**6240,67**	0,00	0,00	0,87	7,09	6240,67	0,39	0,00	2,59	0,00
ta80	5515,46	5730,28	0,00	3,89	5,03	12,88	5752,19	0,95	4,29	2,72	0,38
ta90	8899,05	9069,85	0,00	1,92	0,89	10,58	9069,85	0,25	1,92	2,82	0,00
ta100	8102,04	**8102,04**	0,00	0,00	7,82	17,41	8134,38	0,49	0,40	3,24	0,40
sjc1	17359,75	**17359,75**	0,01	0,00	7,88	19,28	17389,98	1,75	0,17	3,61	0,17
sjc2	33181,65	**33181,65**	0,00	0,00	11,74	104,84	33928,54	0,84	2,25	8,38	2,25
sja3a	45354,38	45359,09	0,11	0,01	135,98	388,32	46679,63	1,59	2,92	18,39	2,91
sjc3b	40661,94	**40661,94**	0,04	0,00	261,91	458,79	41907,77	1,98	3,06	20,53	3,06
sjc4a	61931,60	61944,86	0,05	0,02	401,17	893,54	63999,34	2,16	3,34	31,00	3,32
sjc4b	52227,11	**52227,11**	0,07	0,00	540,30	1199,82	54752,31	1,23	4,84	38,00	4,84
p3038-600	128203,40	128530,89	0,52	0,26	3842,84	9436,96	194415,70	2,16	51,65	791,98	51,26
p3038-700	116051,88	116660,01	0,41	0,52	5359,16	10915,27	176055,06	3,64	51,70	832,79	50,91
p3038-800	106961,20	107627,78	0,37	0,62	5893,56	12810,04	163767,65	4,56	53,11	891,83	52,16
p3038-900	99756,59	100571,28	0,35	0,82	5386,27	13931,29	158491,54	3,16	58,88	898,12	57,59
p3038-1000	92706,38	93508,48	0,51	0,87	4474,16	15451,21	153233,19	2,14	65,29	921,70	63,87
doni1	3021,41	3026,87	0,30	0,18	70,41	105,91	3050,12	1,80	0,95	44,63	0,77
doni2	6080,70	6377,77	0,05	4,89	316,78	410,20	6420,40	0,48	5,59	172,96	0,67
doni3	8438,96	8501,53	0,30	0,74	711,60	924,60	8538,09	2,58	1,17	394,33	0,43
doni4	10952,27	11094,62	0,88	1,30	1395,03	1681,27	11094,62	1,36	1,30	695,84	0,00
doni5	11172,92	11182,87	0,19	0,09	2297,91	2730,88	11335,45	1,09	1,45	1087,83	1,36
doni6	15722,67	15780,97	0,34	0,37	10005,59	14466,46	16235,22	1,99	3,26	4755,58	2,88
doni7	18596,74	18697,35	1,59	0,54	20649,82	29751,42	19712,01	2,78	6,00	8945,77	5,43
médias			**0,24**	**0,68**	**2471,20**	**4629,92**		**1,64**	**13,02**	**823,11**	**12,26**

Tabela 4.13 - CCCP: Resultados Computacionais do CS-Aleatório.

		CS-Aleatório					Aleatório				
problema	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
ta25	1251,44	**1251,45**	0,00	0,00	0,79	7,58	**1251,45**	0,88	0,00	6,92	0,00
ta50	4474,52	**4474,52**	0,00	0,00	0,97	7,61	4492,84	1,96	0,41	4,96	0,41
ta60	5356,58	**5356,58**	0,00	0,00	2,45	7,65	5418,77	3,08	1,16	3,73	1,16
ta70	6240,67	**6240,67**	0,00	0,00	0,93	7,77	6286,23	1,12	0,73	3,08	0,73
ta80	5515,46	5730,28	0,00	3,89	2,95	12,49	5834,42	5,32	5,78	2,29	1,82
ta90	8899,05	9069,85	0,00	1,92	0,95	10,60	9103,22	1,58	2,29	2,38	0,37
ta100	8102,04	**8102,04**	0,00	0,00	7,16	16,81	8223,37	6,28	1,50	2,28	1,50
sjc1	17359,75	**17359,75**	0,00	0,00	7,76	18,56	18281,60	5,66	5,31	2,38	5,31
sjc2	33181,65	**33181,65**	0,00	0,00	23,09	97,97	36136,50	2,81	8,91	3,41	8,91
sja3a	45354,38	45382,84	0,06	0,06	185,19	366,61	50294,21	2,92	10,89	5,03	10,82
sjc3b	40661,94	**40661,94**	0,02	0,00	206,03	420,05	44767,05	3,35	10,10	5,39	10,10
sjc4a	61931,60	**61931,60**	0,05	0,00	538,90	821,72	70415,13	1,98	13,70	7,31	13,70
sjc4b	52227,11	**52227,52**	0,07	0,00	644,63	1093,52	58032,72	2,94	11,12	8,23	11,12
p3038-600	128203,40	128984,05	0,31	0,61	6488,17	9032,81	190551,27	0,92	48,63	735,00	47,73
p3038-700	116051,88	**116051,88**	0,86	0,00	7550,47	10563,79	172470,89	1,77	48,62	820,91	48,62
p3038-800	106961,20	107566,92	0,45	0,57	7282,39	12126,34	162060,79	1,98	51,51	891,22	50,66
p3038-900	99756,59	100371,69	0,61	0,62	10059,25	13634,70	155194,42	1,80	55,57	937,49	54,62
p3038-1000	92706,38	93498,48	0,55	0,85	12706,41	15140,73	147990,90	3,13	59,63	1011,56	58,28
doni1	3021,41	3028,50	0,26	0,23	58,86	75,07	3186,30	3,37	5,46	5,53	5,21
doni2	6080,70	6376,38	0,07	4,86	138,53	263,69	6546,00	5,51	7,65	12,07	2,66
doni3	8438,96	**8438,96**	0,94	0,00	421,08	635,17	9437,62	3,46	11,83	22,08	11,83
doni4	10952,27	11131,69	0,86	1,64	849,87	1190,21	12522,47	3,68	14,34	35,87	12,49
doni5	11172,92	**11172,92**	0,24	0,00	1340,05	1888,70	12748,89	4,89	14,11	53,66	14,11
doni6	15722,67	15748,61	0,48	0,16	8431,54	11221,04	21380,66	3,89	35,99	238,80	35,76
doni7	18596,74	18716,39	0,72	0,64	21155,87	25590,30	27293,93	4,17	46,77	493,02	45,83
médias			**0,26**	**0,64**	**3124,17**	**4170,06**		**3,14**	**18,88**	**212,58**	**18,15**

Tabela 4.14 - CCCP: Resultados Computacionais da Literatura.

		Negreiros e Palhano (2006)		Palhano et al. (2008)		Pereira e Senne (2008)	
problema	**melhor**	**sol***	**gap**	**sol***	**gap**	**sol***	**gap**
ta25	1251,44	**1251,44**	0,00	-	-	1280,49	2,32
ta50	4474,52	4476,12	0,04	-	-	**4474,52**	0,00
ta60	5356,58	**5356,58**	0,00	-	-	5357,34	0,01
ta70	6240,67	6241,55	0,01	-	-	**6240,67**	0,00
ta80	5515,46	**5730,28**	3,89	-	-	**5515,46**	0,00
ta90	8899,05	9103,21	2,29	-	-	**8899,05**	0,00
ta100	8102,04	8122,67	0,25	-	-	8168,36	0,82
sjc1	17359,75	17696,53	1,94	20341,34	17,18	17375,36	0,09
sjc2	33181,65	33423,84	0,73	35211,99	6,12	33357,75	0,53
sja3a	45354,38	47985,29	5,80	50590,49	11,54	45379,69	0,06
sjc3b	40661,94	-	-	-	-	41185,18	1,29
sjc4a	61931,60	66689,96	7,68	69283,05	11,87	61969,06	0,06
sjc4b	52227,11	-	-	-	-	52989,44	1,46
p3038-600	128203,40	192024,83	49,78	135481,99	5,68	-	-
p3038-700	116051,88	176731,07	52,29	123698,76	6,59	-	-
p3038-800	106961,20	184502,38	72,49	117705,48	10,05	-	-
p3038-900	99756,59	176781,51	77,21	111033,27	11,30	-	-
p3038-1000	92706,38	159139,89	71,66	110049,78	18,71	-	-
doni1	3021,41	**3021,41**	0,00	3234,58	7,06	-	-
doni2	6080,70	**6080,70**	0,00	6692,71	10,06	-	-
doni3	8438,96	8769,05	3,91	9797,12	16,09	-	-
doni4	10952,27	11516,14	5,15	11594,07	5,86	-	-
doni5	11172,92	11635,18	4,14	11827,69	5,86	-	-
doni6	15722,67	18443,50	17,31	-	-	-	-
doni7	18596,74	23478,79	26,25	-	-	-	-
média			**17,51**		**10,28**		**0,51**

4.10 Considerações Finais

Este Capítulo apresentou os resultados da aplicação do CS a dois problemas de localização de facilidades, o Problema de p-Medianas Capacitado (CPMP) e o Problema de Agrupamento Centrado Capacitado (CCCP). Cinco versões do CS foram implementadas utilizando diferentes métodos para gerar soluções para o processo de agrupamento.

Os resultados do CS para o CPMP mostram que essa abordagem é competitiva para resolver esse problema em um tempo computacional razoável. Todas as versões do CS encontraram o ótimo global para o primeiro conjunto de instâncias considerado nos testes computacionais. Para o segundo conjunto também foi obtida a solução ótima para a maioria das instâncias, exceto em uma instância na qual soluções próximas do ótimo foram encontradas.

As mesmas versões do CS implementadas para o CPMP foram adaptadas ao CCCP, nas quais foi modificado somente o cálculo da função objetivo. O CS foi testado em diferentes tipos de instâncias, e encontrou resultados melhores que os conhecidos na literatura em quase todas. Para algumas instâncias o CS obteve melhorias significativas em relação à qualidade da solução.

Todas as versões do CS tiveram comportamentos semelhantes, encontrando boas soluções em um tempo computacional competitivo. Porém, observa-se que as abordagens que utilizam uma meta-heurística como gerador de soluções para o processo de agrupamento convergem mais rapidamente para a melhor solução encontrada pelo método. No CS com gerador de soluções aleatórias, apesar de obter soluções boas, tem-se um tempo de processamento maior até o método convergir. Este comportamento pode ser explicado pelo fato da geração aleatória de soluções não levar em consideração o espaço de busca do problema.

Portanto, estes resultados validam a aplicação do CS ao CPMP e ao CCCP. Esta abordagem é uma alternativa interessante para resolver esses problemas com menos custos econômicos, uma vez que, não utiliza nenhum *software* comercial.

5 PROBLEMAS DO CAIXEIRO VIAJANTE COM LUCRO

5.1 Introdução

O Problema do Caixeiro Viajante (TSP, do inglês *Traveling Salesman Problem*) (DANTZIG et al., 1954) é um dos mais tradicionais e conhecidos problemas de otimização combinatória. O TSP consiste em otimizar uma sequência de visitas a clientes partindo de um depósito central, sendo que, uma característica deste problema é que todos clientes precisam ser visitados exatamente uma vez e, consequentemente, nenhum valor é associado ao serviço de atendimento ao cliente. Porém, algumas generalizações deste problema propõem selecionar clientes dependendo de um valor de prêmio (benefício) que é ganho quando a visita acontecer. Essa característica dá origem a uma classe de problemas que na literatura foi nomeada Problemas do Caixeiro Viajante com Lucros (TSPP, do inglês *Traveling Salesman Problems with Profits*) (FEILLET et al., 2005).

Os TSPP podem ser vistos como problemas do caixeiro viajante bi-objetivos com dois objetivos opostos, um que pressiona o caixeiro a viajar (ou seja, coletar prêmios) e outro que estimula o caixeiro a minimizar os custos de viagem (permitindo a ele não visitar alguns clientes). Visto assim, resolver o TSPP deveria resultar em encontrar uma fronteira de Pareto, ou seja, um conjunto de soluções viáveis tal que nenhum objetivo possa ser melhorado sem deteriorar o outro. Na prática, a maioria das pesquisas existentes sobre esses problemas os trata como versões de único objetivo. Assim, ou os dois objetivos são calculados e combinados linearmente, ou um dos objetivos é transformado em restrição com um valor limite especificado.

Problemas com essas características são de fácil adaptação a situações da vida real, tendo várias aplicações práticas e de relevância econômica. Em linhas gerais, pode ser associado a um universo de clientes em potencial, a um conjunto de cidades ou a pontos turísticos a serem visitados, onde se deseja determinar quem deve ser visitado e qual a sequência das visitas.

Os TSPP pertencem à classe de problemas NP-*hard*. Isto é intuitivamente claro, uma vez que, uma solução do TSP pode ser declarada como uma solução de um TSPP em que se precisa coletar todos os prêmios dos vértices, sendo o TSP classificado como NP-*hard* (FEILLET et al., 2005).

Neste Capítulo é abordado um dos problemas da classe TSPP, chamado Problema

do Caixeiro Viajante com Coleta de Prêmios (PCTSP, do inglês *Prize Collecting Traveling Salesman Problem*) (BALAS, 1989). Propõe-se aplicar o método CS para resolver esse problema de forma aproximada.

O *software* CPLEX (ILOG, 2007) foi utilizado para resolver a formulação matemática do PCTSP, objetivando validar os resultados computacionais obtidos pelo CS. O CPLEX é capaz de resolver apenas instâncias de pequeno porte.

5.2 Definições e Formulações Matemáticas

Os TSPP, de uma forma geral, podem ser representados em um grafo completo $G = (V, A)$, onde $V = \{v_1, v_2, ..., v_n\}$ é um conjunto de n vértices e A é um conjunto de arcos (grafo direcionado) ou arestas (grafo não direcionado). Para cada vértice $v_i \in V$ existe associado um prêmio p_{v_i}, e cada aresta $(v_i, v_j) \in A$ possui um custo de deslocamento $c_{v_i v_j}$. Supondo que o vértice v_1, sem perda de generalidade, seja o depósito ou a cidade de origem do caixeiro, esse vértice deve ter prêmio nulo ($p_{v_1} = 0$). O objetivo é determinar um circuito elementar (ou seja, um circuito no qual cada vértice é visitado no máximo uma vez) que contenha o vértice v_1, levando em consideração o prêmio coletado e os custos de deslocamento.

Os problemas que formam os TSPP surgem das diferentes maneiras que existem para tratar os objetivos:

1. Os dois objetivos são tratados separadamente, gerando um problema multiobjetivo, onde os objetivos são minimizar os custos de deslocamento e maximizar os prêmios coletados;

2. Ambos os objetivos são combinados numa função objetivo, visando encontrar um circuito que minimize os custos de deslocamento menos os prêmios coletados;

3. O objetivo do custo de deslocamento é definido como uma restrição, e o objetivo é encontrar um circuito que maximize o prêmio coletado tal que o custo de deslocamento não exceda um valor máximo;

4. O objetivo do prêmio é definido como uma restrição, e o objetivo é encontrar um circuito que minimize os custos de deslocamento e que o prêmio coletado não seja menor que um valor pré-definido.

No problema 1 temos um problema bi-objetivo conhecido como Problema do Vendedor com Multiobjetivos (*Multiobjetive Vending Problem*) (KELLER; GOODCHILD, 1988), que consiste em minimizar os custos de deslocamento e maximizar os prêmios coletados.

O problema 2 é conhecido na literatura como Problema de Rotas Lucrativas (PTP, do inglês *Profitable Tour Problem*) (DELL'AMICO et al., 1995). Nesse problema é inserido um conceito de penalidade para cada vértice ($\gamma_{v_i}, \forall v_i \in V$), e ao invés de coletar um prêmio por visitar uma cidade, o caixeiro paga uma penalidade caso não visite alguma cidade. Portanto, o objetivo é minimizar os custos de deslocamento somado às penalidades pagas nas cidades não visitadas. Uma restrição de mochila pode ser adicionada ao problema 2, sendo esse problema definido como Problema do Caixeiro Viajante com Sub-rotas (*Traveling Salesman Subtour Problem*) (GENSCH, 1978), onde existe associado a cada vértice um lucro e um peso. O objetivo deste problema é minimizar os custos de deslocamentos menos o lucro dos vértices que são visitados. Para a solução ser viável é preciso que a soma dos pesos dos vértices visitados não ultrapasse um limite pré-determinado.

O problema 3 é chamado de Problema Orientado (*Orienteering Problem*) (TSILIGIRIDES, 1984; GOLDEN et al., 1987). Esse problema é definido geralmente em termos de um caminho, ao invés de um circuito, no qual o objetivo é encontrar uma rota iniciando na origem e finalizando no destino, visitando um subconjunto de vértices de modo que a soma dada pelo total de prêmios coletados seja maximizada e a rota seja completada dentro de um limite de custo ou tempo pré-estipulado. Existem na literatura alguns trabalhos que consideram esse problema como um circuito, classificando-o como Problema do Caixeiro Viajante Seletivo (*Selective Traveling Salesman Problem*) (LAPORTE; MARTELLO, 1990) ou Problema de Máxima Coleta (*Maximum Collection Problem*) (KATAOKA; MORITO, 1988).

O problema 4 é definido como Problema do Caixeiro Viajante com Coleta de Prêmios (PCTSP) (BALAS, 1989). Assim como no PTP, nesse problema também é inserido o conceito de penalidades, porém o objetivo do prêmio mínimo é levado em consideração, sendo tratado como uma restrição. Portanto, o PCTSP consiste em minimizar o somatório dos custos de deslocamento somado às penalidades pagas nos vértices não visitados, e necessita incluir no circuito um número suficiente de vértices que permitam coletar um prêmio mínimo (p_{min}) pré-estabelecido. Existem também na literatura, trabalhos que não consideram as penalidades para os vértices

não visitados, sendo esse problema chamado de Problema do Caixeiro Viajante com Quota (QTSP, do inglês *Quota Traveling Salesman Problem*) (AWERBUCH et al., 1998).

Os problemas PCTSP, PTP e QTSP são muito similares, como pode ser observado por meio das formulações matemáticas desses problemas apresentadas a seguir, diferenciando basicamente no cálculo da função objetivo. Desta forma, é possível resolver os três problemas utilizando a mesma abordagem do CS, conforme apresentado em Chaves e Lorena (2007). Porém, neste Capítulo foi tratado somente o PCTSP, visto que, o CS apresentou comportamento semelhante para os três problemas.

As formulações matemáticas encontradas na literatura para os TSPP possuem algumas restrições semelhantes, sendo assim, é possível definir um conjunto de restrições básicas para essas formulações. Considere $x_{v_i v_j} (v_i, v_j \in V, v_i \neq v_j)$ sendo uma variável binária igual a 1 se a aresta (v_i, v_j) pertencer à solução, e $x_{v_i v_j}$ igual a 0 caso contrário, e uma variável binária $y_{v_i} (v_i \in V)$ que controla se o vértice v_i está presente na solução, assumindo valor 1 caso seja visitado e valor 0 caso contrário.

Todos TSPP compartilham as restrições a seguir.

$$\sum_{v_j \in V \setminus v_i} x_{v_i v_j} = y_{v_i} \qquad \forall v_i \in V \tag{5.1}$$

$$\sum_{v_i \in V \setminus v_j} x_{v_i v_j} = y_{v_j} \qquad \forall v_j \in V \tag{5.2}$$

$$\text{restrições para eliminar sub-rotas} \tag{5.3}$$

$$y_{v_1} = 1 \tag{5.4}$$

$$x_{v_i v_j} \in \{0, 1\} \qquad \forall v_i, v_j \in V \tag{5.5}$$

$$y_{v_i} \in \{0, 1\} \qquad \forall v_i \in V \tag{5.6}$$

As restrições 5.1 e 5.2 são chamadas restrições de atribuição e garantem que cada vértice seja visitado no máximo uma vez. A restrição 5.3 elimina as sub-rotas que não envolvem o depósito (vértice v_1). A restrição 5.4 assegura que o depósito seja visitado. As restrições 5.5 e 5.6 asseguram que as variáveis x_{ij} e y_i sejam binárias.

A função objetivo, por outro lado, é diferente para cada um dos problemas, e uma restrição de recurso também precisa ser levada em consideração em alguns casos.

Para o PTP, a função objetivo é

$$\min \sum_{v_i \in V} \sum_{v_j \in V} c_{v_i v_j} x_{v_i v_j} + \sum_{v_i \in V} \gamma_{v_i} (1 - y_{v_i}) \tag{5.7}$$

sujeito a (5.1 - 5.6).

Para o PCTSP, a formulação é

$$\min \sum_{v_i \in V} \sum_{v_j \in V} c_{v_i v_j} x_{v_i v_j} + \sum_{v_i \in V} \gamma_{v_i} (1 - y_{v_i}) \tag{5.8}$$

sujeito a (5.1 - 5.6) e

$$\sum_{v_i \in V} p_{v_i} y_{v_i} \geq p_{min} \tag{5.9}$$

Para o QTSP, a função objetivo é

$$\min \sum_{v_i \in V} \sum_{v_j \in V} c_{v_i v_j} x_{v_i v_j} \tag{5.10}$$

sujeito a (5.1 - 5.6) e (5.9).

A restrição 5.9 é conhecida como restrição de cobertura generalizada, e assegura que a soma dos prêmios coletados seja maior que o prêmio mínimo (p_{min}) definido a *priori*.

A restrição 5.3 pode ser tratada de diferentes formas, neste trabalho optou-se por utilizar um conjunto de restrições proposto em Torres e Brito (2003) para proibir a formação de sub-rotas. Essas restrições foram desenvolvidas por meio da inclusão de variáveis de fluxo, sendo que, a vantagem desse conjunto de restrições é que o número de restrições utilizadas é polinomial, ao contrário das restrições comumente usadas no TSP, que exigem um número exponencial de restrições para o atendimento dessa condição.

Esse conjunto de restrições é apresentado a seguir, sendo a variável $f_{v_i v_j}$ $(v_i, v_j \in V, v_i \neq v_j)$ a quantidade de fluxo (valor em prêmios) escoada na aresta (v_i, v_j).

$$\sum_{v_j \in V \setminus \{v_1\}} f_{v_1 v_j} = 0 \tag{5.11}$$

$$\sum_{v_j \in V \setminus \{v_i\}} f_{v_i v_j} = \sum_{v_j \in V \setminus \{v_i\}} f_{v_j v_i} + p_{v_i} y_{v_i} \qquad \forall v_i \in V \setminus \{v_1\} \tag{5.12}$$

$$\sum_{v_j \in V \setminus \{v_1\}} f_{v_j v_1} = \sum_{v_j \in V \setminus \{v_1\}} p_{v_j} y_{v_j} \tag{5.13}$$

$$f_{v_i v_j} > x_{v_i v_j} - 1 \qquad \forall v_i \in V \setminus \{v_1\}, \;\; \forall v_j \in V, \;\; v_i \neq v_j \tag{5.14}$$

$$\left(\sum_{v_j \in V} p_{v_j} \right) x_{v_i v_j} \geq f_{v_i v_j} \qquad \forall v_i \in V \setminus \{v_1\}, \;\; \forall v_j \in V \tag{5.15}$$

$$f_{v_i v_j} \geq 0 \qquad \forall v_i, v_j \in V \tag{5.16}$$

As restrições 5.11, 5.12 e 5.13 garantem a conectividade da solução, ou seja, evitam rotas desconexas da origem. As variáveis de fluxo $f_{v_i v_j}$ impedem que sub-rotas sejam criadas, associando a quantidade de prêmio do vértice visitado à aresta que sai deste. A quantidade de fluxo que sai do vértice origem (v_1) tem que ser igual a 0, e a quantidade de fluxo que volta para v_1 tem que ser igual à soma dos prêmios coletados na rota. As restrições 5.14 e 5.15 conectam as variáveis $x_{v_i v_j}$ e $f_{v_i v_j}$, fazendo com

que a rota gerada por ambas seja a mesma. As restrições 5.14 garantem que se uma aresta fizer parte da solução, a quantidade de fluxo escoada por esta tem que ser maior que 0, e as restrições 5.15 asseguram que o valor do fluxo escoado por uma aresta não será maior que o total de prêmios de todos os vértices da rota. Por fim, as restrições 5.16 garantem que as variáveis $f_{v_i v_j}$ sejam não-negativas.

5.3 Revisão Bibliográfica

O PCTSP foi introduzido no meio acadêmico por Balas (1989), que apresentou algumas propriedades estruturais do problema e duas formulações matemáticas para este, sendo a segunda uma simplificação da primeira. O autor apresenta também um algoritmo para o problema, que combina a técnica de planos de corte com uma relaxação de programação linear a ser resolvida pelo método *simplex*. Balas e Martin (1985) desenvolveram uma aplicação prática do PCTSP, produzindo um *software* que utilizou a combinação de várias heurísticas, objetivando encontrar boas soluções para a programação diária de uma indústria de bobinas de aço.

Fischetti e Toth (1988) apresentam diferentes limitantes inferiores para o PCTSP, obtidos por meio da exploração de diferentes relaxações do problema, utilizando relaxação Lagrangiana e relaxação baseada em disjunção. Os autores descrevem também um algoritmo *branch* e *bound* para obter a solução ótima para o PCTSP, sendo este aplicado apenas às instâncias pequenas.

Dell'Amico et al. (1995) apresentam uma nova abordagem para obter limitantes inferiores para o PTP e o PCTSP. Utilizando a formulação apresentada em Balas (1989) eles propuseram uma relaxação Lagrangiana, relaxando a restrição de prêmio mínimo, e utilizam o método de subgradientes para resolver o problema dual.

Dell'Amico et al. (1998) propuseram um outro algoritmo para PCTSP, utilizando a relaxação Lagrangiana apresentada anteriormente, e um algoritmo baseado numa heurística clássica para o TSP, conhecida como heurística de Inserção Mais Barata (*Cheapest Insertion heuristic*), para transformar a solução relaxada em uma solução viável. Propuseram ainda, utilizar uma heurística chamada Extensão e Colapso (*Extension and Collapse heuristic*) para melhorar a solução viável obtida.

Awerbuch et al. (1998) propuseram um algoritmo aproximativo para o QTSP com performance $O(\log^2 n)$, para isso utilizaram um algoritmo que agrupava os pontos do problema dentro de componentes, assim como o algoritmo de Kruskal, e procurava

unir (usando o caminho mais curto) os dois componentes que apresentassem a menor razão entre a distância entre eles e o número mínimo de pontos dentro dos dois componentes.

Gomes et al. (2000) e Melo e Martinhon (2004) apresentam meta-heurísticas híbridas para solucionar o PCTSP. O primeiro combinando GRASP e VND, e o segundo combinando GRASP Progressivo e VNS como método de busca local. Torres e Brito (2003) apresentam uma nova formulação matemática para o PCTSP baseada na formulação apresentada em Balas (1989). Nessa formulação é proposto um novo conjunto de restrições para evitar a formação de sub-rotas na solução.

Chaves et al. (2004) e Chaves et al. (2007) exploraram duas abordagens para o PCTSP, a primeira é uma abordagem exata capaz de resolver instâncias de pequenas dimensões, e a segunda é uma abordagem heurística, combinando as meta-heurísticas GRASP e VNS/VND.

Outras informações sobre os TSPP podem ser encontradas em Feillet et al. (2005), onde os autores apresentam uma pesquisa sobre os TSPP, identificando e comparando diferentes classes de aplicação, modelando algumas formulações e analisando a complexidade de algumas técnicas de solução exatas e heurísticas.

Ausiello et al. (2006) apresentam uma versão *online* do PCTSP, na qual os pedidos chegam em tempo real e o caixeiro (servidor) precisa decidir quais pedidos atender e em que ordem atender, sem ainda conhecer a sequência completa de pedidos. O objetivo é, assim como no PCTSP *offline*, coletar um prêmio mínimo enquanto minimiza a soma do tempo para completar a rota e as penalidades associadas aos pedidos não atendidos na rota. Os autores desenvolveram um algoritmo 7/3 aproximativo para esta versão *online* do PCTSP.

Tang e Wang (2008) propuseram uma variação do PCTSP, chamada *Capacitated Prize-Collecting Travelling Salesman Problem*. O objetivo do problema é minimizar os custos de viagem e as penalidades pagas nos clientes não visitados, de tal forma que a quantidade de prêmios coletados seja grande o suficiente e as demandas dos clientes visitados não excedam a capacidade do caixeiro. Os autores utilizaram a meta-heurística Busca Local Iterativa (ILS) para resolver esse problema, obtendo resultados eficientes quando comparados a outros métodos e também aos resultados de um caso real.

Dois trabalhos recentes resolveram o TSPP de forma bi-objetiva utilizando o conceito de otimalidade de Pareto. Jozefowiez et al. (2008) apresentam uma meta-heurística híbrida, a qual utiliza um processo de *ejection chain* com um algoritmo evolutivo multiobjetivo. Bérubé et al. (2008) apresentam um algoritmo exato que divide o problema bi-objetivo em subproblemas, e duas heurísticas que auxiliam o algoritmo *branch* e *cut* na resolução dos subproblemas.

5.4 Representação do Problema

A representação de uma solução do PCTSP é definida por meio de dois vetores dinâmicos. O primeiro vetor armazena os vértices que fazem parte da rota na sequência em que são visitados. O segundo vetor armazena os vértices não visitados. A Figura 5.1 ilustra a representação de uma solução, na qual a sequência de visitas é $(v_1, v_3, v_6, v_{10}, v_4, v_9, v_7)$ e os vértices v_2, v_5 e v_8 não são visitados.

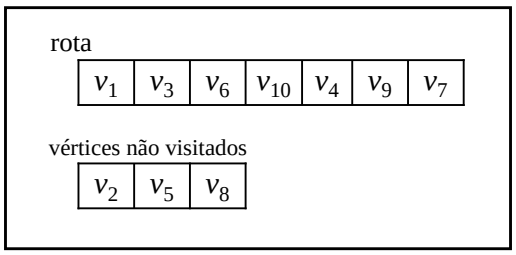

Figura 5.1 - Exemplo da representação de uma solução do PCTSP.

Assim, é possível realizar o cálculo da função objetivo de forma rápida. Esta representação também torna simples a definição de movimentos que permitam uma busca ampla sobre o espaço de busca.

5.5 Estruturas de Vizinhança

Três movimentos são definidos possibilitando que qualquer solução possa ser alcançada no espaço de busca, por meio de um número finito de movimentos. Os movimentos são:

- m_1: inserir um vértice que não está sendo visitado;
- m_2: retirar um vértice que está sendo visitado;
- m_3: trocar de posição dois vértices que estão sendo visitados;

O único movimento que pode causar inviabilidade é o movimento m_2, pois o prêmio mínimo pode deixar de ser coletado. Contudo, soluções inviáveis são permitidas desde que sejam penalizadas no valor da função objetivo.

5.6 Função Objetivo

O cálculo da função objetivo do PCTSP é realizado considerando os dois vetores que formam uma solução. A partir do primeiro vetor calcula-se a distância percorrida na rota e o total de prêmios coletados nos vértices visitados. O segundo vetor fornece a quantidade de penalidades pagas nos vértices não visitados. Caso o total de prêmios coletados seja menor que o prêmio mínimo, a solução é penalizada. Portanto, uma solução é avaliada com base na seguinte função objetivo:

$$f(s) = \left(\sum_{i=1}^{n_r} c_{i,i+1} + \sum_{j=1}^{n_u} \gamma_j \right) + \psi * f_p \tag{5.17}$$

sendo n_r e n_u o tamanho dos vetores que contém os vértices visitados e não visitados, respectivamente, e ψ o peso que reflete a penalidade imposta se o prêmio mínimo não for coletado ($f_p = max\{0, p_{min} - \sum_{i=1}^{n_r} p_i\}$). Neste trabalho foi utilizado $\psi = 10^8$.

5.7 Medida de Distância entre duas Soluções

A medida de distância entre duas soluções do PCTSP é baseada na similaridade entre duas rotas. Sendo assim, a distância é dada pelo número de arestas diferentes entre duas soluções. Quanto maior o número de arestas diferentes entre duas soluções, maior será a distância entre elas e menor será a similaridade.

O custo computacional dessa medida de distância é significativo, pois para cada aresta é preciso localizar os vértices referentes a esta na outra solução. Porém, este custo é compensado pela qualidade retornada na classificação das soluções em clusters similares.

5.8 CS Aplicado ao PCTSP

Nesta seção é apresentada uma abordagem heurística para resolver o PCTSP, na qual cinco versões do CS são implementadas com diferentes métodos geradores de soluções para o processo de agrupamento.

Uma solução inicial para o PCTSP é gerada de forma construtiva, via um procedimento parcialmente aleatório baseado na fase de construção da meta-heurística GRASP (FEO; RESENDE, 1995).

Inicialmente insere-se na rota o vértice origem (v_1) e dois vértices escolhidos aleatoriamente. Em seguida, utiliza-se uma função de inserção (Equação 5.18) para construir a lista dos elementos candidatos a serem inseridos na rota. Forma-se, então, uma lista de candidatos restrita de tamanho |LCR|, contendo os melhores candidatos, e seleciona-se um candidato dessa lista aleatoriamente. O vértice do candidato é inserido na rota e a lista de candidatos é atualizada a cada iteração. Uma solução é dita construída quando não houver mais candidatos com função de inserção positiva e o prêmio mínimo tenha sido coletado. Se |LCR| = 1, gera-se uma solução gulosa, e se |LCR| for igual ao número de candidatos será gerada uma solução aleatória. Neste trabalho utilizou-se |LCR| igual a 20. A Figura 5.2 apresenta o pseudocódigo desse método.

```
algoritmo ConstruirSolucao
    s = ∅
    enquanto (solução não construída) faça
        produzir a lista de candidatos
        e = selecionar um elemento aleatoriamente da LCR
        s ← s ∪ {e}
        atualizar a lista de candidatos
    fim-enquanto
fim-algoritmo
```

Figura 5.2 - Pseudocódigo do método para construir uma solução do PCTSP

A função de inserção para adicionar o vértice v_k entre os vértices v_i e v_j é

$$g(v_k) = c_{v_i v_j} + \gamma_{v_k} - c_{v_i v_k} - c_{v_k v_j} \tag{5.18}$$

sendo composta pelo custo da aresta (v_i, v_j), a penalidade do vértice v_k e os custos das arestas (v_i, v_k) e (v_k, v_j), respectivamente.

5.8.1 Meta-heurísticas

As meta-heurísticas implementadas para gerar soluções para o processo de agrupamento foram Algoritmo Genético (GA), Recozimento Simulado (SA), Pesquisa em Vizinhança Variável (VNS) e Busca Local Iterativa (ILS). Um gerador de soluções aleatórias também foi implementado, no qual todos os vértices são inseridos na rota em uma posição qualquer e alguns vértices são removidos aleatoriamente, sem violar a restrição de prêmio mínimo. Neste método são geradas 10000 soluções aleatórias que são enviadas a cada iteração para o processo de agrupamento. Os detalhes de cada meta-heurística são descritos a seguir.

5.8.1.1 Algoritmo Genético

O Algoritmo Genético (GA) (Subseção 2.2.1) começa a partir de uma população com μ indivíduos gerados aleatoriamente. A cada geração os pais que participarão do processo de evolução são selecionados por meio do operador de seleção por torneio.

O operador de cruzamento cíclico (OLIVER et al., 1987) foi implementado, visando preservar o tanto quanto possível a posição absoluta de vértices dos pais nos filhos. Esse operador previne que os filhos tenham vértices repetidos. Uma posição é sorteada no pai_1 para passar o seu respectivo vértice para a mesma posição no $filho_2$. Para garantir que não haja repetição de vértices nos filhos, identifica-se no pai_2, o vértice que foi selecionado no pai_1, e transfere-se esse vértice para o $filho_1$, na mesma posição em que foi encontrada no pai_2, formando assim um ciclo, no qual todos os vértices dentro desse ciclo são transferidos da mesma maneira. Os vértices que permanecem fora desse ciclo, são transferidos do pai_1 e pai_2, sem alterações, para as mesmas posições do $filho_1$ e $filho_2$, respectivamente. O cruzamento é realizando considerando apenas os vértices visitados. A Figura 5.3 exemplifica esse operador.

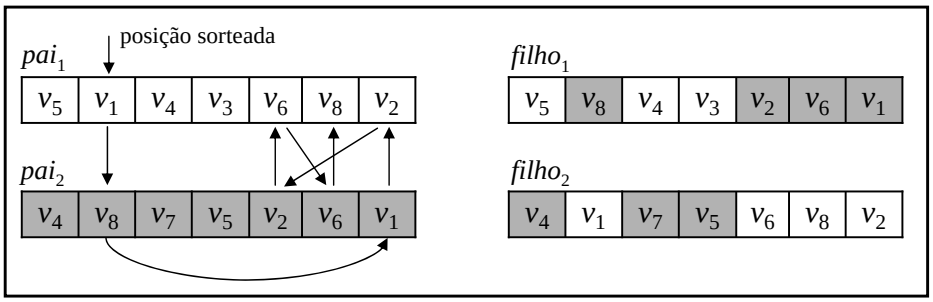

Figura 5.3 - Exemplo do cruzamento cíclico para o PCTSP.

No operador de mutação é verificada a probabilidade de mutação para cada vértice visitado e não visitado, realizando uma permutação simples entre dois vértices aleatoriamente. Se o vértice estiver na rota troca-se a sequência de visitas, e se o vértice não estiver na rota insere-se este na rota e retira-se outro vértice.

As probabilidades de cruzamento e mutação utilizadas foram $p_c = 0,95$ e $p_m = 0,001$, respectivamente. A população possui 80 indivíduos e o GA é executado por 1000 gerações. Então, a cada geração 12% dos filhos são enviados aleatoriamente para o processo de agrupamento.

5.8.1.2 Recozimento Simulado

O Recozimento Simulado (SA) (ver Subseção 2.2.2) parte de uma solução inicial gerada pelo método construtivo. A cada temperatura são executadas SA_{max} iterações, sendo que, em cada iteração um vizinho qualquer é gerado em uma das estruturas de vizinhança escolhida aleatoriamente.

Os parâmetros de controle do SA foram assim definidos: taxa de resfriamento $\alpha = 0,95$, número de iterações para cada temperatura $SA_{max} = 1000$ e temperatura inicial $T_0 = 10^6$. A cada 100 vizinhos gerados, a solução corrente do SA é enviada para o processo de agrupamento.

5.8.1.3 Pesquisa em Vizinhança Variável

A Pesquisa em Vizinhança Variável (VNS) (ver Subseção 2.2.3) também parte de uma solução inicial gerada pelo método construtivo, e explora a vizinhança da solução corrente por meio de movimentos cada vez mais distantes.

Neste trabalho foram definidas nove estruturas de vizinhança aninhadas, e uma solução vizinha é gerada aleatoriamente a uma distância φ da solução corrente. Essas estruturas são baseadas nos três movimentos definidos anteriormente. A distância está relacionada com o número de vértices visitados ou não visitados, dependendo do movimento realizado. Neste trabalho são gerados vizinhos a 10%, 20% e 30% da solução corrente ($\varphi \times n_r$ ou $\varphi \times n_u$), sendo $\varphi = \{0,1;\ 0,2 \text{ e } 0,3\}$.

As heurísticas Inserir-Vértice e Remover-Vértice (ver Subseção 5.8.3) são utilizadas para encontrar um ótimo local para o vizinho gerado, as quais buscam adicionar e retirar vértices da rota, possibilitando melhorar a solução. Essas heurísticas são

executadas enquanto houver soluções que melhorem o valor da função objetivo da solução corrente.

5.8.1.4 Busca Local Iterativa

A Busca Local Iterativa (ILS) (ver Subseção 2.2.4) inicializa a partir de uma solução gerada pelo método construtivo, na qual é aplicada uma busca local. A cada iteração é executada uma perturbação aleatória na solução corrente, por meio dos três movimentos definidos. O tamanho da perturbação é estabelecido aleatoriamente e depende do número de vértices do problema, variando de 25% a 75%. Cada movimento da perturbação é escolhido aleatoriamente entre os três movimentos possíveis.

Novamente, as heurísticas Inserir-Vértice e Retirar-Vértice (ver Subseção 5.8.3) são utilizadas para pesquisar a vizinhança de um solução.

5.8.2 Processo de Agrupamento

O processo de agrupamento compara uma nova solução s_k recebida com todos os centros dos clusters, e agrupa essa solução no cluster que possuir a menor distância entre a solução s_k e o centro c_j. Esse é considerado o cluster mais similar em relação a solução s_k.

Após realizar o agrupamento iterativo, o centro c_j é atualizado com parte da nova informação contida na solução s_k por meio do processo de assimilação, neste caso o método Reconexão por Caminho (PR) (ver Seção 3.4). Partindo do centro c_j para a solução s_k, caminhos são gerados e explorados na busca de soluções melhores. Para gerar estes caminhos, movimentos são selecionados por meio da troca de um vértice que esteja na rota do centro c_j por um vértice da solução s_k. A busca continua a partir da melhor solução em um movimento. Para diminuir o esforço computacional e também para não deslocar o centro para uma solução muito distante da posição atual foram pesquisados somente 20% dos movimentos possíveis. A Figura 5.4 apresenta um exemplo do PR aplicado ao PCTSP.

Uma análise do volume é executada toda vez que uma solução for agrupada a um cluster, verificando se o cluster pode ser considerado promissor, ou seja, se o volume v_j atingiu o limitante λ. Neste trabalho foi utilizado $\lambda = 15$. Caso o volume v_j seja maior ou igual a λ e o índice de ineficácia da heurística neste cluster seja baixo

($r_j < 5$) aplica-se uma busca local no centro c_j, objetivando intensificar a busca por soluções melhores por meio de uma análise da vizinhança do cluster. Porém, se o índice de ineficácia r_j for maior ou igual a cinco é necessário aplicar uma perturbação aleatória no centro c_j. Tal perturbação é realizada trocando aleatoriamente a posição de 30% dos vértices que estão na rota.

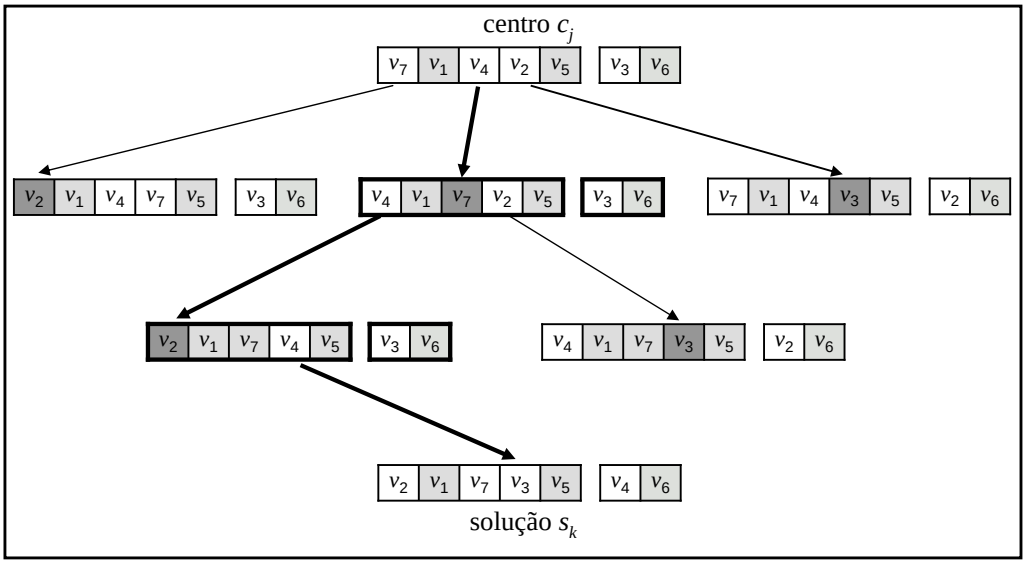

Figura 5.4 - Assimilação por caminho aplicada ao PCTSP.

5.8.3 Heurísticas de Busca Local

O método Descida em Vizinhança Variável (VND) (ver Seção 3.5) foi implementado como componente de busca local, sendo composto por três heurísticas de refinamento: 2-Opt (CROES, 1958), Remover-Vértice (GOMES et al., 2000) e Inserir-Vértice (GOMES et al., 2000). A cada iteração o VNS executa uma das heurísticas, retornando para a primeira heurística toda vez que a solução corrente for melhorada.

A heurística 2-Opt é baseada na troca entre pares de arestas do grafo. Removem-se duas arestas, quebrando o circuito em dois caminhos, e os reconecta da outra maneira possível. Os movimentos 2-Opt somente são realizados se as novas arestas incluídas possuírem custo menor que as removidas. O objetivo do 2-Opt é reduzir a distância entre os vértices visitados por meio da substituição de arestas de maior custo por outras de menor custo. A Figura 5.5 ilustra um exemplo de uma troca entre pares de arestas não consecutivas de uma rota, e a Figura 5.6 apresenta o pseudocódigo da heurística 2-Opt, no qual não é realizado chamadas à função objetivo.

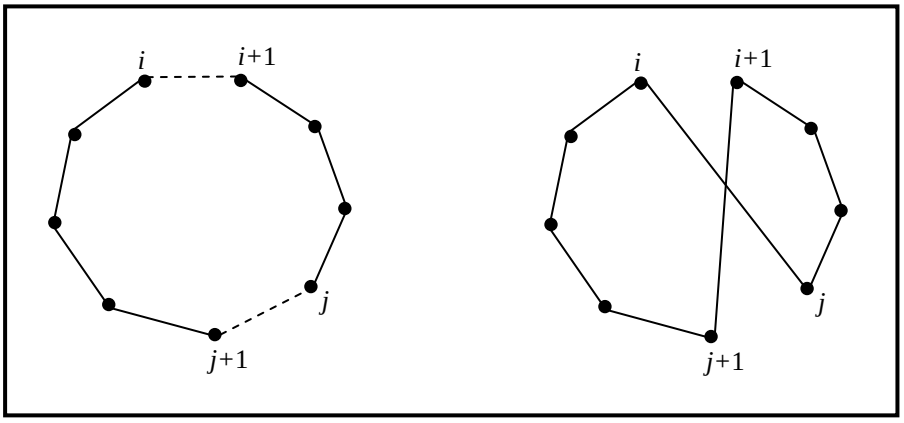

Figura 5.5 - Exemplo de um movimento 2-Opt.

---

algoritmo 2-Opt (s)
  para (i de 1 até n_r) **faça**
    $j \leftarrow i + 2$
    enquanto ((($j + 1$) *mod* n_r) $\neq i$) **faça**
      se ($c_{i\,i+1} + c_{j\,j+1} - c_{i\,j} - c_{i+1\,j+1} > 0$) **então**
        { trocar as arestas $(i,i+1)$ *e* $(j, j+1)$ *por* (i, j) *e* $(i+1, j+1)$ *}*
        inicio $\leftarrow (i + 1)$ *mod* n_r
        fim $\leftarrow j$
        se (inicio > fim) **então**
          $tam = n_r - inicio + fim + 1$
        senão $tam = fim - inicio + 1$
        $p1 \leftarrow inicio$ e $p2 \leftarrow fim$
        para (k de 1 até $tam/2$) **faça**
          troque os vértices $p1$ e $p2$ de posição
          $p1 \leftarrow p1 + 1$
          $p2 \leftarrow p2 - 1$
        fim-para
        $f(s) \leftarrow f(s) + c_{i\,i+1} + c_{j\,j+1} - c_{i\,j} - c_{i+1\,j+1}$
      fim-se
    fim-enquanto
  fim-para
fim-algoritmo

Figura 5.6 - Pseudocódigo da heurística 2-Opt.

A heurística Remover-Vértice consiste em analisar a economia de remoção de cada vértice visitado e, retirar da rota o vértice que tiver a maior economia positiva e não viole a restrição de prêmio mínimo. A economia de retirar um vértice k da rota é dada por $h(k) = c_{p_k k} + c_{k s_k} - c_{p_k s_k} - \gamma_k$, sendo p_k e s_k o predecessor e o sucessor de k, respectivamente. A Figura 5.7 apresenta o pseudocódigo dessa heurística.

algoritmo Remover-Vertice
  para (k de 1 até n_r) **faça**
    $L \leftarrow h(k) = c_{p_k k} + c_{k s_k} - c_{p_k s_k} - \gamma_k$
  fim-para
  selecione $k \equiv$ max { $h(k)$ para cada $k \in L$ }
  remova k da rota
  atualize $f(s)$
fim-algoritmo

Figura 5.7 - Pseudocódigo da heurística Remover-Vértices.

A heurística Inserir-Vértice consiste em analisar, para todas as posições possíveis da rota, a economia de inserir cada vértice não visitado. O vértice que tiver a maior economia positiva é inserido na rota. A economia de inserir um vértice k entre os vértices i e j é dada por $g(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj}$. A Figura 5.8 apresenta o pseudocódigo dessa heurística.

algoritmo Inserir-Vertice
  para (k de 1 até n_u) **faça**
    $L \leftarrow g(k) = \max\{c_{ij} + \gamma_k - c_{ik} - c_{kj}\}, \forall (i, j) \in rota$
  fim-para
  selecione $k \equiv$ max { $g(k)$ para cada $k \in L$ }
  insira k na rota entre os vértices i e j
  atualize $f(s)$
fim-algoritmo

Figura 5.8 - Pseudocódigo da heurística Inserir-Vértices.

Essas três heurísticas foram implementadas sem fazer chamada a função objetivo, o que reduziu consideravelmente o tempo computacional do CS.

Caso o VND encontre uma solução que seja melhor que o centro do cluster, este é atualizado com a nova solução encontrada. E, se a solução for a melhor solução encontrada até então neste cluster zera-se o índice de ineficácia, caso contrário, é somado uma unidade neste.

5.9 Resultados Computacionais

O CS para o PCTSP foi codificado em C++ e os experimentos foram conduzidos em um PC com processador Pentium 4, 2.6 GHz e memória de 1 GB. Os experimentos foram realizados com objetivo de validar a abordagem proposta, mostrando que o CS pode ser competitivo para resolução de problemas da classe TSPP.

O PCTSP não possui uma biblioteca pública de instâncias, sendo assim, foram selecionadas algumas instâncias clássicas do TSP obtidas na TSPLIB (REINELT, 1991) para serem transformadas em instâncias do PCTSP. Foram consideradas as instâncias TSP com coordenadas dos vértices. O número de vértices dessas instâncias são indicados em seus nomes (por exemplo, *burma14* possui 14 vértices). Para transformar essas em instâncias do PCTSP foram utilizadas as regras definidas em Dell'Amico et al. (1998) e Bérubé et al. (2008). Os prêmios e as penalidades são gerados aleatoriamente com distribuição uniforme. Os prêmios variam entre 1 e 100, e as penalidades entre 1 e W, sendo $W = \{100, 1000, 10000\}$. O valor do prêmio mínimo foi definido como $p_{min} = \sigma \sum_{i=1}^{n} p_i$, com $\sigma = \{0,2; 0,5; 0,8\}$. Essas instâncias estão disponíveis em `http://www.lac.inpe.br/~lorena/intancias.html`.

Para validar os resultados obtidos pelo CS, utilizou-se o *software* CPLEX versão 11.1.1 (ILOG, 2007) para resolver a formulação matemática do PCTSP apresentada na Seção 5.2. Porém, devido à complexidade do problema, o CPLEX consegue resolver de forma ótima apenas instâncias de pequeno porte. Todavia, este teste possibilita ter uma medida da qualidade das soluções providas pelo CS, uma vez que, não foi possível comparar o método com outras abordagens heurísticas da literatura.

As Tabelas 5.2 - 5.4 apresentam os resultados do CPLEX para as instâncias do PCTSP. As colunas das tabelas representam as soluções inteira e não inteira obtidas pelo CPLEX, o *gap* entre essas soluções e o tempo de execução do CPLEX. O tempo

máximo de execução do CPLEX foi definido como 10^5 segundos. O CPLEX consegue encontrar a solução ótima global para instâncias menores, porém, para instâncias acima de 100 vértices o CPLEX é interrompido sem conseguir fechar o *gap* entre os limitantes superior e inferior. O CPLEX não foi executado para as instâncias maiores, já que os *gaps* das instâncias *pr152* foram muito grandes (acima de 30%).

As Tabelas 5.5 - 5.19 apresentam os resultados computacionais das cinco versões do CS aplicado ao PCTSP. As tabelas possuem as seguintes colunas:

- **melhor**: melhor solução conhecida;
- **sol***: melhor solução encontrada pelo método;
- **desvio**: erro relativo entre a solução média e a melhor solução encontrada pelo método, ver Equação 4.7;
- **gap**: erro relativo entre a melhor solução encontrada pelo método e a melhor solução conhecida, ver Equação 4.8;
- **T***: tempo médio gasto pelo método para encontrar a melhor solução (em segundos);
- **T**: tempo médio de execução do método (em segundos);
- **delta**: erro relativo entre a melhor solução do CS e a melhor solução do método gerador de soluções, ver Equação 4.9.

Os valores em negrito mostram as instâncias nas quais o CS encontrou as melhores soluções conhecidas em termos de função objetivo. Para avaliar a robustez do método, o CS foi executado 20 vezes para cada instância.

Observando as tabelas nota-se que o CS encontrou todas as soluções ótimas para as instâncias nas quais essas são conhecidas. Para as instâncias nas quais o CPLEX não conseguiu fechar o *gap*, o CS encontrou soluções melhores que as soluções inteiras obtidas pelo CPLEX. Observa-se também que o tempo computacional do CS foi competitivo, resolvendo o PCTSP em alguns segundos para as instâncias menores, e no máximo em poucas horas para as instâncias maiores.

Todas as versões do CS obtiveram bons resultados, encontrando soluções próximas das melhores soluções conhecidas em quase todas as instâncias. O CS também se

mostrou robusto, obtendo desvios pequenos entre a melhor solução encontrada e a solução média. Entretanto, a versão do CS com a meta-heurística VNS como gerador de soluções para o processo de agrupamento obteve os melhores resultados para o PCTSP com relação à qualidade da solução.

A instância *pr152* com penalidade baixa ($\gamma_i \in [1, 100]$) apresentou um comportamento distinto. A meta-heurística VNS encontra soluções muito boas para essa instância, sendo que, os demais componentes do CS e as outras meta-heurísticas não foram capazes de obter soluções próximas à estas. Este fato resulta em *gaps* muito grandes para essa instância.

A convergência das versões do CS com o SA e o VNS foram muito boas, encontrando a melhor solução em aproximadamente 10% do tempo de execução.

A melhora dos resultados das meta-heurísticas pelos demais componentes do CS também pode ser observada nestas tabelas por meio da análise dos valores da coluna *delta*, notando que os valores médios variam de 12% a mais de 40%. Mostrando que o CS melhora significativamente os resultados obtidos pelas meta-heurísticas quando aplicadas isoladamente.

Com relação as variações das instâncias, pode-se observar que quanto maior o valor máximo da penalidade maior é o tempo de execução do CS, o que pode ser explicado pelo fato de um número maior de vértices serem visitados quando a penalidade é aumentada. Todavia, apesar dos tempos computacionais serem maiores para as instâncias com valores altos de penalidades, essas instâncias são mais fáceis de resolver, pois é mais simples descobrir quais vértices não serão visitados e, assim, o problema se aproxima do TSP clássico.

A mesma observação se aplica a variação do valor de prêmio mínimo. Quanto maior for o valor de prêmio mínimo maior é o tempo computacional do CS, em razão do número de vértices visitados ser maior. Porém, em relação à dificuldade de resolução não podem ser observados grandes alterações causadas pelo valor de prêmio mínimo.

O valor de prêmio mínimo praticamente não tem interferência no valor da função objetivo das instâncias com penalidades altas, os vértices que são visitados são os mesmos não importando a quantidade de prêmios que necessita ser coletada. Em alguns casos todos os vértices do problema são visitados.

Por causa do grande número de tabelas, fez-se um resumo dos resultados do CS que é apresentado na Tabela 5.1. As colunas desta tabela representam o número de instâncias nas quais o CS encontrou a melhor solução conhecida, a qualidade média das soluções do CS, a robustez do CS em relação à melhor solução encontrada e a solução média, o tempo médio que o CS gastou para convergir para a melhor solução encontrada e o tempo médio de execução do CS. A última coluna mostra a melhora do CS em relação à solução obtida pelo gerador de soluções para o processo de agrupamento.

Tabela 5.1 - PCTSP: Resumo dos resultados computacionais do CS.

		melhor solução	**Qualidade (gap)**	**Robustez (desvio)**	**Tempo Melhor**	**Tempo Total**	**Melhora (delta)**
$\lambda \in$ **[1,100]**	**CS-GA**	18/45	1,93	0,90	355,57	457,14	39,04
	CS-SA	19/45	1,53	0,71	66,14	407,18	13,36
	CS-VNS	41/45	0,03	0,55	156,66	635,05	12,37
	CS-ILS	20/45	1,80	0,90	463,03	693,43	40,54
	CS-Aleatório	20/45	1,42	1,01	110,09	489,20	441,09

		ótimo global	**Qualidade (gap)**	**Robustez (desvio)**	**Tempo Melhor**	**Tempo Total**	**Melhora (delta)**
$\lambda \in$ **[1,1000]**	**CS-GA**	28/45	0,29	0,26	538,60	745,03	38,28
	CS-SA	24/45	0,30	0,36	42,06	743,86	17,98
	CS-VNS	32/45	0,26	0,33	76,85	853,13	14,37
	CS-ILS	35/45	0,18	0,32	681,12	974,58	16,01
	CS-Aleatório	27/45	0,41	0,20	113,61	664,71	625,05

		ótimo global	**Qualidade (gap)**	**Robustez (desvio)**	**Tempo Melhor**	**Tempo Total**	**Melhora (delta)**
$\lambda \in$ **[1,10000]**	**CS-GA**	31/45	0,13	0,19	660,04	940,47	41,13
	CS-SA	28/45	0,30	0,18	90,67	1145,60	35,78
	CS-VNS	34/45	0,08	0,23	194,68	1028,90	15,36
	CS-ILS	39/45	0,03	0,20	746,61	1077,94	20,76
	CS-Aleatório	29/45	0,30	0,17	139,92	774,27	3692,13

Tabela 5.2 - PCTSP: $\gamma_i \in [1, 100]$. Resultados Computacionais do CPLEX.

		CPLEX			
instância	σ	**solução inteira**	**solução não inteira**	**gap-CPLEX**	**tempo**
burma14		**797**	797,00	0,00	0,03
ulysses16		**1244**	1244,00	0,00	0,14
ulysses22		**1699**	1699,00	0,00	2,64
berlin52		**2189**	2189,00	0,00	55,83
st70		665	654,44	1,59	100000,06
pr76		**20390**	20390,00	0,00	1697,31
kroA100		8004	7513,45	6,13	100000,05
kroB100	0,2	7600	7037,12	7,41	100000,08
bier127		12395	12286,34	0,88	100000,56
pr152		26310	13486,57	48,74	76157,86
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-
burma14		**1438**	1438,00	0,00	1,38
ulysses16		**2360**	2360,00	0,00	3,80
ulysses22		**2144**	2144,00	0,00	7,33
berlin52		**3147**	3147,00	0,00	141,99
st70		665	653,26	1,77	100000,09
pr76		**39144**	39144,00	0,00	21480,11
kroA100		10826	10594,37	2,14	100000,06
kroB100	0,5	11606	9853,15	15,10	100000,13
bier127		28256	24685,78	12,64	100000,11
pr152		46211	23981,17	48,11	100000,28
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-
burma14		**2026**	2026,00	0,00	0,72
ulysses16		**4079**	4079,00	0,00	1,67
ulysses22		**3436**	3436,00	0,00	6,03
berlin52		**5322**	5322,00	0,00	4187,19
st70		665	655,89	1,37	100000,03
pr76		**64205**	64205,00	0,00	26761,97
kroA100		15247	14899,06	2,28	100000,22
kroB100	0,8	15478	14199,64	8,26	100000,14
bier127		64601	47685,29	26,18	100000,25
pr152		66883	38331,74	42,69	100000,17
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-

Tabela 5.3 - PCTSP: $\gamma_i \in [1, 1000]$. Resultados Computacionais do CPLEX.

		CPLEX			
instância	σ	**solução inteira**	**solução não inteira**	**gap-CPLEX**	**tempo**
burma14		**2313**	2313,00	0,00	0,14
ulysses16		**4909**	4909,00	0,00	0,41
ulysses22		**4866**	4866,00	0,00	3,25
berlin52		**7374**	7374,00	0,00	1962,67
st70		**675**	675,00	0,00	36784,63
pr76		**45853**	45853,00	0,00	1157,19
kroA100		20079	19623,03	2,27	100000,09
kroB100	0,2	20769	19350,80	6,83	100000,14
bier127		53141	49881,23	6,13	100000,31
pr152		73908	44690,29	39,53	100000,28
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-
burma14		**2313**	2313,00	0,00	0,16
ulysses16		**4909**	4909,00	0,00	0,39
ulysses22		**4866**	4866,00	0,00	3,30
berlin52		**7374**	7374,00	0,00	6146,19
st70		**675**	675,00	0,00	36816,16
pr76		**57252**	57252,00	0,00	9233,47
kroA100		20079	19682,32	1,98	100000,08
kroB100	0,5	20783	19292,42	7,17	100000,09
bier127		55930	50895,79	9,00	100000,17
pr152		73547	44824,43	39,05	100000,09
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-
burma14		**2321**	2321,00	0,00	0,16
ulysses16		**5106**	5106,00	0,00	0,33
ulysses22		**4887**	4887,00	0,00	2,58
berlin52		**7374**	7374,00	0,00	399,09
st70		**675**	675,00	0,00	32189,75
pr76		73272	72603,15	0,91	100000,05
kroA100		20202	19716,88	2,40	100000,06
kroB100	0,8	21539	19108,96	11,28	100000,08
bier127		71645	61781,94	13,77	100000,22
pr152		70041	47408,89	32,31	100000,50
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-

Tabela 5.4 - PCTSP: $\gamma_i \in [1, 10000]$. Resultados Computacionais do CPLEX.

		CPLEX			
instância	σ	**solução inteira**	**solução não inteira**	**gap-CPLEX**	**tempo**
burma14		**3050**	3050,00	0,00	0,17
ulysses16		**6859**	6859,00	0,00	0,31
ulysses22		**7013**	7013,00	0,00	0,69
berlin52		**7542**	7542,00	0,00	53,98
st70		**675**	675,00	0,00	74371,06
pr76		**97269**	97269,00	0,00	66271,91
kroA100		21296	20757,91	2,53	100000,13
kroB100	0,2	21702	21306,49	1,82	100000,05
bier127		111746	105838,49	5,29	100000,42
pr152		86939	56471,03	35,05	100000,52
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-
burma14		**3050**	3050,00	0,00	0,17
ulysses16		**6859**	6859,00	0,00	0,30
ulysses22		**7013**	7013,00	0,00	0,69
berlin52		**7542**	7542,00	0,00	53,95
st70		**675**	675,00	0,00	74331,28
pr76		**97269**	97269,00	0,00	65426,55
kroA100		21296	20640,22	3,08	100000,76
kroB100	0,5	21853	21246,72	2,77	100000,42
bier127		112362	105814,39	5,82	100000,26
pr152		87674	56430,68	35,63	100000,31
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-
burma14		**3050**	3050,00	0,00	0,19
ulysses16		**6859**	6859,00	0,00	0,30
ulysses22		**7013**	7013,00	0,00	0,70
berlin52		**7542**	7542,00	0,00	45,70
st70		**675**	675,00	0,00	29629,18
pr76		97269	96913,80	0,36	100000,20
kroA100		21673	20458,95	5,60	100000,32
kroB100	0,8	21853	21246,73	2,77	100000,38
bier127		117339	106062,82	9,61	100000,57
pr152		86373	56443,68	34,65	100000,34
gr202		-	-	-	-
tsp225		-	-	-	-
a280		-	-	-	-
lin318		-	-	-	-
gr431		-	-	-	-

Tabela 5.5 - PCTSP: $\gamma_i \in [1, 100]$. Resultados Computacionais do CS-GA.

			CS-GA					GA				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		797	**797**	0,00	0,00	0,00	1,90	1162	0,00	45,80	1,39	45,80
ulysses16		1244	**1244**	0,00	0,00	0,00	2,10	1700	0,00	36,66	1,48	36,66
ulysses22		1699	**1699**	0,00	0,00	0,57	2,55	1941	2,38	14,24	1,58	14,24
berlin52		2189	**2189**	0,00	0,00	1,19	10,46	2458	25,56	12,29	2,63	12,29
st70		665	**665**	0,00	0,00	24,03	89,61	763	27,80	14,74	11,62	14,74
pr76		20390	**20390**	3,59	0,00	6,76	13,25	28707	23,59	40,79	3,71	40,79
kroA100		8002	**8002**	0,11	0,00	7,87	25,35	10086	15,53	26,04	4,64	26,04
kroB100	0,2	7300	**7300**	0,52	0,00	12,28	27,87	7871	27,86	7,82	4,77	7,82
bier127		12395	**12395**	1,79	0,00	21,68	35,42	17776	35,11	43,41	5,96	43,41
pr152		22511	26421	1,85	17,37	41,51	63,89	34803	32,96	54,60	7,61	31,72
gr202		14724	14759	0,11	0,24	72,75	100,96	18545	5,45	25,95	12,45	25,65
tsp225		3732	3736	0,47	0,11	518,52	799,60	5470	29,10	46,57	18,23	46,41
a280		2562	2567	0,37	0,20	1106,11	1682,08	4262	29,94	66,35	29,76	66,03
lin318		18155	20232	1,98	11,44	248,68	388,42	27726	9,50	52,72	33,15	37,04
gr431		27817	28082	0,85	0,95	335,81	502,01	33190	68,87	19,32	77,16	18,19
burma14		1438	**1438**	0,10	0,00	1,90	2,34	1541	0,00	7,16	1,54	7,16
ulysses16		2360	**2360**	0,00	0,00	0,88	3,12	2973	1,24	25,97	1,67	25,97
ulysses22		2144	**2144**	0,00	0,00	1,04	3,88	2188	3,03	2,05	1,86	2,05
berlin52		3147	**3147**	0,23	0,00	10,42	21,35	3571	20,73	13,47	4,41	13,47
st70		665	**665**	0,00	0,00	16,19	78,16	819	16,83	23,16	6,26	23,16
pr76		39144	39551	2,50	1,04	22,49	32,88	52106	21,52	33,11	7,24	31,74
kroA100		10813	10842	2,20	0,27	46,57	67,13	15887	10,63	46,92	10,52	46,53
kroB100	0,5	11020	11206	0,93	1,69	39,38	70,53	14943	9,82	35,60	10,90	33,35
bier127		26688	27583	1,29	3,35	76,82	102,07	47639	16,97	78,50	15,81	72,71
pr152		38350	46181	1,86	20,42	126,14	178,30	61509	20,31	60,39	21,06	33,19
gr202		18386	18590	0,79	1,11	249,26	307,07	22905	13,16	24,58	31,14	23,21
tsp225		3719	3726	0,96	0,19	608,47	923,44	6975	8,21	87,55	38,89	87,20
a280		2566	2578	0,43	0,47	1470,82	1940,93	4587	32,97	78,76	61,10	77,93
lin318		23381	23601	1,57	0,94	763,47	937,53	37527	29,59	60,50	77,68	59,01
gr431		44515	49136	5,29	10,38	1457,02	1689,03	88284	25,22	98,32	156,21	79,67
burma14		2026	**2026**	0,00	0,00	1,04	3,51	2026	0,00	0,00	1,76	0,00
ulysses16		4079	**4079**	0,00	0,00	1,17	4,04	4079	0,63	0,00	1,97	0,00
ulysses22		3436	**3436**	0,00	0,00	1,70	5,70	3436	1,27	0,00	2,48	0,00
berlin52		5322	5380	1,51	1,09	21,09	36,90	6305	13,79	18,47	7,20	17,19
st70		665	**665**	0,00	0,00	10,88	58,53	879	13,40	32,18	3,38	32,18
pr76		64205	66149	1,30	3,03	41,09	62,75	88600	20,98	38,00	13,31	33,94
kroA100		15242	15385	0,80	0,94	94,14	128,54	21299	25,14	39,74	21,34	38,44
kroB100	0,8	15400	15505	0,35	0,68	82,91	124,64	20224	29,53	31,32	21,39	30,44
bier127		55731	56962	1,59	2,21	185,99	224,60	87164	17,50	56,40	32,54	53,02
pr152		60445	61389	1,31	1,56	247,17	353,93	101544	54,10	67,99	45,00	65,41
gr202		24119	24586	0,96	1,94	554,72	622,84	34530	23,60	43,17	63,72	40,45
tsp225		3723	3737	0,53	0,38	785,55	1027,34	6901	18,73	85,36	78,98	84,67
a280		2558	2579	0,33	0,82	1886,86	1952,71	4309	58,81	68,45	120,13	67,08
lin318		31166	31696	0,66	1,70	1255,60	1962,10	61577	38,36	97,58	154,55	94,27
gr431		82458	84558	1,38	2,55	3542,18	3899,97	183002	17,80	121,93	285,49	116,42
média			**0,90**	**1,93**	**355,57**	**457,14**		**19,95**	**41,87**	**33,68**	**39,04**	

Tabela 5.6 - PCTSP: $\gamma_i \in [1, 1000]$. Resultados Computacionais do CS-GA.

			CS-GA					GA				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		2313	**2313**	0,00	0,00	0,63	2,89	2313	0,00	0,00	1,49	0,00
ulysses16		4909	**4909**	0,00	0,00	0,68	3,68	5106	0,24	4,01	1,52	4,01
ulysses22		4866	**4866**	0,00	0,00	0,72	5,52	4887	2,81	0,43	1,66	0,43
berlin52		7374	**7374**	0,00	0,00	3,59	30,12	8059	7,94	9,29	2,80	9,29
st70		675	**675**	0,00	0,00	8,15	59,16	885	19,95	31,11	3,49	31,11
pr76		45853	**45853**	0,00	0,00	6,04	21,38	47695	43,51	4,02	3,60	4,02
kroA100		20079	**20079**	0,00	0,00	5,32	111,28	25583	14,85	27,41	5,16	27,41
kroB100	0,2	20442	**20442**	0,04	0,00	43,71	128,98	26898	16,74	31,58	5,27	31,58
bier127		52763	**52763**	0,04	0,00	34,50	89,00	71547	18,10	35,60	6,96	35,60
pr152		62448	62460	0,12	0,02	135,41	336,76	86914	24,17	39,18	10,03	39,15
gr202		32913	32931	0,44	0,05	437,83	605,79	46364	19,90	40,87	18,30	40,79
tsp225		3939	3958	0,24	0,48	678,82	950,98	7392	38,17	87,66	24,27	86,76
a280		2586	2606	0,40	0,77	1017,52	1744,40	5152	35,75	99,23	41,06	97,70
lin318		40543	**40543**	0,26	0,00	1448,89	2249,44	67395	26,72	66,23	57,72	66,23
gr431		113316	**113316**	0,72	0,00	2623,29	3184,03	169073	5,81	49,20	139,04	49,20
burma14		2313	**2313**	0,00	0,00	0,78	3,15	2313	0,00	0,00	1,53	0,00
ulysses16		4909	**4909**	0,00	0,00	0,86	4,05	5106	1,07	4,01	1,67	4,01
ulysses22		4866	**4866**	0,00	0,00	1,30	6,31	4887	2,53	0,43	2,00	0,43
berlin52		7374	**7374**	0,00	0,00	5,84	41,01	8119	4,08	10,10	4,39	10,10
st70		675	**675**	0,00	0,00	19,58	79,22	854	18,96	26,52	6,24	26,52
pr76		57252	**57252**	0,06	0,00	19,66	40,47	72913	15,96	27,35	7,13	27,35
kroA100		20079	**20079**	0,00	0,00	15,43	159,77	26438	16,15	31,67	10,86	31,67
kroB100	0,5	20442	**20442**	0,01	0,00	92,54	162,99	27043	18,73	32,29	10,99	32,29
bier127		54325	54505	0,22	0,33	98,44	138,51	77729	9,68	43,08	15,93	42,61
pr152		65328	68227	1,09	4,44	267,04	395,08	96873	15,11	48,29	21,93	41,99
gr202		32813	32914	0,79	0,31	498,02	710,60	49403	6,08	50,56	33,42	50,10
tsp225		3925	3964	0,67	0,99	762,94	1130,48	7805	15,73	98,85	41,85	96,90
a280		2587	2605	0,60	0,70	1096,65	2011,09	4491	70,81	73,60	66,11	72,40
lin318		40410	40635	0,41	0,56	2192,55	2553,50	76219	25,31	88,61	87,57	87,57
gr431		113549	114231	0,31	0,60	2885,57	3720,41	174794	8,36	53,94	181,77	53,02
burma14		2321	**2321**	0,00	0,00	1,02	3,61	2321	0,00	0,00	1,81	0,00
ulysses16		5106	**5106**	0,00	0,00	1,19	4,62	5106	0,44	0,00	2,00	0,00
ulysses22		4887	**4887**	0,00	0,00	1,63	6,77	4887	1,03	0,00	2,53	0,00
berlin52		7374	**7374**	0,00	0,00	6,89	45,05	8098	5,34	9,82	7,19	9,82
st70		675	**675**	0,00	0,00	31,83	91,26	787	41,83	16,59	11,58	16,59
pr76		73272	**73272**	1,09	0,00	56,40	70,68	97559	17,53	33,15	13,21	33,15
kroA100		20079	**20079**	0,00	0,00	26,87	174,01	25831	28,50	28,65	21,09	28,65
kroB100	0,8	20442	**20442**	0,04	0,00	91,50	178,54	29354	15,64	43,60	21,42	43,60
bier127		68902	69472	1,52	0,83	169,26	243,58	92478	23,87	34,22	32,42	33,12
pr152		69516	69554	0,19	0,05	316,17	444,74	102093	30,65	46,86	45,35	46,78
gr202		32867	33145	0,10	0,85	546,40	806,43	44846	28,71	36,45	64,61	35,30
tsp225		3952	3990	0,81	0,96	865,55	1252,24	7316	6,76	85,12	79,47	83,36
a280		2588	2607	0,31	0,73	1601,10	2160,14	6272	42,42	142,35	121,58	140,58
lin318		40265	40466	0,86	0,50	2124,90	2735,44	70421	19,42	74,89	156,85	74,03
gr431		115874	**115874**	0,54	0,00	3993,92	4629,17	205299	42,88	77,17	290,37	77,17
média				**0,26**	**0,29**	**538,60**	**745,03**		**17,96**	**38,76**	**37,49**	**38,28**

Tabela 5.7 - PCTSP: $\gamma_i \in [1, 10000]$. Resultados Computacionais do CS-GA.

			CS-GA					GA				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		3050	**3050**	0,00	0,00	0,65	3,47	3050	0,22	0,00	1,42	0,00
ulysses16		6859	**6859**	0,00	0,00	0,69	4,11	6859	0,94	0,00	1,43	0,00
ulysses22		7013	**7013**	0,00	0,00	0,71	6,78	7043	2,13	0,43	1,73	0,43
berlin52		7542	**7542**	0,00	0,00	1,43	30,92	8302	13,25	10,08	2,75	10,08
st70		675	**675**	0,00	0,00	10,78	62,15	794	39,77	17,63	3,68	17,63
pr76		97269	**97269**	0,00	0,00	9,05	69,69	116817	20,78	20,10	3,87	20,10
kroA100		21282	**21282**	0,00	0,00	35,91	137,70	35871	24,30	68,55	5,27	68,55
kroB100	0,2	21697	**21697**	0,05	0,00	69,91	146,53	33263	27,97	53,31	5,22	53,31
bier127		108605	**108605**	0,02	0,00	111,88	234,69	144209	16,23	32,78	7,62	32,78
pr152		73682	**73682**	0,02	0,00	161,44	420,75	107218	54,46	45,51	9,59	45,51
gr202		38743	38777	0,65	0,09	424,03	714,76	57225	27,13	47,70	18,90	47,57
tsp225		3942	3966	0,56	0,61	690,95	950,67	6803	37,95	72,58	23,95	71,53
a280		2592	2593	0,54	0,04	1262,84	1748,36	5506	31,20	112,42	41,50	112,34
lin318		42361	**42361**	0,78	0,00	1723,13	2559,37	77080	89,09	81,96	58,10	81,96
gr431		168632	169204	0,58	0,34	3614,12	5556,84	271694	26,81	61,12	144,67	60,57
burma14		3050	**3050**	0,00	0,00	0,80	3,74	3050	0,80	0,00	1,58	0,00
ulysses16		6859	**6859**	0,00	0,00	0,88	4,78	6870	0,66	0,16	1,63	0,16
ulysses22		7013	**7013**	0,00	0,00	1,07	7,82	7092	2,25	1,13	2,00	1,13
berlin52		7542	**7542**	0,00	0,00	3,53	42,56	8829	7,53	17,06	4,39	17,06
st70		675	**675**	0,00	0,00	19,21	83,76	854	20,63	26,52	6,52	26,52
pr76		97269	**97269**	0,00	0,00	26,97	87,97	118575	13,83	21,90	7,26	21,90
kroA100		21282	**21282**	0,00	0,00	53,92	169,88	26417	49,51	24,13	10,84	24,13
kroB100	0,5	21697	**21697**	0,08	0,00	110,80	184,51	33765	26,25	55,62	10,95	55,62
bier127		108605	**108605**	0,10	0,00	135,03	278,77	144356	14,05	32,92	16,47	32,92
pr152		73682	**73682**	0,11	0,00	277,02	521,84	100466	77,48	36,35	21,86	36,35
gr202		38882	39038	0,37	0,40	573,93	844,84	61565	9,76	58,34	33,56	57,71
tsp225		3952	3985	0,24	0,84	779,16	1110,09	5728	40,53	44,94	41,64	43,74
a280		2593	2602	0,87	0,35	1501,67	1968,97	4572	33,84	76,32	65,89	75,71
lin318		42484	42701	0,20	0,51	2010,47	2775,86	106142	23,21	149,84	87,47	148,57
gr431		168221	169244	0,64	0,61	5525,47	6147,77	328775	24,54	95,44	183,05	94,26
burma14		3050	**3050**	0,00	0,00	1,03	4,04	3050	0,00	0,00	1,74	0,00
ulysses16		6859	**6859**	0,00	0,00	1,21	5,26	6859	0,69	0,00	1,92	0,00
ulysses22		7013	**7013**	0,00	0,00	1,61	8,36	7065	1,47	0,74	2,52	0,74
berlin52		7542	**7542**	0,00	0,00	6,64	48,15	8601	11,67	14,04	7,22	14,04
st70		675	**675**	0,00	0,00	30,52	91,04	869	19,62	28,74	11,66	28,74
pr76		97269	**97269**	0,02	0,00	59,82	100,90	125268	18,34	28,79	13,26	28,79
kroA100		21282	**21282**	0,00	0,00	104,09	200,23	31323	31,90	47,18	21,25	47,18
kroB100	0,8	21697	**21697**	0,02	0,00	114,25	208,68	29380	37,11	35,41	21,74	35,41
bier127		108605	**108605**	0,14	0,00	215,44	346,58	147715	9,38	36,01	32,98	36,01
pr152		73682	**73682**	0,04	0,00	303,38	560,47	105796	53,83	43,58	45,26	43,58
gr202		38855	39014	0,11	0,41	640,13	941,10	54896	32,25	41,28	64,68	40,71
tsp225		3955	3970	0,66	0,38	828,02	1236,24	6430	58,41	62,58	79,23	61,96
a280		2593	2621	0,12	1,08	1363,08	2129,56	5136	45,05	98,07	121,86	95,96
lin318		42373	42430	0,78	0,13	2428,69	3042,57	84455	37,21	99,31	156,99	99,05
gr431		168765	168879	0,99	0,07	4466,44	6518,10	271339	17,71	60,78	291,98	60,67
média				**0,19**	**0,13**	**660,04**	**940,47**		**25,15**	**41,36**	**37,76**	**41,13**

Tabela 5.8 - PCTSP: $\gamma_i \in [1, 100]$. Resultados Computacionais do CS-SA.

			CS-SA					SA				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		797	**797**	0,00	0,00	0,07	6,08	797	0,00	0,00	5,68	0,00
ulysses16		1244	**1244**	0,00	0,00	0,00	6,55	1244	0,00	0,00	6,12	0,00
ulysses22		1699	**1699**	0,00	0,00	0,46	6,85	1699	0,00	0,00	6,17	0,00
berlin52		2189	**2189**	0,00	0,00	1,25	11,56	2189	0,00	0,00	7,82	0,00
st70		665	**665**	0,00	0,00	1,64	63,33	693	3,19	4,21	7,80	4,21
pr76		20390	21226	0,31	4,10	1,83	10,28	21722	1,59	6,53	7,10	2,34
kroA100		8002	**8002**	0,13	0,00	3,12	18,26	8086	0,99	1,05	9,78	1,05
kroB100	0,2	7300	**7300**	0,65	0,00	2,49	19,33	7630	1,20	4,52	9,95	4,52
bier127		12395	**12395**	0,08	0,00	3,02	22,62	13031	2,21	5,13	10,99	5,13
pr152		22511	26421	0,00	17,37	4,21	34,71	27110	6,67	20,43	12,70	2,61
gr202		14724	14735	0,34	0,07	5,71	58,35	15018	0,50	2,00	17,80	1,92
tsp225		3732	3763	0,33	0,83	7,55	454,31	4192	2,31	12,33	21,26	11,40
a280		2562	2591	0,40	1,13	11,65	1020,21	3258	2,20	27,17	27,78	25,74
lin318		18155	19763	1,21	8,86	13,23	190,60	20757	2,69	14,33	28,31	5,03
gr431		27817	28067	0,54	0,90	24,08	275,66	30080	0,93	8,14	39,41	7,17
burma14		1438	**1438**	0,00	0,00	0,59	6,93	1438	0,00	0,00	6,17	0,00
ulysses16		2360	**2360**	0,00	0,00	0,89	7,73	2360	0,00	0,00	6,32	0,00
ulysses22		2144	**2144**	0,00	0,00	1,09	8,56	2144	0,00	0,00	6,47	0,00
berlin52		3147	**3147**	0,17	0,00	3,79	22,56	3164	2,92	0,54	9,38	0,54
st70		665	**665**	0,00	0,00	5,11	81,57	694	4,50	4,36	11,05	4,36
pr76		39144	**39144**	2,45	0,00	6,80	25,77	40788	3,50	4,20	10,87	4,20
kroA100		10813	10931	2,53	1,09	9,19	55,47	11363	5,67	5,09	16,95	3,95
kroB100	0,5	11020	11252	1,27	2,11	11,80	56,47	12500	3,71	13,43	16,55	11,09
bier127		26688	27572	1,00	3,31	15,56	79,92	31120	6,52	16,61	21,61	12,87
pr152		38350	39859	1,90	3,93	34,21	137,34	45497	13,07	18,64	29,63	14,14
gr202		18386	18770	0,36	2,09	33,80	250,13	20256	2,79	10,17	39,35	7,92
tsp225		3719	3739	0,89	0,54	42,02	721,99	4337	1,88	16,62	47,28	15,99
a280		2566	2574	0,71	0,31	66,08	1502,09	3323	4,50	29,50	66,75	29,10
lin318		23381	23726	0,97	1,48	82,57	772,76	31513	3,86	34,78	79,58	32,82
gr431		44515	47088	3,26	5,78	484,08	1404,91	68363	4,44	53,57	131,81	45,18
burma14		2026	**2026**	0,00	0,00	1,07	8,28	2026	0,00	0,00	6,26	0,00
ulysses16		4079	**4079**	0,00	0,00	1,22	9,11	4079	0,00	0,00	6,63	0,00
ulysses22		3436	**3436**	0,00	0,00	1,70	11,23	3436	0,38	0,00	7,10	0,00
berlin52		5322	**5322**	1,08	0,00	13,60	44,70	5322	4,70	0,00	13,20	0,00
st70		665	**665**	0,00	0,00	11,32	105,79	699	3,40	5,11	17,61	5,11
pr76		64205	64880	3,22	1,05	13,36	64,43	69148	11,55	7,70	17,52	6,58
kroA100		15242	15415	0,69	1,14	21,72	130,67	17327	7,43	13,68	28,19	12,40
kroB100	0,8	15400	15537	0,18	0,89	21,64	128,70	17297	7,81	12,32	28,00	11,33
bier127		55731	57351	1,20	2,91	38,13	226,01	72645	6,31	30,35	39,90	26,67
pr152		60445	61858	0,80	2,34	58,96	347,87	74564	21,81	23,36	56,08	20,54
gr202		24119	24187	2,06	0,28	161,20	636,63	30582	3,37	26,80	76,91	26,44
tsp225		3723	3731	0,87	0,21	103,96	1101,23	4715	0,81	26,65	93,34	26,37
a280		2558	2593	0,15	1,37	167,16	2157,75	3841	5,66	50,16	137,05	48,13
lin318		31166	31869	0,80	2,26	221,95	2032,01	52569	4,47	68,67	171,67	64,95
gr431		82458	84375	1,47	2,32	1261,21	3985,63	168373	7,25	104,19	298,12	99,55
média				**0,71**	**1,53**	**66,14**	**407,18**		**3,71**	**15,16**	**38,13**	**13,36**

Tabela 5.9 - PCTSP: $\gamma_i \in [1, 1000]$. Resultados Computacionais do CS-SA.

			CS-SA					SA				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		2313	**2313**	0,00	0,00	0,68	7,61	2313	0,00	0,00	5,94	0,00
ulysses16		4909	**4909**	0,00	0,00	0,72	8,56	4909	0,00	0,00	6,21	0,00
ulysses22		4866	**4866**	0,00	0,00	0,75	9,55	4866	0,00	0,00	6,35	0,00
berlin52		7374	**7374**	0,00	0,00	1,30	39,78	7397	1,70	0,31	8,70	0,31
st70		675	**675**	0,00	0,00	1,59	96,29	759	3,83	12,44	9,04	12,44
pr76		45853	**45853**	0,00	0,00	1,99	13,70	45853	0,86	0,00	7,58	0,00
kroA100		20079	**20079**	0,00	0,00	2,30	118,80	20202	8,85	0,61	11,20	0,61
kroB100	0,2	20442	20449	0,12	0,03	7,56	100,36	21129	20,34	3,36	11,61	3,33
bier127		52763	**52763**	0,00	0,00	4,95	46,41	52856	0,23	0,18	11,51	0,18
pr152		62448	62460	0,05	0,02	4,08	167,75	64902	1,51	3,93	13,20	3,91
gr202		32913	33185	0,34	0,83	38,15	602,84	34727	3,15	5,51	21,22	4,65
tsp225		3939	3988	0,78	1,24	7,82	1269,14	6230	2,98	58,16	24,08	56,22
a280		2586	2595	1,24	0,35	12,48	2236,30	5109	4,77	97,56	30,47	96,88
lin318		40543	40747	0,41	0,50	16,08	1870,30	47308	2,90	16,69	33,34	16,10
gr431		113316	114373	0,17	0,93	27,66	1663,00	138951	0,78	22,62	42,90	21,49
burma14		2313	**2313**	0,00	0,00	0,86	7,86	2313	0,00	0,00	6,12	0,00
ulysses16		4909	**4909**	0,00	0,00	0,91	8,93	4909	0,00	0,00	6,38	0,00
ulysses22		4866	**4866**	0,00	0,00	1,11	10,71	4866	0,00	0,00	6,91	0,00
berlin52		7374	**7374**	0,00	0,00	3,17	44,34	7477	0,90	1,40	10,09	1,40
st70		675	**675**	0,00	0,00	5,08	104,44	737	7,87	9,19	11,18	9,19
pr76		57252	**57252**	0,06	0,00	6,00	33,54	58155	4,06	1,58	11,43	1,58
kroA100		20079	**20079**	0,00	0,00	9,32	158,62	20245	5,01	0,83	17,17	0,83
kroB100	0,5	20442	**20442**	0,12	0,00	20,73	157,90	21070	2,19	3,07	17,07	3,07
bier127		54325	54480	0,06	0,29	14,20	116,18	56558	3,60	4,11	22,12	3,81
pr152		65328	**65328**	4,55	0,00	44,47	285,33	65328	12,93	0,00	27,25	0,00
gr202		32813	33119	0,41	0,93	34,34	708,48	36468	1,45	11,14	41,43	10,11
tsp225		3925	3974	0,93	1,25	41,78	1366,48	6215	2,74	58,34	48,11	56,39
a280		2587	2627	0,56	1,55	66,45	2390,19	4765	10,18	84,19	67,94	81,39
lin318		40410	40687	0,55	0,69	84,71	2381,76	51909	3,47	28,46	82,75	27,58
gr431		113549	114113	0,12	0,50	161,91	3027,68	140356	3,39	23,61	134,27	23,00
burma14		2321	**2321**	0,00	0,00	1,13	8,51	2321	0,00	0,00	6,36	0,00
ulysses16		5106	**5106**	0,00	0,00	1,21	9,84	5106	0,00	0,00	6,60	0,00
ulysses22		4887	**4887**	0,00	0,00	1,72	12,16	4887	0,00	0,00	7,18	0,00
berlin52		7374	**7374**	0,00	0,00	6,44	54,77	7486	1,88	1,52	13,39	1,52
st70		675	**675**	0,00	0,00	11,28	114,48	770	3,10	14,07	16,29	14,07
pr76		73272	73809	0,62	0,73	18,13	75,81	82250	6,40	12,25	18,17	11,44
kroA100		20079	**20079**	0,00	0,00	21,70	197,03	21875	8,36	8,94	28,78	8,94
kroB100	0,8	20442	**20442**	0,09	0,00	24,80	195,04	21571	8,51	5,52	27,90	5,52
bier127		68902	69123	1,89	0,32	41,64	257,33	84208	3,51	22,21	39,27	21,82
pr152		69516	69554	0,14	0,05	51,22	444,43	76457	28,24	9,98	51,67	9,92
gr202		32867	33012	0,53	0,44	83,46	881,72	39144	2,82	19,10	77,20	18,58
tsp225		3952	3976	0,76	0,61	181,52	1487,03	6027	4,33	52,51	92,86	51,58
a280		2588	2611	0,72	0,89	170,28	2593,19	5093	4,40	96,79	137,53	95,06
lin318		40265	40686	0,43	1,05	215,42	3015,96	68988	5,78	71,33	172,79	69,56
gr431		115874	116165	0,40	0,25	439,67	5073,54	193339	2,96	66,85	299,93	66,43
média				**0,36**	**0,30**	**42,06**	**743,86**		**4,22**	**18,41**	**38,88**	**17,98**

Tabela 5.10 - PCTSP: $\gamma_i \in [1, 10000]$. Resultados Computacionais do CS-SA.

			CS-SA					SA				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		3050	**3050**	0,00	0,00	0,71	8,62	3050	0,00	0,00	5,96	0,00
ulysses16		6859	**6859**	0,00	0,00	0,70	9,41	6859	0,04	0,00	5,85	0,00
ulysses22		7013	**7013**	0,00	0,00	0,78	13,01	7013	0,57	0,00	6,04	0,00
berlin52		7542	**7542**	0,00	0,00	1,32	55,50	7847	4,82	4,04	8,68	4,04
st70		675	**675**	0,00	0,00	1,61	103,07	699	10,93	3,56	9,33	3,56
pr76		97269	**97269**	0,00	0,00	3,80	95,24	100211	5,13	3,02	9,36	3,02
kroA100		21282	**21282**	0,01	0,00	2,39	216,50	28189	3,83	32,45	9,71	32,45
kroB100	0,2	21697	**21697**	0,07	0,00	7,73	222,51	26965	10,44	24,28	11,31	24,28
bier127		108605	108617	0,20	0,01	17,70	323,98	127036	3,26	16,97	10,83	16,96
pr152		73682	**73682**	0,13	0,00	4,09	523,23	91341	22,43	23,97	14,53	23,97
gr202		38743	39252	0,13	1,31	6,48	956,72	55041	3,10	42,07	21,93	40,22
tsp225		3942	3984	0,55	1,07	8,03	1282,97	6428	4,32	63,06	24,26	61,35
a280		2592	2614	0,34	0,85	469,07	2306,55	5434	3,24	109,65	30,63	107,88
lin318		42361	42717	0,66	0,84	15,93	3190,22	100841	2,49	138,05	35,33	136,07
gr431		168632	170061	0,43	0,85	30,22	6571,68	284222	6,56	68,55	50,73	67,13
burma14		3050	**3050**	0,00	0,00	0,83	8,66	3050	0,00	0,00	5,93	0,00
ulysses16		6859	**6859**	0,00	0,00	0,88	9,73	6859	0,10	0,00	5,96	0,00
ulysses22		7013	**7013**	0,00	0,00	1,14	13,59	7013	0,72	0,00	6,32	0,00
berlin52		7542	**7542**	0,00	0,00	3,24	58,19	8009	2,45	6,19	10,24	6,19
st70		675	**675**	0,00	0,00	5,00	107,79	760	6,00	12,59	12,34	12,59
pr76		97269	**97269**	0,00	0,00	7,73	101,88	101773	4,45	4,63	11,92	4,63
kroA100		21282	**21282**	0,01	0,00	9,29	224,59	25678	15,71	20,66	15,36	20,66
kroB100	0,5	21697	**21697**	0,12	0,00	9,28	236,00	26000	13,67	19,83	17,61	19,83
bier127		108605	**108605**	0,11	0,00	14,36	352,07	129290	1,93	19,05	21,11	19,05
pr152		73682	**73682**	0,17	0,00	27,59	561,39	95688	14,33	29,87	27,42	29,87
gr202		38882	39018	0,48	0,35	423,93	1052,14	55697	2,40	43,25	41,72	42,75
tsp225		3952	4018	0,29	1,67	43,10	1412,92	6547	4,15	65,66	47,89	62,94
a280		2593	2631	0,18	1,47	66,68	2477,67	5123	8,23	97,57	67,89	94,72
lin318		42484	42682	0,37	0,47	85,15	3430,41	98834	3,86	132,64	83,70	131,56
gr431		168221	169928	0,56	1,01	161,58	7099,31	301725	1,71	79,36	138,62	77,56
burma14		3050	**3050**	0,00	0,00	1,10	8,92	3050	0,00	0,00	6,13	0,00
ulysses16		6859	**6859**	0,00	0,00	1,23	10,05	6859	0,05	0,00	6,19	0,00
ulysses22		7013	**7013**	0,00	0,00	1,71	14,37	7013	0,82	0,00	6,95	0,00
berlin52		7542	**7542**	0,00	0,00	6,68	63,00	7955	3,13	5,48	13,44	5,48
st70		675	**675**	0,00	0,00	10,82	116,13	761	7,95	12,74	17,32	12,74
pr76		97269	**97269**	0,00	0,00	17,19	116,53	106907	4,88	9,91	18,13	9,91
kroA100		21282	**21282**	0,01	0,00	22,24	249,36	27426	8,41	28,87	26,67	28,87
kroB100	0,8	21697	**21697**	0,13	0,00	24,27	258,97	28874	3,23	33,08	29,53	33,08
bier127		108605	108649	0,09	0,04	34,52	394,04	128898	3,51	18,69	39,02	18,64
pr152		73682	**73682**	0,07	0,00	48,12	620,30	110195	14,48	49,55	51,67	49,55
gr202		38855	38938	0,50	0,21	84,49	1116,35	55093	2,46	41,79	77,11	41,49
tsp225		3955	3973	1,02	0,46	105,23	1493,97	6484	3,90	63,94	92,63	63,20
a280		2593	2621	0,47	1,08	173,63	2615,69	5158	6,17	98,92	136,49	96,80
lin318		42373	42721	0,57	0,82	221,80	3709,51	96701	3,58	128,21	173,39	126,35
gr431		168765	170165	0,47	0,83	1896,75	7739,28	307464	6,33	82,18	302,28	80,69
média				**0,18**	**0,30**	**90,67**	**1145,60**		**5,11**	**36,32**	**39,23**	**35,78**

Tabela 5.11 - PCTSP: $\gamma_i \in [1, 100]$. Resultados Computacionais do CS-VNS.

			CS-VNS					VNS				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		797	**797**	0,00	0,00	0,00	1,68	797	0,00	0,00	1,27	0,00
ulysses16		1244	**1244**	0,00	0,00	0,13	1,61	1244	0,00	0,00	1,21	0,00
ulysses22		1699	**1699**	0,00	0,00	0,50	3,30	1699	0,00	0,00	2,68	0,00
berlin52		2189	**2189**	0,00	0,00	1,01	8,13	2189	0,04	0,00	4,51	0,00
st70		665	**665**	0,00	0,00	1,68	51,99	856	9,49	28,72	3,47	28,72
pr76		20390	**20390**	0,00	0,00	1,54	9,15	20390	0,00	0,00	6,02	0,00
kroA100		8002	**8002**	0,00	0,00	6,70	25,23	8002	0,00	0,00	16,23	0,00
kroB100	0,2	7300	**7300**	0,95	0,00	4,36	23,88	7300	1,80	0,00	14,59	0,00
bier127		12395	**12395**	0,00	0,00	5,33	36,06	12395	0,25	0,00	23,99	0,00
pr152		22511	**22511**	0,00	0,00	6,05	72,45	22511	0,00	0,00	47,74	0,00
gr202		14724	**14724**	0,17	0,00	19,25	197,25	14851	0,18	0,86	146,12	0,86
tsp225		3732	**3732**	0,58	0,00	13,07	888,64	4541	5,10	21,68	98,24	21,68
a280		2562	**2562**	0,38	0,00	22,05	1924,84	4568	9,96	78,30	113,42	78,30
lin318		18155	**18155**	1,14	0,00	525,82	801,40	18155	1,14	0,00	584,79	0,00
gr431		27817	**27817**	0,65	0,00	309,06	1634,63	27817	1,49	0,00	1387,69	0,00
burma14		1438	**1438**	0,00	0,00	0,82	2,07	1438	0,00	0,00	1,32	0,00
ulysses16		2360	**2360**	0,00	0,00	0,84	2,64	2360	0,00	0,00	1,36	0,00
ulysses22		2144	**2144**	0,00	0,00	1,05	4,97	2144	0,00	0,00	3,07	0,00
berlin52		3147	**3147**	0,07	0,00	2,75	19,53	3154	1,47	0,22	7,04	0,22
st70		665	**665**	0,00	0,00	4,29	67,87	907	6,78	36,39	6,46	36,39
pr76		39144	**39144**	1,07	0,00	13,47	31,86	39144	1,07	0,00	14,17	0,00
kroA100		10813	**10813**	0,00	0,00	18,87	71,62	10813	0,00	0,00	30,44	0,00
kroB100	0,5	11020	**11020**	0,93	0,00	26,11	69,38	11033	1,20	0,12	26,81	0,12
bier127		26688	**26688**	0,57	0,00	59,60	112,88	26688	0,65	0,00	47,37	0,00
pr152		38350	**38350**	0,05	0,00	85,11	200,48	38350	0,05	0,00	85,17	0,00
gr202		18386	**18386**	0,53	0,00	381,81	478,86	18462	1,72	0,41	216,68	0,41
tsp225		3719	**3719**	0,69	0,00	69,21	1070,89	5131	3,60	37,97	117,13	37,97
a280		2566	2567	0,35	0,04	236,90	2175,39	5220	9,83	103,43	158,62	103,35
lin318		23381	**23381**	1,41	0,00	138,11	1731,45	24282	2,75	3,85	827,29	3,85
gr431		44515	**44515**	1,02	0,00	1655,73	2912,61	44515	1,02	0,00	1604,02	0,00
burma14		2026	**2026**	0,00	0,00	1,17	3,75	2026	0,00	0,00	1,82	0,00
ulysses16		4079	**4079**	0,00	0,00	1,14	3,81	4079	0,00	0,00	1,54	0,00
ulysses22		3436	**3436**	0,00	0,00	1,62	5,96	3436	0,07	0,00	2,28	0,00
berlin52		5322	**5322**	0,69	0,00	11,67	39,36	5327	2,44	0,09	9,37	0,09
st70		665	**665**	0,00	0,00	9,73	88,60	859	11,79	29,17	12,01	29,17
pr76		64205	**64205**	1,39	0,00	37,58	72,56	64205	1,55	0,00	19,09	0,00
kroA100		15242	**15242**	1,09	0,00	72,36	148,53	15245	2,73	0,02	35,78	0,02
kroB100	0,8	15400	**15400**	0,61	0,00	52,78	149,41	15400	1,95	0,00	37,51	0,00
bier127		55731	55931	2,42	0,36	100,29	253,70	57331	1,74	2,87	57,09	2,50
pr152		60445	**60445**	2,38	0,00	165,72	400,51	60445	8,22	0,00	92,05	0,00
gr202		24119	**24119**	1,27	0,00	334,83	861,92	25369	1,67	5,18	206,65	5,18
tsp225		3723	**3723**	0,45	0,00	173,50	1381,40	5687	7,27	52,75	177,79	52,75
a280		2558	2562	0,32	0,16	293,81	2593,57	5872	5,41	129,55	256,80	129,20
lin318		31166	**31166**	0,85	0,00	359,54	2893,87	37524	3,29	20,40	633,51	20,40
gr431		82458	82971	2,57	0,62	1822,54	5047,55	87323	6,84	5,90	1407,26	5,25
média			**0,55**	**0,03**	**156,66**	**635,05**		**2,55**	**12,40**	**189,99**	**12,37**	

Tabela 5.12 - PCTSP: $\gamma_i \in [1, 1000]$. Resultados Computacionais do CS-VNS.

			CS-VNS					VNS				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		2313	**2313**	0,00	0,00	0,78	2,41	2313	0,00	0,00	1,21	0,00
ulysses16		4909	**4909**	0,00	0,00	0,66	2,74	4909	0,04	0,00	1,11	0,00
ulysses22		4866	**4866**	0,00	0,00	0,71	4,02	4866	0,00	0,00	1,52	0,00
berlin52		7374	**7374**	0,00	0,00	1,00	25,93	7400	1,02	0,35	4,47	0,35
st70		675	**675**	0,00	0,00	1,46	61,07	826	19,83	22,37	3,17	22,37
pr76		45853	**45853**	0,00	0,00	1,41	17,53	45853	0,00	0,00	11,40	0,00
kroA100		20079	**20079**	0,00	0,00	2,02	118,26	20473	5,34	1,96	13,68	1,96
kroB100	0,2	20442	**20442**	0,11	0,00	6,59	111,51	20927	3,30	2,37	13,37	2,37
bier127		52763	**52763**	0,06	0,00	8,53	90,04	52847	0,20	0,16	52,04	0,16
pr152		62448	62489	2,63	0,07	43,96	282,25	63796	13,32	2,16	62,64	2,09
gr202		32913	32936	0,38	0,07	13,67	627,41	34198	3,66	3,90	75,62	3,83
tsp225		3939	3945	0,96	0,15	119,89	1041,09	5689	3,48	44,43	55,33	44,21
a280		2586	2594	0,24	0,31	98,07	1902,41	5349	5,39	106,84	102,41	106,21
lin318		40543	40547	0,30	0,01	26,20	2381,74	46847	3,83	15,55	226,44	15,54
gr431		113316	113938	0,43	0,55	26,11	4249,16	122364	3,51	7,98	1442,69	7,40
burma14		2313	**2313**	0,00	0,00	0,91	2,84	2313	0,00	0,00	1,33	0,00
ulysses16		4909	**4909**	0,00	0,00	0,84	3,32	4909	0,00	0,00	1,29	0,00
ulysses22		4866	**4866**	0,00	0,00	1,11	5,53	4866	0,00	0,00	2,26	0,00
berlin52		7374	**7374**	0,00	0,00	2,67	33,73	7400	1,38	0,35	6,20	0,35
st70		675	**675**	0,00	0,00	5,67	77,74	851	14,05	26,07	6,28	26,07
pr76		57252	**57252**	0,00	0,00	7,66	41,70	57252	0,00	0,00	15,92	0,00
kroA100		20079	**20079**	0,00	0,00	8,17	149,55	20106	6,20	0,13	19,96	0,13
kroB100	0,5	20442	**20442**	0,09	0,00	9,26	141,77	21185	3,17	3,63	18,92	3,63
bier127		54325	**54325**	0,22	0,00	55,80	149,53	54325	0,45	0,00	55,47	0,00
pr152		65328	68333	1,75	4,60	22,58	354,52	70057	8,58	7,24	71,97	2,52
gr202		32813	**32813**	0,56	0,00	56,27	728,11	35253	2,38	7,44	96,28	7,44
tsp225		3925	3941	0,80	0,41	280,34	1181,78	5524	7,62	40,74	82,25	40,17
a280		2587	**2587**	0,42	0,00	226,94	2119,61	5404	5,22	108,89	142,79	108,89
lin318		40410	**40410**	0,07	0,00	152,46	2640,72	47920	3,61	18,58	275,92	18,58
gr431		113549	113940	1,49	0,34	446,77	4899,00	120631	2,11	6,24	1471,14	5,87
burma14		2321	**2321**	0,00	0,00	1,14	3,46	2321	0,00	0,00	1,55	0,00
ulysses16		5106	**5106**	0,00	0,00	1,14	4,42	5106	0,00	0,00	1,56	0,00
ulysses22		4887	**4887**	0,00	0,00	1,60	7,92	4887	0,00	0,00	3,54	0,00
berlin52		7374	**7374**	0,00	0,00	5,70	45,67	7433	0,90	0,80	9,14	0,80
st70		675	**675**	0,00	0,00	9,74	96,98	902	8,16	33,63	11,60	33,63
pr76		73272	**73272**	0,62	0,00	22,61	79,14	73272	1,55	0,00	18,66	0,00
kroA100		20079	**20079**	0,00	0,00	19,45	193,33	20714	5,25	3,16	29,78	3,16
kroB100	0,8	20442	**20442**	0,12	0,00	18,60	182,81	21165	2,49	3,54	28,99	3,54
bier127		68902	**68902**	1,38	0,00	116,04	272,73	70196	1,68	1,88	57,11	1,88
pr152		69516	69554	0,27	0,05	51,64	472,44	72832	10,14	4,77	84,27	4,71
gr202		32867	**32868**	0,36	0,00	147,92	889,22	34743	4,12	5,71	134,40	5,70
tsp225		3952	3954	0,64	0,05	169,94	1337,02	5406	9,67	36,79	132,14	36,72
a280		2588	**2588**	0,60	0,00	267,38	2362,25	5637	2,03	117,81	219,36	117,81
lin318		40265	40493	0,22	0,57	384,52	3064,12	46853	6,23	16,36	373,60	15,71
gr431		115874	121037	0,17	4,46	612,44	5932,11	124386	6,19	7,35	1245,19	2,77
média			**0,33**	**0,26**	**76,85**	**853,13**			**3,91**	**14,65**	**148,58**	**14,37**

Tabela 5.13 - PCTSP: $\gamma_i \in [1, 10000]$. Resultados Computacionais do CS-VNS.

			CS-VNS					VNS				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		3050	**3050**	0,00	0,00	0,78	2,76	3050	0,00	0,00	1,16	0,00
ulysses16		6859	**6859**	0,00	0,00	0,67	3,08	6859	0,11	0,00	1,01	0,00
ulysses22		7013	**7013**	0,00	0,00	0,71	4,87	7013	0,10	0,00	1,14	0,00
berlin52		7542	**7542**	0,00	0,00	1,03	29,64	7542	2,41	0,00	3,58	0,00
st70		675	**675**	0,00	0,00	2,38	61,89	859	7,21	27,26	4,79	27,26
pr76		97269	**97269**	0,77	0,00	1,42	63,69	97344	2,58	0,08	7,36	0,08
kroA100		21282	**21282**	0,00	0,00	2,27	151,80	21953	4,80	3,15	10,16	3,15
kroB100	0,2	21697	**21697**	0,13	0,00	15,84	140,14	22626	3,53	4,28	8,78	4,28
bier127		108605	**108605**	0,13	0,00	2,74	238,06	112433	2,42	3,52	19,40	3,52
pr152		73682	**73682**	0,17	0,00	4,09	381,79	76473	15,51	3,79	22,96	3,79
gr202		38743	38894	0,46	0,39	12,39	781,83	41583	1,59	7,33	44,68	6,91
tsp225		3942	3947	0,74	0,13	35,44	1055,78	5622	8,84	42,62	52,59	42,44
a280		2592	2597	0,28	0,19	275,97	1969,50	5119	9,20	97,49	98,34	97,11
lin318		42361	42756	0,21	0,93	26,38	2746,88	51795	1,27	22,27	147,91	21,14
gr431		168632	169435	0,56	0,48	2638,44	6256,50	189201	3,51	12,20	516,30	11,67
burma14		3050	**3050**	0,00	0,00	0,92	3,44	3050	0,00	0,00	1,35	0,00
ulysses16		6859	**6859**	0,00	0,00	0,84	4,11	6859	0,07	0,00	1,20	0,00
ulysses22		7013	**7013**	0,00	0,00	1,03	6,70	7013	0,11	0,00	1,49	0,00
berlin52		7542	**7542**	0,00	0,00	2,69	39,27	7542	3,05	0,00	5,28	0,00
st70		675	**675**	0,00	0,00	5,87	79,77	821	13,96	21,63	7,04	21,63
pr76		97269	**97269**	0,66	0,00	7,75	80,22	98614	1,43	1,38	9,13	1,38
kroA100		21282	**21282**	0,01	0,00	9,11	186,10	21521	7,56	1,12	15,76	1,12
kroB100	0,5	21697	**21697**	0,12	0,00	13,32	173,41	22850	2,87	5,31	14,03	5,31
bier127		108605	**108605**	0,17	0,00	13,65	288,60	114194	1,30	5,15	28,94	5,15
pr152		73682	73686	0,13	0,01	18,00	469,47	81648	10,24	10,81	36,88	10,81
gr202		38882	**38882**	0,47	0,00	75,96	892,77	41026	2,71	5,51	66,16	5,51
tsp225		3952	**3952**	0,62	0,00	95,14	1197,55	5674	7,26	43,57	79,48	43,57
a280		2593	**2593**	0,30	0,00	345,58	2174,22	5508	3,22	112,42	139,85	112,42
lin318		42484	42568	0,44	0,20	1322,31	3027,40	52356	3,45	23,24	201,34	22,99
gr431		168221	168689	0,53	0,28	681,79	6735,64	195740	1,48	16,36	598,13	16,04
burma14		3050	**3050**	0,00	0,00	1,14	3,92	3050	0,00	0,00	1,51	0,00
ulysses16		6859	**6859**	0,00	0,00	1,14	4,96	6859	0,02	0,00	1,54	0,00
ulysses22		7013	**7013**	0,00	0,00	1,59	8,37	7013	0,10	0,00	2,00	0,00
berlin52		7542	**7542**	0,00	0,00	5,67	49,30	7652	2,91	1,46	8,37	1,46
st70		675	**675**	0,00	0,00	9,76	98,10	823	11,07	21,93	11,63	21,93
pr76		97269	**97269**	1,09	0,00	26,84	102,92	97269	3,09	0,00	15,61	0,00
kroA100		21282	**21282**	0,00	0,00	19,54	220,47	21583	6,49	1,41	25,92	1,41
kroB100	0,8	21697	**21697**	0,13	0,00	41,86	212,20	22199	5,68	2,31	25,37	2,31
bier127		108605	**108605**	0,11	0,00	46,38	349,66	113194	2,42	4,23	46,10	4,23
pr152		73682	**73682**	0,08	0,00	61,08	566,74	81084	9,46	10,05	64,01	10,05
gr202		38855	39019	0,20	0,42	152,93	1019,53	40641	3,51	4,60	105,83	4,16
tsp225		3955	**3955**	0,65	0,00	408,18	1361,22	5778	3,85	46,09	129,28	46,09
a280		2593	**2593**	0,50	0,00	454,30	2421,08	5221	6,74	101,35	215,68	101,35
lin318		42373	42629	0,39	0,60	1027,13	3337,10	50053	6,08	18,12	300,19	17,42
gr431		168765	168874	0,37	0,06	888,48	7298,13	191909	2,67	13,71	754,18	13,64
média				**0,23**	**0,08**	**194,68**	**1028,90**		**4,13**	**15,46**	**85,63**	**15,36**

Tabela 5.14 - PCTSP: $\gamma_i \in [1, 100]$. Resultados Computacionais do CS-ILS.

			CS-ILS					ILS				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		797	**797**	0,00	0,00	0,07	2,00	797	0,00	0,00	1,59	0,00
ulysses16		1244	**1244**	0,00	0,00	0,07	1,44	1244	0,00	0,00	1,06	0,00
ulysses22		1699	**1699**	0,00	0,00	0,56	2,01	1699	0,00	0,00	1,28	0,00
berlin52		2189	**2189**	0,00	0,00	1,17	8,98	2189	0,43	0,00	4,38	0,00
st70		665	**665**	0,00	0,00	5,71	67,85	673	2,39	1,20	10,56	1,20
pr76		20390	21246	0,00	4,20	4,28	13,94	24945	11,67	22,34	9,32	17,41
kroA100		8002	**8002**	0,10	0,00	18,72	39,48	8484	9,03	6,02	26,02	6,02
kroB100	0,2	7300	**7300**	0,76	0,00	20,45	39,09	8690	6,48	19,04	23,73	19,04
bier127		12395	**12395**	0,29	0,00	26,05	58,43	18249	7,42	47,23	39,49	47,23
pr152		22511	26371	0,17	17,15	28,30	126,69	37752	4,14	67,70	88,74	43,16
gr202		14724	14735	0,36	0,07	124,44	361,65	17015	3,77	15,56	304,21	15,47
tsp225		3732	3746	0,18	0,38	565,11	1188,65	3816	1,70	2,25	485,95	1,87
a280		2562	2565	0,55	0,12	1208,00	2268,75	2769	1,52	8,08	662,97	7,95
lin318		18155	20590	2,17	13,41	550,63	1504,39	23662	5,77	30,33	1197,35	14,92
gr431		27817	28057	0,79	0,86	1414,96	2051,81	45403	3,63	63,22	1746,22	61,82
burma14		1438	**1438**	0,00	0,00	0,79	3,64	1438	0,00	0,00	2,90	0,00
ulysses16		2360	**2360**	0,00	0,00	0,86	2,69	2360	0,01	0,00	1,32	0,00
ulysses22		2144	**2144**	0,00	0,00	1,05	3,80	2144	1,17	0,00	1,61	0,00
berlin52		3147	**3147**	0,28	0,00	5,84	21,97	3756	4,61	19,35	6,59	19,35
st70		665	**665**	0,00	0,00	13,62	85,18	669	2,99	0,60	13,67	0,60
pr76		39144	39614	2,42	1,20	28,73	37,97	70022	7,65	78,88	16,74	76,76
kroA100		10813	10917	1,96	0,96	65,29	99,21	15845	11,57	46,54	40,38	45,14
kroB100	0,5	11020	11182	1,65	1,47	49,72	96,33	15170	14,19	37,66	37,88	35,66
bier127		26688	27297	1,68	2,28	105,97	137,55	62742	8,88	135,09	57,17	129,85
pr152		38350	46270	1,39	20,65	136,43	280,23	71344	20,35	86,03	123,71	54,19
gr202		18386	18441	1,21	0,30	352,43	586,90	28580	8,04	55,44	341,36	54,98
tsp225		3719	3740	0,41	0,56	1127,74	1306,44	3834	3,06	3,09	483,21	2,51
a280		2566	**2566**	0,27	0,00	1429,32	2453,38	2886	1,16	12,47	668,11	12,47
lin318		23381	23738	0,30	1,53	1174,54	1983,50	38075	17,09	62,85	1157,60	60,40
gr431		44515	47634	9,70	7,01	2895,11	3556,28	130515	9,45	193,19	2169,68	174,00
burma14		2026	**2026**	0,00	0,00	1,03	5,13	2026	0,00	0,00	2,98	0,00
ulysses16		4079	**4079**	0,00	0,00	1,17	4,03	4079	0,39	0,00	1,62	0,00
ulysses22		3436	**3436**	0,00	0,00	1,72	6,28	3447	5,00	0,32	2,14	0,32
berlin52		5322	**5322**	1,43	0,00	25,58	41,52	7020	7,80	31,91	8,00	31,91
st70		665	**665**	0,00	0,00	16,08	100,96	684	11,32	2,86	15,44	2,86
pr76		64205	66175	1,35	3,07	49,06	73,81	106685	22,40	66,16	18,83	61,22
kroA100		15242	15394	0,69	1,00	106,30	163,26	22937	16,91	50,49	33,80	49,00
kroB100	0,8	15400	15479	0,52	0,51	104,29	155,03	25110	21,46	63,05	31,98	62,22
bier127		55731	**55731**	3,48	0,00	198,39	267,51	119023	16,81	113,57	48,24	113,57
pr152		60445	61269	1,07	1,36	228,41	427,79	105074	46,37	73,83	78,97	71,50
gr202		24119	24556	0,58	1,81	425,61	802,40	48922	11,61	102,84	247,13	99,23
tsp225		3723	3738	0,35	0,40	965,44	1277,63	4616	13,26	23,99	301,49	23,49
a280		2558	**2558**	0,33	0,00	1525,07	2412,42	3562	8,61	39,25	422,25	39,25
lin318		31166	31327	0,93	0,52	1888,97	2427,28	84450	23,50	170,97	536,24	169,58
gr431		82458	**82458**	3,36	0,00	3943,50	4649,25	245739	33,30	198,02	985,79	198,02
média				**0,90**	**1,80**	**463,03**	**693,43**		**9,04**	**43,37**	**276,88**	**40,54**

Tabela 5.15 - PCTSP: $\gamma_i \in [1, 1000]$. Resultados Computacionais do CS-ILS.

			CS-ILS					ILS				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		2313	**2313**	0,00	0,00	0,65	2,43	2313	0,00	0,00	1,16	0,00
ulysses16		4909	**4909**	0,00	0,00	0,66	2,88	4909	0,00	0,00	1,19	0,00
ulysses22		4866	**4866**	0,00	0,00	0,70	3,82	4866	0,00	0,00	1,48	0,00
berlin52		7374	**7374**	0,00	0,00	2,40	30,14	7405	1,10	0,42	5,81	0,42
st70		675	**675**	0,00	0,00	10,17	64,59	703	5,86	4,15	4,10	4,15
pr76		45853	**45853**	0,00	0,00	4,41	26,04	48772	3,71	6,37	14,88	6,37
kroA100		20079	**20079**	0,00	0,00	7,06	154,23	20079	0,70	0,00	37,81	0,00
kroB100	0,2	20442	**20442**	0,01	0,00	32,73	149,78	20442	0,75	0,00	37,91	0,00
bier127		52763	**52763**	0,25	0,00	26,92	100,26	52836	0,75	0,14	48,41	0,14
pr152		62448	**62448**	0,31	0,00	211,71	337,66	62448	0,32	0,00	133,06	0,00
gr202		32913	**32913**	0,38	0,00	688,65	885,92	33793	1,40	2,67	365,64	2,67
tsp225		3939	**3939**	0,64	0,00	792,84	1108,09	4721	3,84	19,85	188,86	19,85
a280		2586	**2586**	0,53	0,00	1488,12	1922,44	3292	11,41	27,30	195,19	27,30
lin318		40543	40604	0,36	0,15	2473,69	3300,06	42363	2,19	4,49	915,48	4,33
gr431		113316	113610	0,35	0,26	5034,03	5573,75	117317	0,89	3,53	2915,53	3,26
burma14		2313	**2313**	0,00	0,00	0,80	3,02	2313	0,00	0,00	1,24	0,00
ulysses16		4909	**4909**	0,00	0,00	0,86	3,79	4909	0,00	0,00	1,43	0,00
ulysses22		4866	**4866**	0,00	0,00	1,06	5,39	4866	0,00	0,00	1,76	0,00
berlin52		7374	**7374**	0,00	0,00	4,71	38,40	7374	1,27	0,00	7,18	0,00
st70		675	**675**	0,00	0,00	14,23	87,51	731	27,18	8,30	7,23	8,30
pr76		57252	**57252**	0,00	0,00	25,86	54,59	64749	13,39	13,09	20,06	13,09
kroA100		20079	**20079**	0,00	0,00	11,82	182,19	20079	0,58	0,00	42,40	0,00
kroB100	0,5	20442	**20442**	0,02	0,00	62,85	181,62	20442	1,38	0,00	42,77	0,00
bier127		54325	54480	0,27	0,29	111,75	178,92	64706	8,72	19,11	61,82	18,77
pr152		65328	68318	1,62	4,58	71,84	484,48	69522	3,27	6,42	145,98	1,76
gr202		32813	32934	0,31	0,37	677,03	989,62	33735	2,11	2,81	371,84	2,43
tsp225		3925	**3925**	1,02	0,00	915,20	1260,53	5423	23,83	38,17	214,21	38,17
a280		2587	2592	0,38	0,19	1212,84	2143,41	4374	36,65	69,08	230,79	68,75
lin318		40410	40473	0,71	0,16	2310,20	3501,24	42799	1,70	5,91	930,84	5,75
gr431		113549	**113549**	0,69	0,00	4489,77	5959,05	124936	2,44	10,03	2539,18	10,03
burma14		2321	**2321**	0,00	0,00	1,02	3,75	2321	0,00	0,00	1,43	0,00
ulysses16		5106	**5106**	0,00	0,00	1,14	4,86	5106	0,00	0,00	1,59	0,00
ulysses22		4887	**4887**	0,00	0,00	1,60	7,40	4887	0,05	0,00	2,20	0,00
berlin52		7374	**7374**	0,00	0,00	8,40	48,67	7901	3,82	7,15	7,67	7,15
st70		675	**675**	0,00	0,00	24,80	106,62	867	13,32	28,44	12,66	28,44
pr76		73272	**73272**	0,72	0,00	61,79	86,21	101918	21,98	39,10	19,25	39,10
kroA100		20079	**20079**	0,00	0,00	23,72	209,39	23103	7,15	15,06	33,21	15,06
kroB100	0,8	20442	**20442**	0,03	0,00	109,26	202,60	21617	10,31	5,75	31,27	5,75
bier127		68902	**68902**	1,82	0,00	224,64	290,02	111161	19,57	61,33	48,98	61,33
pr152		69516	69594	0,28	0,11	222,52	517,28	80572	18,61	15,90	81,39	15,77
gr202		32867	**32867**	0,38	0,00	688,26	978,55	46309	6,95	40,90	254,20	40,90
tsp225		3952	**3952**	0,38	0,00	790,36	1410,23	5596	43,49	41,60	248,97	41,60
a280		2588	2599	0,35	0,43	1764,10	2378,96	5285	46,62	104,21	289,00	103,35
lin318		40265	**40265**	0,45	0,00	2374,94	3263,27	59671	8,11	48,20	431,23	48,20
gr431		115874	117552	2,20	1,45	3668,48	5612,67	209501	24,58	80,80	1054,22	78,22
média				**0,32**	**0,18**	**681,12**	**974,58**		**8,44**	**16,23**	**266,72**	**16,01**

Tabela 5.16 - PCTSP: $\gamma_i \in [1, 10000]$. Resultados Computacionais do CS-ILS.

			CS-ILS					ILS				
instância	σ	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		3050	**3050**	0,00	0,00	0,66	2,93	3050	0,00	0,00	1,06	0,00
ulysses16		6859	**6859**	0,00	0,00	0,68	3,53	6859	0,13	0,00	1,01	0,00
ulysses22		7013	**7013**	0,00	0,00	0,71	5,42	7013	0,00	0,00	1,19	0,00
berlin52		7542	**7542**	0,00	0,00	1,60	31,56	7783	7,56	3,20	2,11	3,20
st70		675	**675**	0,00	0,00	5,61	64,70	716	5,04	6,07	3,99	6,07
pr76		97269	**97269**	0,52	0,00	20,00	81,02	97269	2,21	0,00	16,88	0,00
kroA100		21282	**21282**	0,00	0,00	30,08	156,39	22015	9,11	3,44	7,15	3,44
kroB100	0,2	21697	**21697**	0,01	0,00	85,12	150,38	23367	3,35	7,70	7,48	7,70
bier127		108605	**108605**	0,04	0,00	107,03	285,87	118316	4,03	8,94	41,98	8,94
pr152		73682	**73682**	0,12	0,00	230,60	485,48	76827	8,73	4,27	67,18	4,27
gr202		38743	**38743**	0,54	0,00	612,27	833,51	42573	1,91	9,89	166,16	9,89
tsp225		3942	**3942**	0,93	0,00	590,84	1062,65	4710	6,77	19,48	149,18	19,48
a280		2592	**2592**	0,32	0,00	1191,01	1868,57	3094	10,89	19,37	155,14	19,37
lin318		42361	42470	0,22	0,26	1361,13	2807,99	52970	9,34	25,04	124,79	24,72
gr431		168632	**168632**	0,46	0,00	3902,55	6791,54	185846	7,72	10,21	1093,78	10,21
burma14		3050	**3050**	0,00	0,00	0,80	3,66	3050	0,00	0,00	1,20	0,00
ulysses16		6859	**6859**	0,00	0,00	0,86	4,68	6859	0,04	0,00	1,24	0,00
ulysses22		7013	**7013**	0,00	0,00	1,04	7,45	7013	0,15	0,00	1,66	0,00
berlin52		7542	**7542**	0,00	0,00	3,26	42,13	8178	11,39	8,43	3,84	8,43
st70		675	**675**	0,00	0,00	12,07	88,82	786	15,66	16,44	6,93	16,44
pr76		97269	**97269**	0,24	0,00	14,70	97,29	97344	1,27	0,08	19,44	0,08
kroA100		21282	**21282**	0,00	0,00	52,83	189,61	28666	13,82	34,70	12,94	34,70
kroB100	0,5	21697	**21697**	0,07	0,00	68,21	186,77	25935	14,95	19,53	13,46	19,53
bier127		108605	**108605**	0,08	0,00	125,92	338,22	129168	7,50	18,93	49,91	18,93
pr152		73682	**73682**	0,08	0,00	300,15	581,96	79411	13,04	7,78	82,72	7,78
gr202		38882	38923	0,13	0,11	728,44	951,25	48414	22,07	24,52	182,55	24,38
tsp225		3952	3959	0,33	0,18	821,79	1203,55	6364	16,56	61,03	169,07	60,75
a280		2593	2594	0,56	0,04	1177,86	2090,31	4101	28,93	58,16	185,71	58,10
lin318		42484	**42484**	0,18	0,00	2533,08	3123,20	65676	33,93	54,59	183,85	54,59
gr431		168221	**168221**	0,48	0,00	6942,12	7267,07	214431	16,84	27,47	1144,54	27,47
burma14		3050	**3050**	0,00	0,00	1,02	4,22	3050	0,00	0,00	1,37	0,00
ulysses16		6859	**6859**	0,00	0,00	1,17	5,44	6859	0,06	0,00	1,52	0,00
ulysses22		7013	**7013**	0,00	0,00	1,59	9,05	7013	0,38	0,00	2,06	0,00
berlin52		7542	**7542**	0,00	0,00	6,46	53,82	8396	16,30	11,32	6,78	11,32
st70		675	**675**	0,00	0,00	23,67	106,70	813	28,51	20,44	12,26	20,44
pr76		97269	**97269**	1,11	0,00	28,39	110,75	99783	10,49	2,58	18,22	2,58
kroA100		21282	**21282**	0,00	0,00	86,02	228,91	29783	22,94	39,94	23,90	39,94
kroB100	0,8	21697	**21697**	0,04	0,00	121,52	225,69	29573	25,93	36,30	24,16	36,30
bier127		108605	**108605**	0,03	0,00	165,65	386,00	131605	17,73	21,18	51,92	21,18
pr152		73682	**73682**	0,08	0,00	281,28	649,94	93276	15,49	26,59	84,19	26,59
gr202		38855	**38855**	0,21	0,00	760,18	1068,64	51856	14,00	33,46	212,97	33,46
tsp225		3955	3970	0,64	0,38	921,90	1389,18	5825	67,78	47,28	206,10	46,73
a280		2593	2608	0,48	0,58	1607,58	2391,54	4771	30,54	84,00	244,47	82,94
lin318		42373	**42373**	0,65	0,00	2746,23	3454,79	85994	86,81	102,95	270,36	102,95
gr431		168765	**168765**	0,25	0,00	5921,73	7615,20	272335	25,00	61,37	1024,64	61,37
média			**0,20**	**0,03**		**746,61**	**1077,94**		**14,11**	**20,82**	**135,18**	**20,76**

Tabela 5.17 - PCTSP: $\gamma_i \in [1, 100]$. Resultados Computacionais do CS-Aleatório.

			CS-Aleatorio					Aleatorio				
instância	**pmin**	**Melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		797	**797**	0,00	0,00	0,00	2,52	797	0,00	0,00	2,18	0,00
ulysses16		1244	**1244**	0,00	0,00	0,00	2,88	1244	0,00	0,00	2,48	0,00
ulysses22		1699	**1699**	0,00	0,00	0,60	3,23	1699	0,04	0,00	2,56	0,00
berlin52		2189	**2189**	0,00	0,00	1,04	6,56	3244	8,39	48,20	2,73	48,20
st70		665	**665**	0,00	0,00	1,41	25,25	2699	3,19	305,86	3,42	305,86
pr76		20390	21246	0,00	4,20	1,52	7,70	50342	13,59	146,90	3,25	136,95
kroA100		8002	8012	0,02	0,12	2,04	12,56	18703	10,06	133,73	2,53	133,44
kroB100	0,2	7300	**7300**	0,54	0,00	2,04	13,18	18999	6,48	160,26	2,54	160,26
bier127		12395	**12395**	1,22	0,00	3,17	21,06	53379	10,92	330,65	5,28	330,65
pr152		22511	26354	0,21	17,07	3,72	32,03	114048	9,99	406,63	5,59	332,75
gr202		14724	14759	0,18	0,24	11,79	163,83	40420	5,03	174,52	11,76	173,87
tsp225		3732	3753	0,77	0,56	19,91	375,93	13503	2,02	261,82	11,87	259,79
a280		2562	2599	0,35	1,44	42,82	574,53	15059	1,90	487,78	12,10	479,42
lin318		18155	**18155**	1,14	0,00	65,61	309,27	90685	3,23	399,50	11,28	399,50
gr431		27817	28110	0,47	1,05	128,08	417,61	295074	9,75	960,77	19,52	949,71
burma14		1438	**1438**	0,00	0,00	0,56	3,29	1438	0,32	0,00	2,40	0,00
ulysses16		2360	**2360**	0,00	0,00	0,90	3,91	2459	13,22	4,19	2,60	4,19
ulysses22		2144	**2144**	0,00	0,00	1,18	4,89	3085	5,63	43,89	2,84	43,89
berlin52		3147	**3147**	0,13	0,00	2,86	18,09	8890	5,60	182,49	4,44	182,49
st70		665	**665**	0,00	0,00	4,92	56,17	2642	4,29	297,29	6,81	297,29
pr76		39144	39973	1,16	2,12	5,67	26,69	156696	14,40	300,31	6,84	292,00
kroA100		10813	10842	2,87	0,27	8,81	53,22	56317	6,28	420,83	8,43	419,43
kroB100	0,5	11020	11189	1,14	1,53	8,89	54,32	57155	2,82	418,65	8,44	410,81
bier127		26688	27418	1,04	2,74	16,53	92,24	194309	8,14	628,08	16,54	608,69
pr152		38350	40078	15,39	4,51	19,24	141,06	349561	5,55	811,50	18,86	772,20
gr202		18386	18579	1,17	1,05	51,21	352,13	106851	0,85	481,15	23,43	475,12
tsp225		3719	3765	0,46	1,24	79,84	712,71	21322	0,54	473,33	23,97	466,32
a280		2566	2568	0,97	0,08	162,97	1237,21	20260	0,68	689,56	25,28	688,94
lin318		23381	**23381**	1,41	0,00	314,27	1267,27	232219	4,48	893,20	59,48	893,20
gr431		44515	47985	5,79	7,80	683,25	2188,31	939705	4,42	2010,99	107,08	1858,33
burma14		2026	**2026**	0,00	0,00	1,02	4,74	2384	7,79	17,67	2,58	17,67
ulysses16		4079	**4079**	0,00	0,00	1,30	5,37	5060	7,87	24,05	2,94	24,05
ulysses22		3436	**3436**	0,00	0,00	1,76	7,55	5613	9,04	63,36	3,39	63,36
berlin52		5322	**5322**	1,30	0,00	6,12	40,07	16561	5,49	211,18	7,45	211,18
st70		665	**665**	0,00	0,00	10,92	87,75	2638	7,27	296,69	11,96	296,69
pr76		64205	66260	1,01	3,20	12,47	68,62	327107	2,56	409,47	13,26	393,67
kroA100		15242	15364	0,80	0,80	21,36	131,84	96004	8,58	529,86	19,24	524,86
kroB100	0,8	15400	15465	0,40	0,42	21,73	127,42	99521	3,76	546,24	19,21	543,52
bier127		55731	57247	1,40	2,72	38,49	247,17	376052	3,76	574,76	34,00	556,89
pr152		60445	61356	1,09	1,51	80,74	350,56	627546	5,38	938,21	43,41	922,79
gr202		24119	24626	0,68	2,10	112,94	702,92	173670	1,81	620,05	64,82	605,23
tsp225		3723	3763	0,27	1,07	190,90	990,77	29683	1,38	697,29	72,10	688,81
a280		2558	2579	0,48	0,82	343,27	1966,45	25620	0,89	901,56	111,63	893,41
lin318		31166	**31166**	0,85	0,00	974,80	3170,15	405359	1,15	1200,64	148,03	1200,64
gr431		82458	86879	0,63	5,36	1491,24	5932,91	1635780	1,99	1883,77	268,84	1782,83
média			**1,01**	**1,42**	**110,09**	**489,20**		**5,12**	**453,04**	**27,54**	**441,09**	

Tabela 5.18 - PCTSP: $\gamma_i \in [1, 1000]$. Resultados Computacionais do CS-Aleatório.

			CS-Aleatorio					Aleatorio				
instância	pMin	**Melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		2313	**2313**	0,00	0,00	0,66	3,20	3546	4,09	53,31	2,27	53,31
ulysses16		4909	**4909**	0,00	0,00	0,72	3,69	6355	0,00	29,46	2,49	29,46
ulysses22		4866	**4866**	0,00	0,00	0,84	4,50	6718	2,40	38,06	2,60	38,06
berlin52		7374	**7374**	0,00	0,00	1,16	15,88	15306	10,56	107,57	3,24	107,57
st70		675	**675**	0,01	0,00	1,36	24,55	19148	3,29	2736,74	3,18	2736,74
pr76		45853	**45853**	0,00	0,00	1,49	11,84	79400	7,80	73,16	3,33	73,16
kroA100		20079	**20079**	0,00	0,00	2,06	46,93	56611	2,87	181,94	2,61	181,94
kroB100	0,2	20442	**20442**	0,10	0,00	3,98	47,20	53581	2,30	162,11	2,52	162,11
bier127		52763	**52763**	0,03	0,00	2,97	43,29	102405	4,09	94,08	5,45	94,08
pr152		62448	62455	0,04	0,01	23,97	114,99	169266	3,27	171,05	4,56	171,02
gr202		32913	33163	0,41	0,76	38,52	303,75	111349	1,20	238,31	11,77	235,76
tsp225		3939	4006	0,32	1,70	49,56	372,59	82166	0,90	1985,96	11,92	1951,07
a280		2586	2640	0,21	2,09	53,24	551,47	100754	0,38	3796,13	12,09	3716,44
lin318		40543	40547	0,30	0,01	116,78	1076,60	201607	1,27	397,27	11,56	397,22
gr431		113316	114474	0,23	1,02	328,02	1647,56	459658	2,90	305,64	19,59	301,54
burma14		2313	**2313**	0,00	0,00	0,79	3,86	2980	7,27	28,84	2,45	28,84
ulysses16		4909	**4909**	0,00	0,00	1,00	4,70	5852	2,34	19,21	2,61	19,21
ulysses22		4866	**4866**	0,04	0,00	1,19	6,30	6455	4,40	32,66	2,86	32,66
berlin52		7374	**7374**	0,00	0,00	2,92	30,53	16420	10,72	122,67	4,85	122,67
st70		675	**675**	0,00	0,00	4,94	57,48	10838	4,44	1505,63	6,68	1505,63
pr76		57252	**57252**	0,13	0,00	5,34	33,62	193264	4,08	237,57	6,78	237,57
kroA100		20079	**20079**	0,00	0,00	8,95	108,45	81421	3,39	305,50	8,50	305,50
kroB100	0,5	20442	**20442**	0,03	0,00	8,85	107,64	80331	1,94	292,97	8,53	292,97
bier127		54325	54487	0,06	0,30	14,03	114,27	232134	3,45	327,31	15,00	326,04
pr152		65328	68305	0,91	4,56	40,30	272,44	391807	3,61	499,75	18,68	473,61
gr202		32813	33113	0,46	0,91	50,68	547,22	147705	3,57	350,14	23,40	346,06
tsp225		3925	3993	0,44	1,73	70,44	764,08	61604	0,92	1469,53	39,00	1442,80
a280		2587	2605	1,07	0,70	181,38	1244,01	69622	0,93	2591,23	52,09	2572,63
lin318		40410	**40410**	0,07	0,00	299,69	2436,88	311014	0,86	669,65	59,48	669,65
gr431		113549	113942	0,41	0,35	583,13	3920,51	1038828	1,83	814,87	106,94	811,72
burma14		2321	**2321**	0,00	0,00	1,03	4,78	2997	10,13	29,13	2,68	29,13
ulysses16		5106	**5106**	0,00	0,00	1,31	5,89	5954	10,37	16,61	2,92	16,61
ulysses22		4887	**4887**	0,00	0,00	1,78	8,30	6753	15,74	38,18	3,35	38,18
berlin52		7374	**7374**	0,00	0,00	6,16	46,18	18873	11,35	155,94	7,47	155,94
st70		675	**675**	0,00	0,00	11,04	89,65	4656	5,34	589,78	12,28	589,78
pr76		73272	**73272**	0,86	0,00	12,61	74,73	329415	4,93	349,58	13,17	349,58
kroA100		20079	**20079**	0,00	0,00	21,29	175,41	111262	4,00	454,12	19,34	454,12
kroB100	0,8	20442	**20442**	0,03	0,00	35,68	174,11	107507	4,45	425,91	19,26	425,91
bier127		68902	69256	1,09	0,51	33,93	254,93	390316	2,85	466,48	32,43	463,58
pr152		69516	**69516**	0,10	0,00	49,71	449,01	666308	1,87	858,50	43,89	858,50
gr202		32867	33075	0,36	0,63	75,45	827,64	192900	2,02	486,91	64,30	483,22
tsp225		3952	3994	0,39	1,06	93,82	1191,13	44328	1,64	1021,66	77,69	1009,86
a280		2588	2616	0,37	1,08	347,03	1982,12	41416	5,62	1500,31	111,21	1483,18
lin318		40265	40493	0,22	0,57	747,40	3933,26	438669	0,86	989,45	147,61	983,32
gr431		115874	116175	0,45	0,26	1775,24	6774,96	1683522	2,22	1352,89	268,93	1349,13
média			**0,20**	**0,41**	**113,61**	**664,71**		**4,10**	**630,53**	**28,52**	**625,05**	

Tabela 5.19 - PCTSP: $\gamma_i \in [1, 10000]$. Resultados Computacionais do CS-Aleatório.

			CS-Aleatorio					Aleatorio				
instância	pMin	**Melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
burma14		3050	**3050**	0,00	0,00	0,65	3,38	18454	3,08	505,05	2,31	505,05
ulysses16		6859	**6859**	0,00	0,00	0,80	3,79	44640	1,26	550,82	2,46	550,82
ulysses22		7013	**7013**	0,00	0,00	0,85	4,75	47390	3,82	575,75	2,47	575,75
berlin52		7542	**7542**	0,00	0,00	1,07	14,76	111143	5,75	1373,65	3,08	1373,65
st70		675	**675**	0,00	0,00	1,34	24,28	185227	3,87	27341,04	3,15	27341,04
pr76		97269	**97269**	0,01	0,00	1,50	29,24	311556	4,84	220,30	3,41	220,30
kroA100		21282	**21282**	0,00	0,00	2,07	47,43	367962	2,07	1628,98	2,53	1628,98
kroB100	0,2	21697	**21697**	0,16	0,00	2,09	50,05	343294	1,35	1482,22	3,86	1482,22
bier127		108605	108723	0,21	0,11	2,87	78,63	515407	2,11	374,57	4,68	374,06
pr152		73682	73686	0,19	0,01	13,97	120,04	644910	3,85	775,26	4,63	775,21
gr202		38743	39107	0,45	0,94	25,13	201,04	728224	0,94	1779,63	4,34	1762,13
tsp225		3942	3996	0,69	1,37	49,63	369,62	734429	0,43	18530,87	11,86	18279,10
a280		2592	2627	0,43	1,35	114,31	818,60	923918	0,90	35544,98	9,26	35070,08
lin318		42361	42756	0,21	0,93	217,46	1047,37	1183811	1,76	2694,58	11,41	2668,76
gr431		168632	170894	0,31	1,34	334,82	2160,57	1916311	0,82	1036,39	19,68	1021,34
burma14		3050	**3050**	0,00	0,00	0,81	4,18	7382	6,74	142,03	2,39	142,03
ulysses16		6859	**6859**	0,00	0,00	1,01	4,95	25905	5,15	277,68	2,68	277,68
ulysses22		7013	**7013**	0,00	0,00	1,20	7,12	25091	6,79	257,78	2,79	257,78
berlin52		7542	**7542**	0,00	0,00	2,97	30,50	54063	14,01	616,83	4,94	616,83
st70		675	**675**	0,00	0,00	4,59	52,42	88457	6,93	13004,74	5,82	13004,74
pr76		97269	**97269**	0,01	0,00	5,36	60,47	345098	3,36	254,79	6,67	254,79
kroA100		21282	**21282**	0,00	0,00	8,92	116,07	257583	2,80	1110,33	8,48	1110,33
kroB100	0,5	21697	**21697**	0,08	0,00	8,93	118,40	236998	5,88	992,31	9,73	992,31
bier127		108605	**108605**	0,13	0,00	33,81	193,48	492125	2,55	353,13	14,24	353,13
pr152		73682	**73682**	0,08	0,00	40,27	313,11	706522	2,50	858,88	18,52	858,88
gr202		38882	38974	0,57	0,24	68,32	484,60	493000	3,71	1167,94	21,27	1164,95
tsp225		3952	3990	0,33	0,96	110,68	753,29	427112	1,61	10707,49	39,09	10604,56
a280		2593	2619	0,49	1,00	281,60	2073,64	517179	3,37	19845,20	47,07	19647,19
lin318		42484	**42568**	0,44	0,20	504,94	2530,88	908106	1,41	2037,52	59,31	2033,31
gr431		168221	170060	0,58	1,09	1761,05	5138,12	1943330	1,49	1055,22	107,82	1042,73
burma14		3050	**3050**	0,00	0,00	1,03	5,10	3816	10,19	25,11	2,67	25,11
ulysses16		6859	**6859**	0,00	0,00	1,31	6,22	11290	5,10	64,60	2,96	64,60
ulysses22		7013	**7013**	0,00	0,00	1,77	9,69	12457	5,62	77,63	3,55	77,63
berlin52		7542	**7542**	0,00	0,00	6,51	49,11	27698	8,56	267,25	8,12	267,25
st70		675	**675**	0,00	0,00	10,54	87,11	18288	13,78	2609,33	11,39	2609,33
pr76		97269	**97269**	0,01	0,00	12,27	101,00	388158	4,47	299,06	13,22	299,06
kroA100		21282	**21282**	0,00	0,00	31,34	189,29	174371	3,05	719,34	19,25	719,34
kroB100	0,8	21697	**21697**	0,07	0,00	41,24	191,73	165303	4,69	661,87	20,25	661,87
bier127		108605	**108605**	0,23	0,00	53,67	322,78	494899	2,32	355,69	31,81	355,69
pr152		73682	**73682**	0,06	0,00	70,84	517,65	781100	3,66	960,10	44,78	960,10
gr202		38855	39138	0,23	0,73	94,46	841,62	318165	1,48	718,85	91,41	712,93
tsp225		3955	3997	0,27	1,06	195,96	1160,96	146244	6,02	3597,70	176,75	3558,84
a280		2593	2602	0,95	0,35	251,64	1974,77	193086	2,69	7346,43	209,86	7320,68
lin318		42373	42629	0,39	0,60	453,27	4108,50	654091	0,63	1443,65	247,90	1434,38
gr431		168765	170541	0,28	1,05	1467,67	8421,81	2027998	2,51	1101,67	269,25	1089,16
média				**0,17**	**0,30**	**139,92**	**774,27**		**4,00**	**3718,76**	**35,40**	**3692,13**

5.10 Considerações Finais

Este Capítulo apresentou uma aplicação do método híbrido Busca por Agrupamento (CS) para resolver de forma aproximada o Problema do Caixeiro Viajante com Coleta de Prêmios (PCTSP), utilizando diferentes meta-heurísticas para gerar soluções para o processo de agrupamento, e também um método gerador de soluções aleatórias.

Além da abordagem heurística, foi proposto também a resolução da formulação matemática proposta por Balas (1989) utilizando as restrições propostas por Torres e Brito (2003) para evitar que existam sub-rotas na solução. Essa formulação foi resolvida pelo *software* CPLEX versão 11.1.1, sendo este capaz de resolver de forma ótima somente as instâncias menores. Porém estes resultados foram úteis para validar os resultados obtidos pelo CS.

As cinco versões implementadas do CS com diferentes métodos geradores de soluções para o processo de agrupamento encontraram boas soluções em um tempo computacional competitivo. O CS também se mostrou robusto obtendo sempre soluções próximas às melhores soluções conhecidas.

Apesar de todas as versões terem encontrado bons resultados, a versão do CS com a meta-heurística VNS parece ser a melhor estratégia para o PCTSP, pois apresentou os melhores valores médios de *gap* e desvio entre todos os relatados.

Uma questão importante foi a implementação de três heurísticas de busca local que não fazem chamadas a função objetivo, calculando o novo valor da solução por meio das alterações causadas na rota. Desta forma, foi possível resolver instâncias relativamente grandes em tempos computacionais razoáveis.

Enfim, os resultados obtidos mostram o potencial da abordagem proposta, na qual soluções de boa qualidade são obtidas para o PCTSP. Apesar de não haver comparação com outros métodos da literatura, as informações contidas neste Capítulo validam a utilização do CS para resolver problemas da classe TSPP.

6 PROBLEMA DE BALANCEAMENTO E DESIGNAÇÃO DE TRABALHADORES EM LINHA DE PRODUÇÃO

6.1 Introdução

6.1.1 Trabalhadores Portadores de Deficiências

A Organização Mundial de Saúde (OMS) estima que 10% da população mundial, cerca de 610 milhões de pessoas, são portadores de algum tipo de deficiência. Destas, mais da metade estão em idade ativa de trabalho e sofrem com altas taxas de desemprego, variando de 13% no Reino Unido a 80% em países em desenvolvimento.

No Brasil, o Censo 2000 divulgou que 14,5% da população são portadores de algum tipo de deficiência, o que corresponde a 24,5 milhões de pessoas. Das quais 15,14 milhões têm idade e condições de integrarem o mercado formal de trabalho. Contudo, apesar de não haver estudos sobre a proporção de pessoas com deficiências que exercem alguma atividade produtiva, Organizações Não-Governamentais (ONG's), que trabalham com a inclusão social desses cidadãos, estimam que a taxa de desemprego entre os deficientes é muito alta no Brasil.

Visando uma maior inclusão dos deficientes ao mercado de trabalho, desde 1991 existe no Brasil uma lei que obriga as empresas com mais de 100 funcionários a contratarem pessoas portadoras de deficiências (art. 93 da Lei n. 8.213/91). A lei prevê que uma determinada quantidade de vagas, que varia de 2% a 5% do número total de funcionários, deve ser reservada para pessoas deficientes. Apesar disso, a integração de pessoas com deficiências à sociedade e, em particular, ao mercado de trabalho, encontra ainda várias dificuldades e diversas vezes esbarra em discriminações do passado (JAIME; CARMO, 2005). Pessoas com diferentes tipos de deficiências podem exercer praticamente qualquer atividade profissional. Entretanto, encontrar mão-de-obra qualificada também tem sido um desafio.

Como forma de facilitar a inclusão desses trabalhadores no mercado de trabalho, alguns países (como por exemplo: Espanha, Japão, Inglaterra, entre outros) adotaram a estratégia de criar Centros de Trabalho para Deficientes (CTD's). Esses centros funcionam como uma primeira etapa na integração dessas pessoas que, eventualmente, serão absorvidas pelo mercado formal de trabalho.

Esse modelo de integração sócio-trabalho procura afastar-se do estereótipo

tradicional que considera as pessoas deficientes como incapazes de desenvolver um trabalho profissional contínuo. Da mesma maneira que em qualquer outra empresa, um CTD compete em mercados reais e precisa ser suficientemente flexível e eficiente para adaptar-se às variações do mercado, sendo que a única diferença é que um CTD é uma organização sem fins lucrativos.

Sendo assim, o benefício que geralmente pode ser obtido com a melhora de eficiência é o crescimento do CTD, ou seja, mais trabalhos para as pessoas deficientes, gradualmente integrando pessoas com níveis mais elevados de inaptidão, o que é de fato o objetivo principal de todo CTD.

6.1.2 Linhas de Produção no CTD

Em Miralles et al. (2007) é mostrado como a utilização de linhas de produção nesses centros provê muitas vantagens, sendo que, a divisão tradicional do trabalho em tarefas únicas pode se tornar uma ferramenta perfeita para fazer invisíveis determinadas deficiências do trabalhador. Além disso, uma atribuição apropriada da tarefa também pode transformar-se em um bom método terapêutico para reabilitação das deficiências. Mas algumas restrições específicas relacionadas à variabilidade dos tempos surgem neste ambiente e, então, o procedimento de balanceamento aplicado deveria conciliar os seguintes objetivos (que devem ser vistos como complementares):

(1) maximizar a eficiência da linha de produção balanceando a carga de trabalho atribuída a cada trabalhador disponível em cada estação;

(2) satisfazer e respeitar as restrições existentes nesse ambiente devido aos fatores humanos ao atribuir tarefas aos trabalhadores.

O balanceamento da linha de produção é um campo de estudo consolidado que surgiu após a construção da primeira linha de produção movida por meios mecânicos (linha do Ford T criada em 1913 por Henry Ford). Um dos tipos de linhas de produção mais comum são linhas de produção com trabalho individual. Um trabalhador para cada posto de trabalho. Um posto de trabalho em cada estação de trabalho, ou seja, neste caso posto de trabalho e estação de trabalho podem ser considerados sinônimos. O balanceamento é feito considerando-se toda a linha.

Após analisar alguns CTD's, Miralles et al. (2007) estabeleceram algumas características específicas que podem ser encontradas nesses ambientes, as quais serviram de motivação para definir um novo problema de linha de produção, chamado Problema de Balanceamento e Designação de Trabalhadores em Linhas de Produção (ALWABP, do inglês *Assembly Line Worker Assignment and Balancing Problem*), o qual foi apresentado e modelado matematicamente em Miralles et al. (2008).

O ALWABP pode ser classificado como um problema NP-*hard*, pois este é uma generalização do Problema Simples de Balanceamento da Linha de Produção (SALB, do inglês *Simple Assembly Line Balancing Problem*) (GUTJAHR; NEMHAUSER, 1964), no qual cada tarefa possui um tempo de execução fixo independente do trabalhador que a executa. E, o SALBP é classificado como NP-*hard* (SCHOLL; BECKER, 2006). Portanto, a aplicação de métodos heurísticos é uma boa alternativa para obter bons resultados em pouco tempo computacional.

Isto é ainda mais importante se considerar que, em razão do absentismo elevado e dos exames físicos e psicológicos periódicos dos trabalhadores, o responsável pela linha de produção de uma CTD conhece apenas no inicio de cada dia de trabalho quais trabalhadores estão disponíveis. Consequentemente, algoritmos eficientes que forneçam boas soluções são desejáveis para gerar a linha de produção diariamente.

Este Capítulo apresenta uma aplicação do CS para resolver o ALWABP de forma aproximada. Para validar os resultados obtidos pelo CS utiliza-se o *software* CPLEX para obter limitantes para esse problema. Cinco versões do CS foram implementadas utilizando diferentes meta-heurísticas e um gerador de soluções aleatórias.

6.2 Definição e Formulação Matemática

De uma forma geral, uma linha de produção consiste de um conjunto finito de tarefas, cada uma tendo um tempo de execução, e um conjunto de relações de precedência, as quais especificam a ordem de execução das tarefas. O problema do ALWABP é atribuir as tarefas para uma sequência de estações, tal que, as relações de precedências sejam satisfeitas e alguma medida de eficiência seja otimizada. Entretanto, como nos CTD's alguns trabalhadores podem ser muito lentos para executar certas tarefas ou até incapazes, e muito rápidos na execução de outras, o problema não consiste apenas em atribuir tarefas para estações, mas também trabalhadores disponíveis para estações, sempre respeitando as incompatibilidades

quando atribuir tarefas para trabalhadores. A Figura 6.1 ilustra um exemplo de um conjunto de relações de precedências entre 11 tarefas.

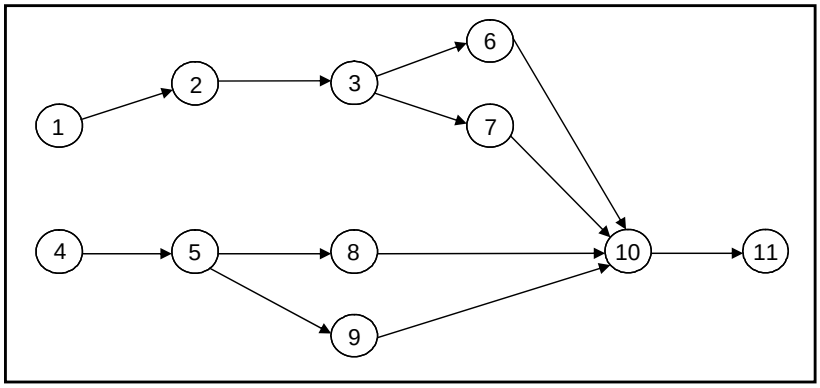

Figura 6.1 - Exemplo de um conjunto de relações de precedências entre tarefas.

As principais características do ALWABP são:

- os tempos de processamento das tarefas e as relações de precedências são determinísticos;
- existe um dado número de trabalhadores disponíveis, e o tempo de processamento das tarefas pode ser diferente dependendo de qual trabalhador executa a tarefa (uma vez que os trabalhadores têm diferentes habilidades e capacidades);
- não existem trabalhadores lentos ou rápidos. Ao invés disso, trabalhadores podem ser muito lentos, ou até incapazes, de executar algumas tarefas, mas muito eficientes quando executam outras tarefas;
- todo trabalhador é atribuído para somente uma estação de trabalho;
- toda tarefa é atribuída para somente uma estação de trabalho, contanto que o trabalhador selecionado para aquela estação seja capaz de realizar a tarefa, e que as relações de precedência sejam satisfeitas.

Analogamente ao SALBP, quando se deseja minimizar o número de estações, o problema é chamado ALWABP-1, e quando o objetivo é minimizar o tempo de ciclo (ou maximizar a taxa de produção), o problema é chamado ALWABP-2. Esta última é a situação mais comum nos CTD's, uma vez que, visa o aumento da eficiência da produção sem retirar nenhum posto de trabalho existente.

A situação mais típica em um CTD é ter um dado número de trabalhadores disponíveis (cada um deles com tempos de operação definidos para cada tarefa) e onde a eficiência da linha de produção deve ser maximizada. O tempo de ciclo (C_{tempo}) é a variável de decisão que representa a quantidade de tempo que um produto pode ser processado por uma estação da linha de produção, tendo relação direta com a taxa de produção da linha. Portanto, maximizar a eficiência significa, neste ambiente, minimizar o tempo de ciclo.

O modelo matemático do ALWABP-2, proposto por Miralles et al. (2008), é apresentado a seguir.

$$\min z = C_{tempo} \tag{6.1}$$

sujeito a:

$$\sum_{h \in H} \sum_{s \in S} x_{shi} = 1 \qquad \forall i \in N \tag{6.2}$$

$$\sum_{\mathrm{s} \in \mathrm{S}} y_{sh} \leq 1 \qquad \forall \mathrm{h} \in \mathrm{H} \tag{6.3}$$

$$\sum_{\mathrm{h} \in \mathrm{H}} y_{sh} \leq 1 \qquad \forall \mathrm{s} \in \mathrm{S} \tag{6.4}$$

$$\sum_{h \in H} \sum_{s \in S} s \cdot x_{shi} \leq \sum_{h \in H} \sum_{s \in S} s \cdot x_{shj} \qquad \forall i, j / i \in P_j \tag{6.5}$$

$$\sum_{\mathrm{i} \in \mathrm{N}} t_{hi} \cdot x_{shi} \leq C_{tempo} \qquad \forall \mathrm{h} \in \mathrm{H}, \forall s \in S \tag{6.6}$$

$$\sum_{\mathrm{i} \in \mathrm{N}} x_{shi} \leq B \cdot y_{sh} \qquad \forall \mathrm{h} \in \mathrm{H}, \forall s \in S \tag{6.7}$$

$$y_{sh} \in [0, 1] \qquad \forall \mathrm{s} \in \mathrm{S}, \forall h \in H \tag{6.8}$$

$$x_{shi} \in [0, 1] \qquad \forall \mathrm{s} \in \mathrm{S}, \forall h \in H, \forall i \in N \tag{6.9}$$

no qual,

- i e j são as tarefas;
- h são os trabalhadores;
- s são as estações de trabalho;
- N é o conjunto de tarefas;
- H é o conjunto de trabalhadores disponíveis;
- S é o conjunto de estações de trabalho;
- C_{tempo} é o tempo de ciclo;
- t_{hi} é o tempo de operação para o trabalhador h executar a tarefa i;
- P_j é o conjunto de tarefas que precedem imediatamente a tarefa j na rede de precedência;
- x_{shi} é uma variável binária igual a 1 somente se a tarefa i é atribuída para o trabalhador h na estação s;
- y_{sh} é uma variável binária igual a 1 somente se o trabalhador h é atribuído para a estação s.

A função objetivo 6.1 do ALWABP-2 minimiza o tempo de ciclo C_{tempo}. As restrições 6.2 garantem que toda tarefa i é atribuída para apenas uma estação s e um trabalhador h. As restrições 6.3 e 6.4 asseguram que todo trabalhador pode ser atribuído para somente uma única estação, e que em toda estação existe somente um trabalhador. O conjunto de restrições 6.5 reflete as relações de precedência entre as tarefas i e j, onde i é predecessor de j. As restrições 6.6 e 6.7 implicam que todo trabalhador h atribuído para estação s pode ter mais que uma tarefa, desde que o tempo de ciclo C_{tempo} não seja extrapolado. Como o tempo de ciclo C_{tempo} e y_{sh} são ambos variáveis, as restrições 6.6 e 6.7 são definidas separadamente para que o modelo seja mantido linear. As restrições 6.8 e 6.9 definem as variáveis y_{sh} e x_{shi} como binárias. A constante B precisa ter um valor maior que o somatório de todos os tempos de processamento.

6.3 Revisão Bibliográfica

O ALWABP é um problema que foi recentemente proposto, sendo assim, ainda não há na literatura muitos trabalhos sobre este. A maioria dos problemas sobre linha de produção encontrados na literatura trabalha com tempos de operação fixos, independentemente de qual trabalhador executa a tarefa. Essa simplificação é justificada por causa do fato de que, na maioria dos casos de linhas de produção reais, a variação de tempo entre os trabalhadores é muito pequena.

São poucos os trabalhos encontrados na literatura que consideram tempos diferentes de operação das tarefas dependendo de qual trabalhador a executa. Mansoor (1968) propôs uma heurística para resolução de uma linha de produção considerando níveis variáveis de performance dos trabalhadores. Bartholdi e Eisensteein (1996) consideram o caso de trabalhadores com velocidades diferentes, mas em um tipo particular de linha, a *Toyota Swen System*. Doerr et al. (2000) estudaram o desenvolvimento de uma linha de produção considerando trabalhadores com diferentes habilidades e onde a hora-extra poderia ser usada quando a quota de produção diária não fosse alcançada. Hopp et al. (2001) e Gel et al. (2002) apresentaram casos nos quais existem trabalhadores lentos e rápidos, independentemente de qual tarefas eles executam, porém as soluções partem de uma linha de produção balanceada e o objetivo é apenas ter uma rotação eficiente dos trabalhadores e não balancear a linha. Corominas et al. (2003) apresentaram um modelo para um problema de balanceamento de linha de produção que considera tempos variáveis de duração das tarefas dependendo de qual estação ela está alocada, sendo aplicado a várias situações industriais nas quais é necessário diferenciar dois tipos de trabalhadores: experientes e inexperientes.

Outra situação que leva em consideração velocidades variáveis é o Problema de Projeto da Linha de Produção (ALDP, do inglês: *Assembly Line Design Problem*) (REKIEK et al., 2002) que trabalha com a instalação de máquinas com velocidades diferentes, sendo necessário selecionar as máquinas e balancear a rede de precedências. Este problema se distingue do encontrado nos CTD's em razão do objetivo ser reduzir os custos de instalação das máquinas e também pelo fato de haver a possibilidade de aquisição de vários equipamentos semelhantes.

Os primeiros estudos sobre as vantagens de utilizar linhas de produção nos CTD's foram realizados por Miralles et al. (2007). Posteriormente, Miralles et al. (2008)

introduziram o ALWABP, definindo uma formulação matemática para o problema e implementando um algoritmo *Branch* e *Bound* com diferentes estratégias de busca. Os autores aplicaram esse método em um ambiente real de um CTD situado na cidade de Valência (Espanha), mostrando que a Pesquisa Operacional pode abranger, além de objetivos econômicos e produtivos, objetivos sociais.

Costa e Miralles (2008) apresentam uma variante para o ALWABP, no qual é estudado como programar a rotação de tarefas neste problema. A rotação pode trazer diversos benefícios, tais como, aumentar a motivação dos trabalhadores, combater certas doenças do trabalho e, principalmente, auxiliar o tratamento terapêutico dos trabalhadores. Os autores propuseram uma formulação inteira mista e um método de decomposição heurístico para resolução desse problema.

6.4 Representação do Problema

Uma solução do ALWABP-2 é composta por dois vetores. O primeiro vetor representa a alocação tarefa/estação. O segundo vetor representa a alocação trabalhador/estação. A Figura 6.2 ilustra a representação de uma solução com 11 tarefas $(n_1, n_2, ..., n_{11})$, 5 trabalhadores $(h_1, h_2, ..., h_5)$ e 5 estações $(s_1, s_2, ..., s_5)$.

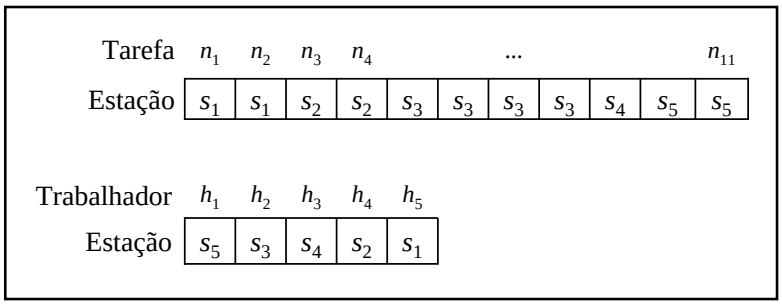

Figura 6.2 - Exemplo da representação de uma solução do ALWABP.

6.5 Estruturas de Vizinhança

Três movimentos diferentes podem ser definidos para compor tipos distintos de vizinhanças, permitindo que o espaço de busca possa ser pesquisado de forma adequada. Os movimentos para o ALWABP são:

- m_1: trocar a alocação de dois trabalhadores de estações;

- m_2: trocar a alocação de duas tarefas de estações diferentes;

- m_3: transferir uma tarefa de uma estação para outra estação.

Todas os movimentos podem produzir inviabilidades. O movimento m_1 pode fazer com que uma tarefa incompatível seja alocada a um trabalhador, e os movimentos m_2 e m_3, além de poderem causar essa inviabilidade trabalhador/tarefa, também podem violar a rede de precedências das tarefas. Neste trabalho optou-se por permitir qualquer movimento, apesar da enorme quantidade de movimentos inviáveis, sendo as soluções que forem inviáveis penalizadas na função objetivo.

6.6 Função Objetivo

O cálculo da função objetivo do ALWABP-2 computa os tempos de operação de cada estação, considerando as tarefas e o trabalhador alocados na estação. A função objetivo corresponde ao tempo de ciclo da linha de produção (C_{tempo}), que é o maior tempo de operação das estações. Soluções que não satisfaçam as restrições da rede de precedências ou aloquem alguma tarefa incompatível com um trabalhador precisam ser penalizadas. Desta forma, uma solução é avaliada pela seguinte função objetivo:

$$f(s) = C_{tempo} + \omega * f_r(s) + \delta * f_t(s) \tag{6.10}$$

no qual, os componentes f_r e f_t mensuram, respectivamente, a inviabilidade da rede de precedências e da alocação tarefa/trabalhador incompatíveis. ω e δ são os pesos que refletem a importância relativa de cada um dos componentes de f. Neste trabalho utilizou-se $\omega = 1000$ e $\delta = 500$, privilegiando as soluções que satisfaçam as restrições da rede de precedências, pois essas inviabilidades são mais complicadas de serem eliminadas.

6.7 Medida de Distância entre duas Soluções

Para calcular a distância entre duas soluções do ALWABP-2 é utilizada uma medida de distância que calcula o número de tarefas atribuídas em estações diferentes entre as soluções. Portanto, quanto maior o número de tarefas alocadas em estações diferentes entre duas soluções, maior será a distância entre elas e menor será a similaridade.

Essa medida de distância foi escolhida pois leva em consideração as alocações das tarefas nas estações de trabalho, sendo esta a maior dificuldade desse problema.

6.8 CS Aplicado ao ALWABP

Cinco versões do CS aplicado ao ALWABP-2 foram implementadas utilizando diferentes métodos para gerar soluções para o processo de agrupamento, sendo quatro meta-heurísticas clássicas e um gerador de soluções aleatórias.

Construir uma solução viável para o ALWABP-2 é uma tarefa muito árdua e que pode conduzir a tempos computacionais muito altos. Assim, optou-se por gerar soluções iniciais aleatórias, as quais são obtidas alocando aleatoriamente as tarefas e os trabalhadores para as estações. Com objetivo de diminuir as inviabilidades, foram definidos intervalos nos quais as tarefas são alocadas em estações de acordo com sua posição na rede de precedências. Portanto, as estações são dividas em quatro grupos, e cada tarefa é alocada aleatoriamente a uma estação do grupo a que ela pertencer de acordo com sua ordem de execução.

6.8.1 Meta-heurísticas

No CS foram implementadas as meta-heurísticas Algoritmo Genético (GA), Recozimento Simulado (SA), Pesquisa em Vizinhança Variável (VNS) e Busca Local Iterativa (ILS) como geradores de soluções para o processo de agrupamento, além de um método para gerar soluções aleatoriamente. Este método se baseia na geração de solução apresentada anteriormente, sendo que, a cada iteração uma solução é gerada aleatoriamente e enviada para o processo de agrupamento. Neste método foram geradas 10000 soluções aleatórias. A seguir são descritos as particularidades de cada meta-heurística.

6.8.1.1 Algoritmo Genético

No Algoritmo Genético (GA) (ver Subseção 2.2.1) a população inicial possui μ indivíduos gerados aleatoriamente. A cada geração realiza-se a seleção dos pais por meio do método de seleção por torneio, no qual três indivíduos são escolhidos aleatoriamente na população corrente e o melhor indivíduo é selecionado para fazer parte da população de pais. Esse processo é repetido até μ pais serem selecionados.

O operador de cruzamento *Block Order Crossover* (BOX) (SYSWERDA, 1989) foi

implementado no GA, combinando dois pais em um único filho por meio da cópia de blocos aleatórios de ambos os pais. Blocos copiados de um pai não são copiados do outro. Esse cruzamento é aplicado sobre o vetor tarefa/estação. Para realocar os trabalhadores nas estações de trabalho foi implementado o cruzamento cíclico (OLIVER et al., 1987), evitando que o filho tenha trabalhadores repetidos. A Figura 6.3 apresenta um exemplo do cruzamento BOX aplicado sobre as tarefas e a Figura 6.4 exemplifica o cruzamento cíclico aplicado sobre os trabalhadores. O segundo filho é criado repetindo esses operadores para o mesmo par de pais.

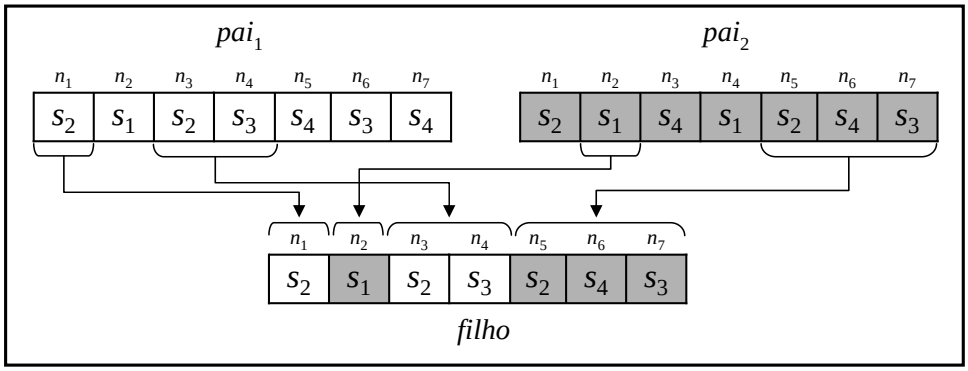

Figura 6.3 - Exemplo do cruzamento BOX para o ALWABP.

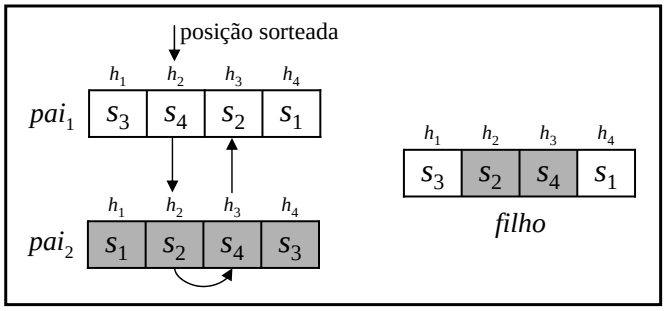

Figura 6.4 - Exemplo do cruzamento cíclico para o ALWABP.

O operador de mutação consiste em transferir uma tarefa para uma estação vizinha aleatoriamente. A probabilidade de mutação é verificada para cada tarefa. Após aplicar o operador de mutação na população de filhos, esta se torna a nova população corrente.

A probabilidade de cruzamento (p_c) é 0,95 e a probabilidade de mutação (p_m) é 0,05. O tamanho da população (μ) foi definido como 70 indivíduos e o GA é executado por 1000 gerações. Então, a cada geração uma amostragem aleatória de 15% dos filhos é enviada para o processo de agrupamento.

6.8.1.2 Recozimento Simulado

O Recozimento Simulado (SA) parte de uma solução inicial gerada aleatoriamente, e a cada iteração gera-se um vizinho qualquer em uma das estruturas de vizinhança escolhida aleatoriamente. Os demais passos do SA são idênticos ao SA clássico apresentado na Subseção 2.2.2.

Os parâmetros de controle do SA utilizados neste trabalho foram: taxa de resfriamento $\alpha = 0,96$, número de iterações para cada temperatura $SA_{max} = 2000$ e temperatura inicial $T_0 = 10^6$. O intervalo entre duas soluções enviadas para o processo de agrupamento foi de 300 vizinhos.

6.8.1.3 Pesquisa em Vizinhança Variável

A Pesquisa em Vizinhança Variável (VNS) também parte de uma solução inicial aleatória, e explora vizinhanças cada vez mais distantes durante o processo de busca, retornando para a primeira estrutura de vizinhança se alguma solução melhor for encontrada (ver Subseção 2.2.3).

A partir dos três movimentos para o ALWABP-2 são definidas nove estruturas de vizinhança aninhadas ($k_{max} = 9$). Um vizinho é gerado a uma distância φ da solução corrente. Essa distância está relacionada com o número de tarefas ou de trabalhadores, dependendo do movimento selecionado. Neste trabalho utilizou-se três distâncias diferentes, gerando um vizinho a 10%, 15% ou 20% da solução corrente ($\varphi = 0,1; 0,15; 0,2$). A cada três estruturas de vizinhança a distância é aumentada.

Duas heurísticas de busca local (Trocar Trabalhador e Trocar Tarefa) foram utilizadas no VNS, sendo baseadas na troca da alocação entre trabalhador/estação e tarefa/estação. Essas heurísticas serão explicadas na Subseção 6.8.3.

6.8.1.4 Busca Local Iterativa

A partir de uma solução inicial aleatória, a Busca Local Iterativa (ILS) (ver Subseção 2.2.4) executa uma perturbação na solução corrente e aplica um método de busca local sobre a nova solução.

A perturbação é realizada escolhendo aleatoriamente um dos três movimentos definidos para o ALWABP-2 e aplicando uma quantidade de movimentos que está relacionada à uma porcentagem β do número de trabalhadores ou tarefas, novamente dependendo do movimento escolhido. A porcentagem β varia de 25% a 75% do tamanho do problema.

As heurísticas Trocar Trabalhador e Trocar Tarefa (ver Subseção 6.8.3) foram aplicadas para obter um ótimo local para a nova solução.

6.8.2 Processo de Agrupamento

As soluções geradas são enviadas para o processo de agrupamento que procura agrupa-las como uma solução conhecida, de acordo com a métrica de distância. O cluster que possuir a menor distância em relação à solução s_k é considerado o cluster mais similar (C_j). A nova informação da solução s_k deve causar uma perturbação (assimilação) no centro do cluster mais similar.

No processo de assimilação foi utilizado o método Reconexão por Caminho (PR), o qual parte do centro c_j em direção a solução s_k, analisando o caminho que interconecta essas soluções. Para gerar esses caminhos, movimentos são selecionados por meio da troca da alocação de uma tarefa i no centro c_j pela alocação da tarefa i na solução s_k, mudando a alocação tarefa/estação da solução corrente. O método termina quando 30% do caminho é analisado. A melhor solução encontrada no caminho será o novo centro c_j. A Figura 6.5 mostra um exemplo do PR aplicado ao ALWABP-2.

Após atualizar o centro c_j, é preciso conduzir uma análise do volume v_j, verificando se este cluster pode ser considerado promissor, ou seja, se o número de soluções agrupadas atingiu o limitante λ ($v_j \geq \lambda$). Neste trabalho utilizou-se $\lambda = 10$.

Então, se o volume v_j atingir λ e a heurística de busca local estiver obtendo sucesso nesse cluster (índice de ineficácia $r_j < 5$), uma intensificação é aplicada no centro

do cluster por meio das heurísticas de busca local. Caso contrário, se o índice de ineficácia r_j for maior que cinco, é necessário aplicar uma perturbação no centro c_j, trocando aleatoriamente a alocação de 30% das tarefas.

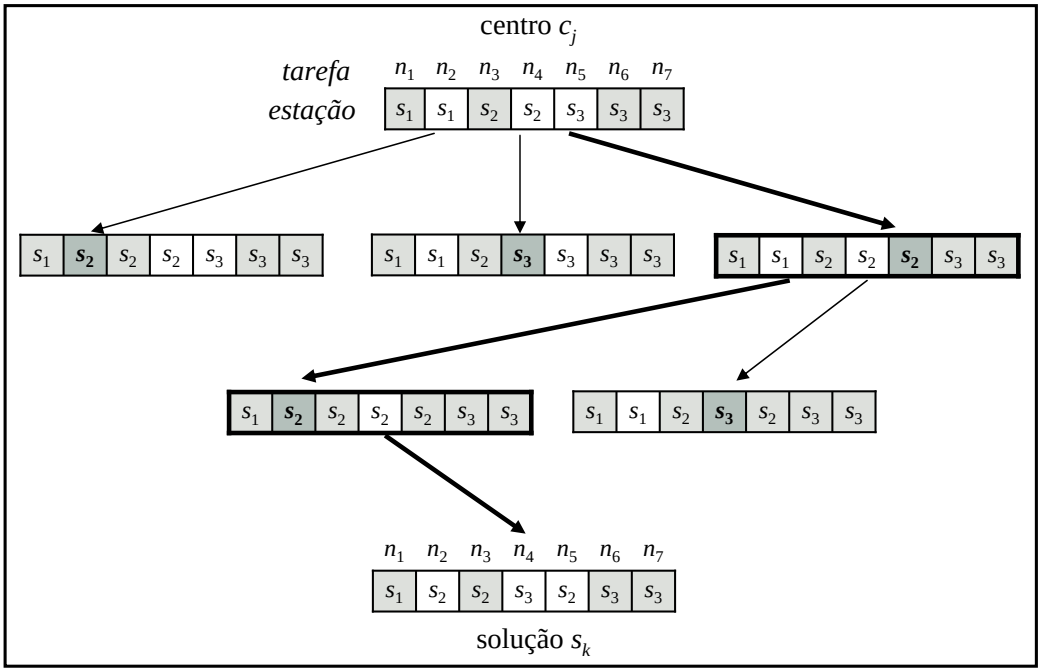

Figura 6.5 - Assimilação por caminho aplicada ao ALWABP.

6.8.3 Heurísticas de Busca Local

O componente de busca local é ativado quando um suposto cluster promissor é identificado. Sendo a busca intensificada na vizinhança do centro do cluster por meio do método Descida em Vizinhança Variável (VND) (ver Seção 3.5).

Três tipos de movimentos são relevantes em um método de busca local para o ALWABP-2: transferir tarefas de uma estação para outra, trocar tarefas entre estações e trocar trabalhadores entre estações. Sendo assim, o VND utiliza três heurísticas baseadas nesses movimentos.

As heurísticas do VND são:

- Trocar Tarefa: realizar o melhor movimento de trocar duas tarefas que estão atribuídas à diferentes estações;

- Transferir Tarefa: realizar o melhor movimento de transferir uma tarefa de uma estação para outra;
- Trocar Trabalhador: realizar o melhor movimento de trocar a posição de dois trabalhadores.

As heurísticas são executadas em sequência e se uma solução melhor for obtida, o VND retorna para a primeira heurística continuando a busca da melhor solução. A condição de parada do VND é que não existam mais melhoras para a solução corrente. O centro do cluster é atualizado se a solução encontrada pelo VND for melhor que o centro atual. Mas, o índice de ineficácia só será atualizado caso o VND encontre uma solução que seja a melhor até o momento neste cluster.

6.9 Resultados Computacionais

6.9.1 Instâncias Geradas para Teste

Pelo fato do ALWABP ser um problema inovador, não existe um conjunto padrão de instâncias que possam ser usados para avaliar o CS. Portanto, foram geradas instâncias baseadas em uma coleção de problemas clássicos do SALBP (HOFFMANN, 1990), disponíveis no site de pesquisa sobre balanceamento de linhas de produção (`http://www.assembly-line-balancing.de`).

Os problemas originais para gerar as instâncias para o ALWABP foram cuidadosamente selecionados dessa coleção. Foram incluídos problemas com diferentes tamanhos (número de tarefas) e propriedades estruturais nas redes de precedências (*Order Strength - OS*). O parâmetro OS define quão complexa é uma rede de precedência do ALWABP, quanto maior o valor de OS mais relações de precedências existem entre as tarefas.

A rede de precedência foi preservada para cada problema e os tempos originais de operação das tarefas foram utilizados para o primeiro trabalhador. Os novos tempos de operação dos outros trabalhadores foram aleatoriamente gerados a partir dos tempos do primeiro. Com base na experiência obtida por Miralles et al. (2007) nos CTD's define-se que a variação entre os tempos gerados não deve ser maior que três vezes o tempo original. Quando um trabalhador está acima desse limite, assume-se que essa tarefa não deveria ser atribuída ao trabalhador, sendo definido um tempo infinito para esse trabalhador executar a tarefa. Portanto,

diferentes porcentagens de incompatibilidades na matriz tarefa/trabalhador também são definidas. As instâncias foram geradas levando em consideração os seguintes fatores:

- relação entre o número de tarefas e o número de trabalhadores (tamanho da matriz tarefa/trabalhador).
- variabilidade dos tempos de uma tarefa para cada trabalhador.
- porcentagem de incompatibilidades tarefa/trabalhador.

Para o primeiro fator dois níveis foram definidos, número de tarefas quatro vezes maior que o número de trabalhadores e número de tarefas sete vezes maior que o número de trabalhadores. Os diferentes tempos de operação para cada tarefa i foram aleatoriamente gerados de uma distribuição uniforme com a variação selecionada de acordo com o tempo original (t_{1i}). Assim, são definidos dois níveis para a variabilidade dos tempos de operação utilizando a distribuição U[1,t_{1i}] e U[1,3t_{1i}], obtendo uma variabilidade de tempo pequena (L1) e grande (H3), respectivamente. Por fim, o terceiro fator é a porcentagem de incompatibilidade na matriz tarefa/trabalhador que foi definida como 10% (I10) e 20% (I20).

Portanto, as instâncias foram criadas de acordo com cinco fatores:

- *Tam*: número de tarefas;
- *OS*: complexidade da rede de precedência;
- *NrT*: número de trabalhadores;
- *Var*: variabilidade dos tempos de operação;
- *Inc*: % de incompatibilidades tarefa/trabalhador;

Todos os fatores possuem dois níveis, sendo assim, tem-se 2^5 combinações possíveis. Gerando 10 instâncias para cada combinação tem-se um conjunto de 320 instâncias para o ALWABP, que permitem extrair conclusões sobre o comportamento do método CS diante de diferentes tipos de variações encontradas no ALWABP.

A Tabela 6.1 mostra os problemas selecionados e as variações geradas.

Tabela 6.1 - ALWABP: Características das instâncias testadas.

problemas	***Tam***	***OS***	***NrT***	***Var***	***Inc***
Heskia	28	22,49	4; 7	L1; H3	I10; I20
Roszieg	25	71,67	4; 6	L1; H3	I10; I20
Wee-Mag	75	22,67	10; 17	L1; H3	I10; I20
Tonge	70	59,42	11; 19	L1; H3	I10; I20

6.9.2 Resultados

O CS para o ALWABP-2 foi codificado em C++ e os testes computacionais foram executados em um PC com processador Pentium 4 2,6 GHz e memória de 1 GB. Os experimentos foram empreendidos com o objetivo de evidenciar a qualidade dos resultados do CS, mostrando que este pode ser competitivo para resolver o ALWABP-2.

As tabelas a seguir apresentam os resultados dos testes computacionais realizados neste trabalho. Em razão do número elevado de instâncias, optou-se por apresentar os valores médios de cada 10 instâncias com as mesmas características. Essas instâncias e os resultados completos estão disponíveis em `http://www.lac.inpe.br/~lorena/intancias.html`.

Para avaliar a qualidade das soluções providas pelo CS, executou-se o *software* comercial CPLEX versão 10.1 (ILOG, 2006) para encontrar limitantes para as 320 instâncias. Como era esperado, devido à natureza NP-*hard* do problema, o CPLEX encontrou grandes dificuldades para resolver as instâncias maiores.

A Tabela 6.2 apresenta os resultados obtidos pelo CPLEX, mostrando a solução inteira, a solução não inteira, o tempo computacional e o *gap* obtido pelo CPLEX entre os limitantes inferior e superior. O CPLEX encontra a solução ótima para todas as instâncias *Roszieg* e *Heskia*. Porém, é sensível em relação à variação do número de trabalhadores, uma vez que, o tamanho da matriz tarefa/trabalhador é aumentado e consequentemente a complexidade do problema também. Os resultados do CPLEX para as instâncias *Tonge* e *Wee-Mag* não foram satisfatórios, sendo que,

os *gaps* entre os limitantes superiores e inferiores foram maiores que 90% para todas as instâncias. Para as instâncias *Tonge*, o CPLEX termina a execução em razão do estouro de memória do computador em 79 das 80 instâncias testadas. Além disso, para duas instâncias são obtidas soluções inviáveis devido à alocação de trabalhador e tarefa incompatível, e para outras duas instâncias não foi possível obter solução inteira. No problema *Wee-Mag*, o CPLEX também é interrompido em razão do estouro de memória em 68 das 80 instâncias testadas, obtendo soluções inviáveis em 11 instâncias e não obtendo solução inteira em outras 13 instâncias. Nas instâncias nas quais não ocorreu estouro de mémoria, o CPLEX foi interrompido devido ao tempo máximo de processamento definido como 10^5 segundos.

As Tabelas 6.3 - 6.7 apresentam os resultados do CS aplicado ao ALWABP-2, com diferentes geradores de soluções para o processo de agrupamento. As tabelas possuem as seguintes colunas:

- **melhor**: melhor solução conhecida;
- **sol***: melhor solução encontrada pelo método;
- **desvio**: erro relativo entre a solução média e a melhor solução encontrada pelo método (ver Equação 4.7);
- **gap**: erro relativo entre a melhor solução do método e a melhor solução conhecida, (ver Equação 4.8);
- **T***: tempo médio gasto pelo método para encontrar a melhor solução (em segundos);
- **T**: tempo médio de execução do método (em segundos);
- **delta**: erro relativo entre a melhor solução do CS e a melhor solução do método gerador de soluções (ver Equação 4.9).

Os valores em negrito mostram as instâncias nas quais o CS encontrou as melhores soluções conhecidas em termos de função objetivo. Para avaliar a robustez do método, o CS foi executado 20 vezes para cada instância.

O CS aplicado ao ALWABP não obteve desempenho semelhante ao dos outros problemas tratados nesta tese. Apesar de ter encontrado a solução ótima para todas as instâncias menores (*Roszieg* e *Heskia*) e soluções melhores que as obtidas pelo CPLEX para as instâncias maiores (*Tonge* e *Wee-Mag*), o comportamento do CS foi diferente para cada método gerador de soluções. Outro problema encontrado nesta abordagem foram os valores de desvio, caracterizando que o método não foi robusto. Este problema, em parte, é causado pelo fato dos tempos de operação entre trabalhadores e tarefas serem pequenos. Outras causas para esses resultados não satisfatórios serão discutidas na próxima seção.

Todas as versões do CS obtiveram bons resultados para as instâncias *Roszieg* e *Heskia*, encontrando as soluções ótimas ou soluções próximas destas. Porém, utilizando o SA como gerador de soluções obtêm-se os melhores resultados, encontrando a solução ótima em 157 das 160 instâncias. Nessas instâncias as meta-heurísticas também encontraram bons resultados, o que pode ser observado pelos baixos valores na coluna *delta*. Diferente do CPLEX, no CS não ocorreu variações de tempo significativas para instâncias com diferentes quantidades de trabalhadores.

O CS com SA também obteve os melhores resultados para as instâncias *Tonge*, encontrando em poucos segundos de execução soluções melhores que as obtidas pelo CPLEX. As demais versões tiveram *gaps* de até 30% em relação a melhor solução conhecida.

Para as instâncias *Wee-Mag*, novamente o CS com SA obteve os melhores resultados, mas o CS com GA também encontrou resultados satisfatórios. Por outro lado, as versões do CS com VNS e ILS foram muito ruins, obtendo *gaps* muito altos.

As meta-heurísticas VNS e ILS foram as que tiveram os melhores desempenhos, enquanto, SA e GA não conseguiram encontrar soluções viáveis para as instâncias *Tonge*. Porém, o CS utilizando as meta-heurísticas VNS e ILS como geradores de soluções obteve os piores resultados, tendo *gaps* médios acima de 20%.

Os demais fatores utilizados para gerar as instâncias de teste (*Inc*: porcentagem de incompatibilidades tarefa/trabalhador; *Var*: variabilidade dos tempos; *OS*: complexidade da rede de precedência) não interferem no comportamento do CS.

Tabela 6.2 - ALWABP: Resultados Computacionais do CPLEX.

				CPLEX			
	NrT	**Var**	**Inc**	**solução inteira**	**solução não inteira**	**tempo**	**gap-CPLEX**
		L1	I10	**20,1**	20,1	4,7	0,0
	4		I20	**31,5**	31,5	3,8	0,0
		H3	I10	**28,1**	28,1	5,5	0,0
			I20	**28,0**	28,0	4,3	0,0
Roszieg							
		L1	I10	**9,7**	9,7	332,2	0,0
	6		I20	**11,0**	11,0	281,0	0,0
		H3	I10	**16,0**	16,0	390,1	0,0
			I20	**15,1**	15,1	686,8	0,0
		L1	I10	**102,3**	102,3	6,6	0,0
	4		I20	**122,6**	122,6	5,8	0,0
		H3	I10	**172,5**	172,5	6,7	0,0
			I20	**171,2**	171,2	7,3	0,0
Heskia							
		L1	I10	**34,9**	34,9	177,4	0,0
	7		I20	**42,6**	42,6	216,4	0,0
		H3	I10	**75,2**	75,2	260,9	0,0
			I20	**67,2**	67,2	341,8	0,0
			I10	144,6	10,6	20223,0	92,7
	10		I20	162,4	12,0	23863,7	92,5
		H3	I10	276,7	16,4	18286,2	94,0
			I20	292,8	18,6	21241,9	93,6
Tonge							
		L1	I10	107,2	1,6	57480,3	98,4
	17		I20	133,8	1,7	44656,3	98,8
		H3	I10	231,4	2,6	54316,3	98,8
			I20	248,4	2,6	57772,4	99,0
			I10	42,4	2,8	17884,9	93,4
	11		I20	50,8	3,4	21507,3	93,3
		H3	I10	77,4	4,9	17227,8	93,6
			I20	72,9	4,7	17039,1	93,6
Wee-Mag							
		L1	I10	43,8	0,8	33323,7	98,5
	19		I20	161,0	0,9	43084,6	99,7
		H3	I10	95,6	0,9	51772,6	99,3
			I20	88,8	0,9	56884,9	99,3

Tabela 6.3 - ALWABP: Resultados Computacionais do CS-GA.

					CS-GA					GA				
	NrT	**Var**	**Inc**	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
		L1	I10	20,1	**20,1**	1,5	0,0	1,4	3,1	20,4	10,9	1,6	2,1	1,6
	4		I20	31,5	**31,5**	3,5	0,0	1,3	3,1	33,9	20,2	8,8	2,1	8,8
		H3	I10	28,1	**28,1**	0,7	0,0	1,5	3,0	28,4	4,9	1,0	2,1	1,0
			I20	28,0	**28,0**	0,1	0,0	1,3	3,1	28,3	4,5	1,1	2,1	1,1
Roszieg														
		L1	I10	9,7	**9,8**	5,5	1,1	2,0	3,6	10,5	8,4	8,4	2,3	7,3
	6		I20	11,0	11,2	13,6	2,1	2,1	3,8	13,2	205,9	20,0	2,3	17,8
		H3	I10	16,0	**16,0**	5,7	0,0	2,3	3,7	17,4	10,2	8,6	2,4	8,6
			I20	15,1	**15,1**	5,7	0,0	2,1	3,7	16,1	30,3	6,8	2,4	6,8
		L1	I10	102,3	**102,3**	2,1	0,0	2,5	4,5	103,6	4,6	1,1	2,7	1,1
	4		I20	122,6	**122,6**	1,9	0,0	2,3	4,5	125,1	7,2	2,0	2,7	2,0
		H3	I10	172,5	172,8	2,2	0,2	2,2	4,5	174,6	6,5	1,2	2,7	1,0
			I20	171,2	**171,3**	1,9	0,0	2,6	4,5	171,9	5,4	0,4	2,7	0,4
Heskia														
		L1	I10	34,9	35,5	13,1	1,6	4,0	6,2	41,6	13,3	19,9	3,1	18,2
	7		I20	42,6	43,7	5,4	2,6	3,8	6,3	47,7	12,1	12,6	3,2	9,7
		H3	I10	75,2	75,7	5,8	0,9	3,4	6,2	80,3	9,0	8,7	3,2	7,7
			I20	67,2	69,5	8,5	3,6	3,6	6,2	74,7	12,2	11,8	3,2	7,9
		L1	I10	103,2	113,6	57,9	9,5	81,7	127,3	385,8	164,5	288,6	62,9	258,0
	10		I20	125,9	133,7	80,5	6,3	77,8	128,7	823,5	30,4	560,8	62,5	524,3
		H3	I10	178,6	193,7	43,9	8,4	84,3	128,2	621,7	70,9	247,4	63,7	220,2
			I20	182,8	198,0	47,0	8,5	82,6	128,7	645,3	56,9	257,3	63,7	231,1
Tonge														
		L1	I10	52,5	59,1	116,0	13,7	98,5	178,0	469,5	246,0	781,5	75,6	662,1
	17		I20	64,5	83,1	107,3	31,5	105,8	181,3	926,7	13,5	1360,4	76,2	1023,2
		H3	I10	98,4	118,7	85,2	21,4	103,1	180,0	754,0	53,4	667,8	77,3	531,8
			I20	107,0	125,4	85,8	18,1	100,8	180,7	858,6	32,4	718,5	77,4	589,8
		L1	I10	31,8	34,4	14,7	8,6	82,4	157,1	65,9	163,5	107,7	65,1	91,9
	11		I20	37,7	40,3	13,8	7,1	86,6	162,7	82,6	264,5	120,3	66,7	105,6
		H3	I10	55,6	59,1	16,0	6,3	93,6	160,6	113,4	84,3	104,7	67,1	92,6
			I20	53,4	57,6	15,8	8,0	95,7	160,5	112,0	108,4	110,9	67,2	95,4
Wee-Mag														
		L1	I10	16,7	17,7	25,7	6,3	124,8	215,3	42,0	579,5	152,1	83,7	137,6
	19		I20	19,4	20,7	21,7	7,0	124,4	223,0	52,9	697,1	173,2	86,4	156,0
		H3	I10	28,3	30,3	20,5	7,5	140,1	220,0	72,5	313,5	157,8	87,0	140,4
			I20	27,5	29,3	28,8	6,7	135,0	221,4	73,4	404,3	168,4	87,2	151,9
média						**26,8**	**5,8**	**51,7**	**88,2**		**114,0**	**190,4**	**37,8**	**159,8**

Tabela 6.4 - ALWABP: Resultados Computacionais do CS-SA.

					CS-SA					SA				
	NrT	**Var**	**Inc**	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
		L1	I10	20,1	**20,1**	0,5	0,0	2,1	5,5	20,1	1,2	0,0	4,7	0,0
	4		I20	31,5	**31,5**	6,1	0,0	2,1	5,4	31,8	28,8	0,8	4,7	0,8
		H3	I10	28,1	**28,1**	0,1	0,0	2,1	5,5	28,1	0,3	0,0	4,7	0,0
			I20	28,0	**28,0**	0,4	0,0	1,8	5,5	28,0	0,9	0,0	4,7	0,0
Roszieg														
		L1	I10	9,7	**9,7**	5,0	0,0	3,3	6,2	9,8	4,7	1,1	5,0	1,1
	6		I20	11,0	**11,0**	8,4	0,0	3,5	6,2	11,0	11,0	0,0	5,0	0,0
		H3	I10	16,0	**16,0**	1,7	0,0	3,5	6,2	16,0	2,5	0,0	5,0	0,0
			I20	15,1	**15,1**	1,8	0,0	3,3	6,2	15,1	3,3	0,0	5,0	0,0
		L1	I10	102,3	**102,3**	1,0	0,0	3,0	6,2	102,3	2,1	0,0	5,1	0,0
	4		I20	122,6	**122,6**	0,7	0,0	2,6	6,2	122,7	2,5	0,1	5,1	0,1
		H3	I10	172,5	**172,5**	0,4	0,0	2,6	6,2	172,5	0,8	0,0	5,1	0,0
			I20	171,2	**171,2**	0,3	0,0	2,9	6,2	171,2	0,5	0,0	5,1	0,0
Heskia														
		L1	I10	34,9	35,0	3,2	0,3	4,6	7,6	35,0	5,5	0,3	5,6	0,0
	7		I20	42,6	42,9	3,2	0,7	4,1	7,6	43,0	3,7	1,0	5,6	0,3
		H3	I10	75,2	**75,2**	1,7	0,0	3,5	7,6	75,2	2,1	0,0	5,6	0,0
			I20	67,2	**67,2**	4,5	0,0	4,2	7,6	67,2	5,8	0,0	5,6	0,0
		L1	I10	103,2	109,6	42,7	6,4	43,0	62,0	206,7	586,6	97,4	20,2	78,6
	10		I20	125,9	133,1	57,7	5,7	41,8	61,9	138,1	576,8	9,4	20,1	3,3
		H3	I10	178,6	183,3	44,6	2,5	44,3	61,8	185,5	372,2	3,7	20,4	1,1
			I20	182,8	192,1	43,5	4,8	43,3	61,9	192,9	370,0	5,2	20,4	0,4
Tonge														
		L1	I10	52,5	55,6	74,0	6,4	67,4	96,6	343,1	1076,6	528,8	24,7	456,9
	17		I20	64,5	70,4	92,0	8,8	63,5	96,3	533,0	750,9	679,2	24,7	581,4
		H3	I10	98,4	108,0	72,8	9,6	64,9	96,2	467,1	486,8	367,1	25,1	309,8
			I20	107,0	118,6	57,9	10,1	64,8	96,1	668,2	301,5	503,5	25,1	436,2
		L1	I10	31,8	31,9	16,4	0,3	55,9	76,7	31,9	103,4	0,3	21,0	0,0
	11		I20	37,7	37,9	12,7	0,5	56,0	76,3	38,0	29,0	0,8	21,3	0,3
		H3	I10	55,6	55,9	14,4	0,6	56,3	76,5	55,9	61,1	0,6	21,4	0,0
			I20	53,4	53,6	16,8	0,3	54,7	76,5	53,6	28,8	0,3	21,3	0,0
Wee-Mag														
		L1	I10	16,7	17,0	26,8	1,7	76,3	111,4	17,4	3065,5	4,1	26,5	2,4
	19		I20	19,4	19,5	25,1	0,6	75,7	111,6	19,7	2412,4	1,6	26,9	1,0
		H3	I10	28,3	28,9	26,1	2,0	78,8	111,9	29,5	1479,0	4,0	27,1	2,0
			I20	27,5	**27,5**	29,1	0,0	81,9	111,6	28,1	1642,4	2,0	27,1	2,0
média						**21,6**	**1,9**	**31,8**	**46,5**		**419,3**	**69,1**	**14,2**	**58,7**

Tabela 6.5 - ALWABP: Resultados Computacionais do CS-VNS.

					CS-VNS					VNS				
	NrT	**Var**	**Inc**	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
		L1	I10	20,1	**20,1**	0,8	0,0	2,3	5,9	20,1	3,4	0,0	5,0	0,0
	4		I20	31,5	31,5	1,7	0,0	1,8	5,8	31,5	4,6	0,0	5,0	0,0
		H3	I10	28,1	28,1	0,8	0,0	1,9	5,8	28,1	2,5	0,0	5,0	0,0
			I20	28,0	28,0	0,8	0,0	2,0	5,9	28,0	2,7	0,0	5,0	0,0
Roszieg														
		L1	I10	9,7	10,4	9,7	7,3	3,6	8,2	10,5	17,8	8,4	7,0	1,1
	6		I20	11,0	11,6	17,7	5,7	3,6	8,3	12,8	19,8	16,7	7,0	10,7
		H3	I10	16,0	16,3	11,5	1,8	3,9	8,2	17,2	16,9	7,4	7,0	5,6
			I20	15,1	15,4	11,2	2,1	4,0	8,2	16,2	14,4	7,5	7,0	5,3
		L1	I10	102,3	**102,3**	1,0	0,0	3,8	7,9	102,4	2,6	0,1	6,6	0,1
	4		I20	122,6	122,7	0,7	0,1	3,0	7,9	122,8	2,1	0,2	6,6	0,1
		H3	I10	172,5	**172,5**	0,8	0,0	3,4	8,0	172,7	2,7	0,1	6,6	0,1
			I20	171,2	**171,3**	0,8	0,0	3,2	8,0	171,7	1,7	0,3	6,6	0,3
Heskia														
		L1	I10	34,9	36,6	13,7	4,6	6,9	12,2	40,6	15,3	16,7	10,0	11,6
	7		I20	42,6	43,9	6,6	3,2	6,5	12,3	45,6	12,3	6,9	10,0	3,6
		H3	I10	75,2	76,3	6,0	1,9	5,4	12,3	79,4	11,5	7,1	10,0	5,1
			I20	67,2	69,9	8,3	4,2	6,6	12,4	74,3	14,4	10,8	10,1	6,2
		L1	I10	103,2	125,4	24,1	21,6	110,3	186,5	166,5	88,0	62,0	155,1	33,4
	10		I20	125,9	152,7	36,4	21,8	117,2	189,7	217,9	115,8	73,8	156,0	42,9
		H3	I10	178,6	206,5	25,8	15,5	113,3	188,7	273,5	87,4	52,8	155,5	32,2
			I20	182,8	219,6	30,1	20,2	113,8	188,8	301,7	68,9	66,9	155,8	38,9
Tonge														
		L1	I10	52,5	70,8	40,2	36,6	174,5	279,1	112,4	271,0	118,7	230,8	60,0
	17		I20	64,5	83,4	65,4	30,3	182,2	281,0	148,8	260,3	133,6	231,4	81,5
		H3	I10	98,4	124,6	41,2	26,8	192,2	280,1	195,0	148,0	98,7	229,6	57,9
			I20	107,0	125,2	39,8	18,0	178,1	279,6	211,5	130,4	98,7	229,7	70,1
		L1	I10	31,8	45,2	16,3	42,7	121,1	223,9	55,8	19,6	76,5	185,2	23,7
	11		I20	37,7	53,0	19,3	40,9	110,4	225,8	67,4	33,1	80,1	183,5	27,5
		H3	I10	55,6	79,0	14,5	42,4	124,5	228,9	96,8	19,6	74,3	184,5	22,7
			I20	53,4	76,9	16,1	44,5	116,1	228,1	94,9	20,1	78,6	184,3	23,7
Wee-Mag														
		L1	I10	16,7	30,4	15,5	82,0	198,1	359,3	36,5	25,2	118,9	306,0	20,5
	19		I20	19,4	32,8	19,6	69,9	208,5	365,4	42,8	43,8	121,2	305,3	30,5
		H3	I10	28,3	52,1	13,1	85,0	209,4	364,4	62,1	27,9	121,5	302,6	20,0
			I20	27,5	50,0	17,3	82,7	195,2	364,8	64,8	33,6	137,2	302,7	29,7
média						**16,5**	**22,2**	**79,0**	**136,6**		**48,0**	**49,9**	**112,9**	**20,8**

Tabela 6.6 - ALWABP: Resultados Computacionais do CS-ILS.

					CS-ILS					ILS				
	NrT	**Var**	**Inc**	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
		L1	I10	20,1	**20,1**	0,3	0,0	2,1	5,6	20,1	1,3	0,0	4,8	0,0
	4		I20	31,5	**31,5**	0,3	0,0	1,5	5,5	31,5	1,3	0,0	4,7	0,0
		H3	I10	28,1	**28,1**	0,4	0,0	2,2	5,5	28,1	1,2	0,0	4,7	0,0
			I20	28,0	**28,0**	0,3	0,0	1,8	5,7	28,0	1,0	0,0	4,8	0,0
Roszieg														
		L1	I10	9,7	10,1	11,1	4,2	4,4	7,7	10,6	11,3	9,8	6,3	5,3
	6		I20	11,0	11,5	14,6	5,1	4,1	7,8	12,8	11,6	17,0	6,4	11,4
		H3	I10	16,0	16,4	7,9	2,6	4,9	7,8	16,9	10,7	5,8	6,3	3,2
			I20	15,1	15,3	8,4	1,4	4,8	7,8	15,6	13,4	3,4	6,4	2,0
		L1	I10	102,3	**102,3**	0,7	0,0	3,5	7,6	102,8	1,2	0,5	6,3	0,5
	4		I20	122,6	**122,6**	0,5	0,0	3,0	7,6	122,7	1,2	0,1	6,3	0,1
		H3	I10	172,5	**172,5**	0,9	0,0	3,0	7,8	172,5	2,0	0,0	6,4	0,0
			I20	171,2	**171,3**	0,4	0,0	3,7	7,8	171,5	1,0	0,2	6,4	0,1
Heskia														
		L1	I10	34,9	37,6	12,9	7,4	7,4	11,4	41,3	16,9	18,4	9,3	10,4
	7		I20	42,6	43,9	7,4	3,1	6,0	11,6	47,0	9,4	10,5	9,3	7,1
		H3	I10	75,2	75,6	9,2	0,6	5,7	11,6	81,2	8,3	10,5	9,4	9,8
			I20	67,2	71,9	6,6	7,3	6,4	11,7	76,1	11,5	14,1	9,4	6,4
		L1	I10	103,2	113,2	18,0	9,9	128,1	187,3	163,4	300,9	59,6	142,9	46,2
	10		I20	125,9	131,7	21,0	5,0	128,5	191,0	187,5	264,2	51,2	145,1	43,9
		H3	I10	178,6	189,3	16,4	6,0	131,2	189,7	268,8	187,2	50,5	143,9	42,1
			I20	182,8	194,0	18,6	6,5	134,8	190,2	267,1	178,3	46,9	144,4	38,6
Tonge														
		L1	I10	52,5	71,1	23,5	37,5	171,9	267,5	128,0	463,5	148,3	201,7	80,7
	17		I20	64,5	83,0	32,7	30,1	172,9	272,7	158,2	324,2	146,0	204,0	91,4
		H3	I10	98,4	117,5	36,2	19,9	184,4	271,6	233,0	255,9	137,3	203,6	99,7
			I20	107,0	127,2	27,5	19,7	177,0	271,0	238,2	241,1	125,9	203,0	87,5
		L1	I10	31,8	52,3	15,0	65,2	104,3	203,6	77,0	20,4	144,2	153,8	47,6
	11		I20	37,7	58,7	20,8	56,2	127,3	205,1	87,8	28,0	133,6	153,9	50,1
		H3	I10	55,6	92,7	13,5	67,0	146,7	207,9	122,9	16,5	121,8	155,3	33,0
			I20	53,4	89,3	16,1	67,4	141,5	207,7	118,9	19,8	124,4	155,3	33,9
Wee-Mag														
		L1	I10	16,7	35,2	31,1	110,7	72,6	309,3	64,6	33,5	288,2	236,6	87,3
	19		I20	19,4	41,3	28,8	113,1	112,0	311,7	75,9	36,4	294,1	236,4	86,9
		H3	I10	28,3	60,9	29,2	116,7	108,9	316,1	103,3	31,2	267,4	238,7	71,0
			I20	27,5	63,8	26,1	133,6	99,9	315,3	107,6	30,4	290,9	237,8	70,0
média						**14,3**	**28,0**	**69,0**	**126,5**		**79,2**	**78,8**	**95,7**	**33,3**

Tabela 6.7 - ALWABP: Resultados Computacionais do CS-Aleatório.

					CS-Aleatório					Aleatório				
	NrT	**Var**	**Inc**	**melhor**	**sol***	**desvio**	**gap**	**T***	**T**	**sol***	**desvio**	**gap**	**T**	**delta**
		L1	I10	20,1	**20,1**	1,4	0,0	1,6	4,4	1018,1	49,7	5016,4	3,8	5016,4
	4		I20	31,5	**31,5**	4,8	0,0	1,3	4,4	2317,0	62,3	6138,7	3,7	6138,7
		H3	I10	28,1	**28,1**	0,9	0,0	1,5	4,4	731,8	582,5	2412,6	3,7	2412,6
			I20	28,0	**28,0**	0,7	0,0	1,5	4,4	929,2	230,0	3261,5	3,7	3261,5
Roszieg														
		L1	I10	9,7	9,9	9,3	2,1	2,2	4,4	919,5	262,8	9679,6	3,3	9475,2
	6		I20	11,0	11,4	19,2	3,7	2,1	4,4	1521,5	73,5	14096,5	3,3	13672,5
		H3	I10	16,0	16,1	6,2	0,6	2,3	4,4	1025,1	37,2	6373,7	3,3	6335,8
			I20	15,1	15,2	6,4	0,8	2,3	4,4	1022,6	35,9	6718,7	3,3	6662,1
		L1	I10	102,3	102,7	4,2	0,4	2,1	4,5	313,7	292,9	189,8	3,4	188,9
	4		I20	122,6	122,7	5,0	0,1	2,0	4,5	850,5	183,4	587,2	3,4	586,5
		H3	I10	172,5	173,1	2,8	0,3	2,3	4,5	409,0	165,9	131,8	3,3	131,1
			I20	171,2	171,2	5,5	0,0	2,7	4,5	421,2	186,2	144,4	3,3	144,4
Heskia														
		L1	I10	34,9	39,2	14,8	12,5	3,0	4,5	799,1	144,4	2250,3	2,4	1958,1
	7		I20	42,6	44,3	8,9	3,8	2,6	4,5	1195,9	100,8	2854,5	2,3	2752,3
		H3	I10	75,2	78,0	11,9	4,9	2,3	4,5	781,9	101,4	987,0	2,3	931,0
			I20	67,2	72,8	11,0	8,8	2,6	4,5	845,8	82,7	1133,3	2,3	1061,5
		L1	I10	103,2	125,8	19,0	21,7	42,8	57,9	6007,2	47,9	5821,1	10,2	4802,7
	10		I20	125,9	153,0	40,3	22,3	43,5	62,8	12467,4	29,5	9792,6	11,4	8043,1
		H3	I10	178,6	216,4	17,0	21,2	43,6	59,3	6711,3	28,7	3653,5	10,6	2997,9
			I20	182,8	216,3	21,6	18,8	44,7	59,6	6557,2	42,6	3531,7	10,6	2967,6
Tonge														
		L1	I10	52,5	62,7	30,2	21,1	71,5	94,0	5198,4	67,1	9893,8	13,0	8215,0
	17		I20	64,5	78,4	52,2	22,5	76,7	100,5	11482,4	12,8	17894,1	14,3	14590,4
		H3	I10	98,4	114,2	33,3	16,2	72,0	95,3	4859,3	60,4	4867,8	13,2	4213,3
			I20	107,0	122,9	27,1	15,4	72,6	95,3	5055,6	53,2	4670,0	13,3	4044,6
		L1	I10	31,8	40,1	13,9	26,4	46,5	65,2	5335,2	68,9	16848,0	10,4	13276,9
	11		I20	37,7	46,7	13,9	24,1	50,1	68,5	12593,4	32,4	33494,4	12,1	27000,3
		H3	I10	55,6	70,5	13,2	27,2	51,9	67,1	6295,2	51,5	11278,9	10,8	8872,8
			I20	53,4	67,3	14,9	26,3	51,2	67,1	6511,9	46,0	12062,1	10,7	9598,9
Wee-Mag														
		L1	I10	16,7	20,9	17,7	25,4	72,6	100,0	4193,9	90,8	25188,2	13,2	20133,0
	19		I20	19,4	24,3	15,4	25,8	72,6	103,8	11958,7	10,8	62011,1	15,1	49300,9
		H3	I10	28,3	34,8	20,8	23,3	77,9	102,4	4779,8	98,1	17278,0	13,9	13933,5
			I20	27,5	35,8	18,4	30,6	73,3	102,6	5062,9	91,3	18374,1	13,9	14033,1
média						**15,1**	**12,7**	**31,2**	**42,9**		**107,0**	**9957,4**	**7,7**	**8336,0**

6.10 Considerações Finais

Este Capítulo apresenta uma solução para o Problema de Balanceamento e Designação de Trabalhadores para Linha de Produção (ALWABP) utilizando o método Busca por Agrupamentos (CS). Os experimentos computacionais mostram que o CS é competitivo para resolver esse problema em um tempo computacional razoável. Porém, algumas melhoras precisam ser implementadas tornando o método mais robusto para resolver este problema.

Alguns dos motivos que levaram o CS a não obter soluções melhores para o ALWABP são a realização excessiva de movimentos inviáveis e o método de busca local não ter sido muito eficiente. A maioria das soluções analisadas no CS são soluções inviáveis, o que aumenta o espaço de busca e prejudica a análise de regiões promissoras. Pode-se notar também que a busca local conseguiu melhorar a solução que está no centro do cluster em poucas oportunidades em que foi aplicada, sendo que, o índice de sucesso da busca local foi muito maior nos outros problemas tratados nesta tese.

Apesar disso, os resultados do CS utilizando a meta-heurística SA como gerador de soluções obteve resultados muito satisfatórios, conseguindo boas soluções em um tempo computacional competitivo.

Na prática, esses tempos computacionais pequenos tornam viável um rápido balanceamento da linha de produção. Isto é muito importante se levar em consideração que, por causa do alto absentismo e dos exames psicológicos e físicos periódicos dos trabalhadores, o gerente do CTD conhece apenas no início de cada dia quais trabalhadores estão disponíveis. Portanto, abordagens como o CS, que proveem boas soluções em pouco tempo computacional, são muito interessantes para realizar um balanceamento diário da linha de produção. Procedimentos de balanceamento ágeis também são necessários para rotinas de rotação de trabalho, as quais são necessárias neste ambiente, obtendo múltiplas combinações facilmente.

Uma vantagem adicional do CS, que pode facilitar uma implementação real nos CTD's, é o fato de não utilizar nenhum *software* comercial. Desta forma, não acarreta maiores custos aos CTD's que são entidades sem fins lucrativos.

7 ANÁLISE DE DESEMPENHO DO CS

Este Capítulo apresenta um conjunto de experimentos computacionais que foram realizados com objetivo de estudar o comportamento do CS diante das variações de seus parâmetros e também com relação a cada componente do processo de agrupamento.

Por meio dos resultados destes testes é possível determinar as principais características que o CS precisa possuir para obter sucesso na resolução de problemas de otimização combinatória. Essas informações podem auxiliar futuras implementações do CS para outros problemas de otimização.

Para cada teste foi selecionado um problema e algumas instâncias, de modo que, possa ser realizada uma análise coerente em um tempo computacional razoável.

7.1 Desempenho do CS com Diferentes Números de Clusters

A quantidade de regiões na qual o CS divide o espaço de busca é determinado pelo número de clusters definido inicialmente. Desta forma, ter apenas um cluster significa tratar o espaço de busca como uma única região, e o cluster atinge o limitante λ a cada λ soluções geradas para o processo de agrupamento. Por outro lado, definir um número maior de clusters permite ao CS descobrir quais regiões são promissoras, intensificando a busca somente nestas regiões.

É importante notar que, o número de vezes que os clusters atingem o limitante λ não sofre alterações significativas independentemente do número de clusters, pois isto está relacionado com o número de soluções geradas para o processo de agrupamento e com o valor do limitante λ (como será visto na Seção 7.2).

Para analisar o comportamento do CS com diferentes números de clusters foi utilizado a abordagem do CS aplicada ao CPMP, utilizando para testes as instâncias *sjc*. Os números de clusters testados foram $|C| = \{1, 5, 10, 15, 20, 25\}$. Os demais parâmetros do CS não foram modificados.

A Figura 7.1 apresenta o gráfico com os *gaps* médios das cinco versões do CS, visto que, todas as versões apresentaram comportamento parecido. Os *gaps* são calculados entre as melhores soluções encontradas pelo CS e a solução ótima. Pode-se notar que a qualidade das soluções foi satisfatória com qualquer número de clusters. Sendo que,

o CS sempre encontra a solução ótima para as instâncias menores independentemente do número de clusters. Porém, o CS com 20 clusters obteve os melhores resultados para as instâncias maiores.

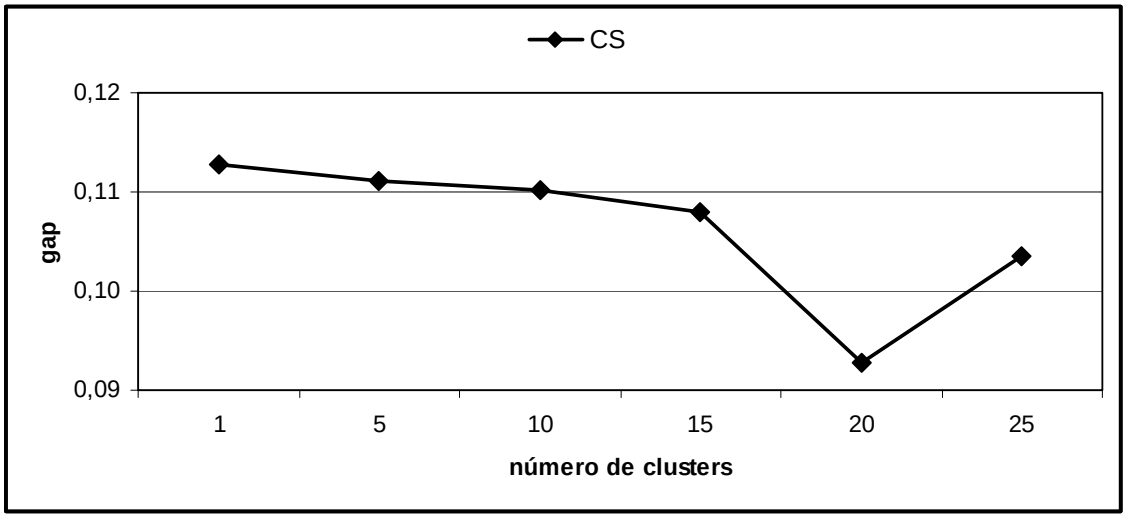

Figura 7.1 - Gráfico com os *gaps* do CS variando a quantidade de clusters.

Na Figura 7.2 é apresentado um gráfico, em escala logarítmica, com a eficiência do componente de busca local para as diferentes quantidades de clusters. Neste gráfico observa-se que, quanto maior o número de clusters maior é a eficiência da busca local, ou seja, a probabilidade do CS conseguir escapar de um ótimo local é maior quando se divide o espaço de busca em mais regiões.

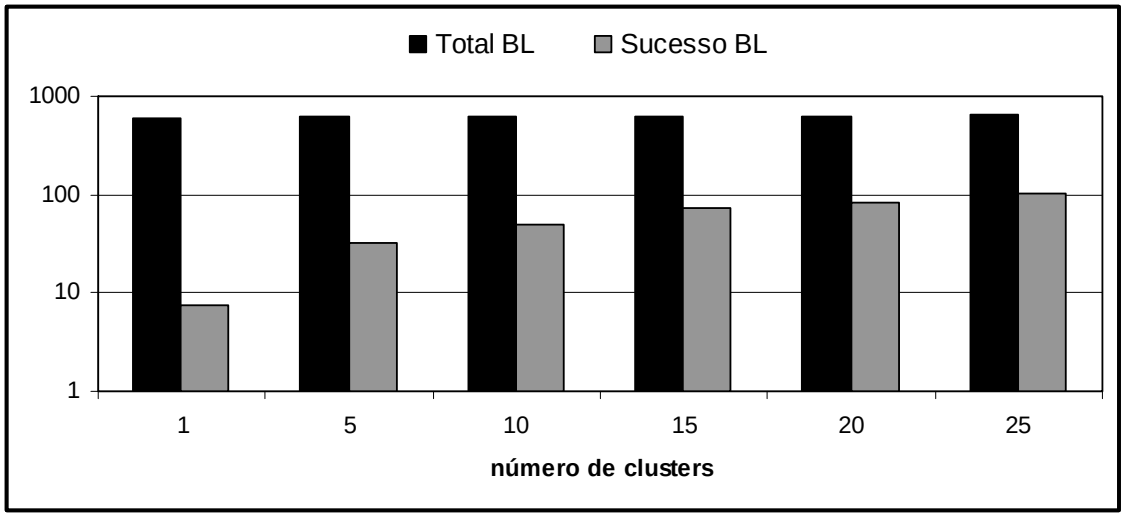

Figura 7.2 - Gráfico com a eficiência do componente de busca local variando a quantidade de clusters.

7.2 Desempenho do CS com Diferentes Limitantes de Busca Local

O valor do limitante λ tem influência na quantidade de vezes que o componente de busca local é executado. Portanto, quanto maior o valor de λ mais soluções serão necessárias para um cluster se tornar promissor e, então, menos procedimentos de busca local serão realizados.

Para analisar o desempenho do CS com diferentes valores de limitante λ foi utilizada a abordagem do CS aplicada ao PCTSP, fazendo uso das instâncias menores (até *pr152*). Os valores do limitante de busca local utilizados foram $\lambda = \{1, 5, 10, 15, 20, 25\}$. Se λ for igual a 1 será aplicado busca local no centro do cluster toda vez que uma solução for agrupada.

A Figura 7.3 ilustra o gráfico com o tempo computacional médio das cinco versões do CS com diferentes métodos geradores de solução. Todas as versões apresentaram comportamento semelhante, sendo assim, é apresentado um único gráfico para estas. Como era esperado, o tempo computacional do CS com $\lambda = 1$ foi muito superior aos demais valores testados, o que é explicado pela maior quantidade de busca local realizada. Assim, quanto maior o valor de λ menor é o tempo computacional do CS.

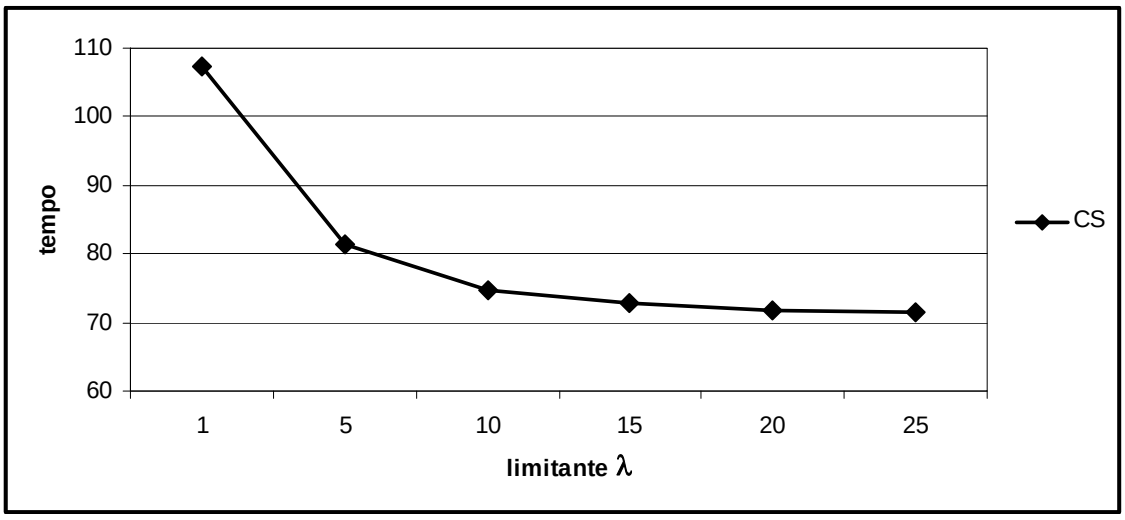

Figura 7.3 - Gráfico com os tempos computacionais do CS variando o valor de λ.

Na Figura 7.4 é apresentado o gráfico com o *gap* do CS, ou seja, o erro relativo entre a melhor solução encontrada pelo CS e a melhor solução conhecida para as

instâncias. Neste gráfico pode-se notar que, a aplicação de busca local em todas as soluções geradas para o processo de agrupamento ($\lambda = 1$) produz os melhores resultados. Entretanto, valores de λ iguais a 5, 10 e 15 também provêm resultados satisfatórios, tendo um *gap* um pouco maior que o CS com λ igual a 1. Sendo assim, é preciso levar em consideração o tempo computacional gasto em cada configuração. E, neste sentido, a abordagem com λ igual a 15 foi a mais competitiva, obtendo bons resultados em pouco tempo computacional.

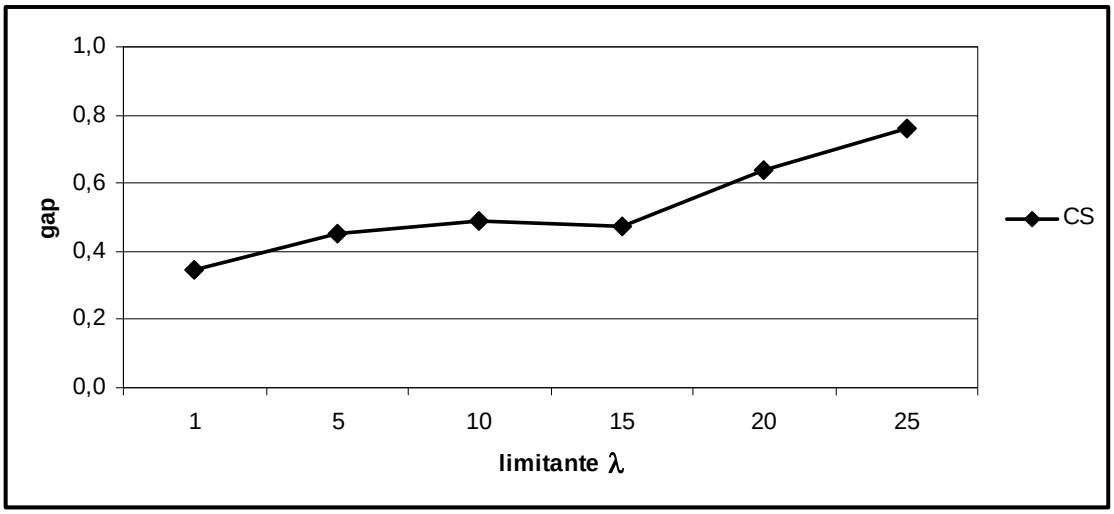

Figura 7.4 - Gráfico com os *gaps* do CS variando o valor de λ.

7.3 Desempenho do Método para Criar os Cluster Iniciais

Uma das modificações proposta neste trabalho foi a inclusão de uma fase inicial para criar os clusters, de forma que exista uma diversidade entre os clusters iniciais. Assim, possibilita-se que diferentes regiões do espaço de busca sejam representadas inicialmente. O objetivo deste método é obter uma divisão mais adequada do espaço de busca, fazendo com que o CS identifique com mais precisão regiões realmente promissoras e, assim, consiga convergir para soluções melhores.

Para verificar se tal objetivo foi alcançado testou-se o CS sem este método de gerar os clusters iniciais, ou seja, os clusters foram gerados aleatoriamente, não levando em consideração suas localizações no espaço de busca.

O desempenho do CS com as duas formas de gerar os clusters iniciais foi analisada

por meio da abordagem aplicada ao CCCP, utilizando para testes as instâncias *sjc*. O CS com e sem o método para criar os clusters com máxima diversidade possuem os mesmos componentes e valores de parâmetros.

A Figura 7.5 apresenta o gráfico, em escala logarítmica, com os *gaps* médios do CS com o método de máxima diversidade (*MaxDiversidade*) e sem este método (*Aleatório*), sendo testada cada uma das versões do CS com os diferentes métodos geradores de soluções para o processo de agrupamento. Observa-se que o CS com o método de máxima diversidade encontrou resultados melhores em todas as versões do CS, mostrando que este pré-processamento inicial foi benéfico ao CS.

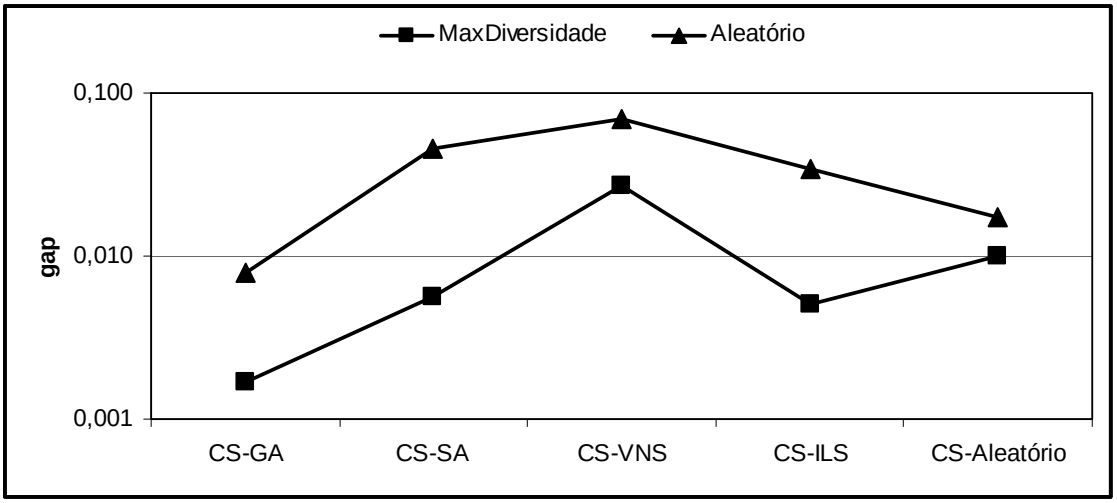

Figura 7.5 - Gráfico com os *gaps* dos CS's com diferentes métodos para gerar os clusters iniciais.

Na Figura 7.6 são apresentados os desvios médios do CS, representando o erro relativo entre a melhor e a média das soluções encontradas pelo CS. Neste gráfico é possível notar que o CS com o método de máxima diversidade foi superior ao CS no qual os clusters iniciais foram gerados aleatoriamente. Ou seja, além de obter melhores resultados, a fase inicial proposta neste trabalho também deixa o método mais robusto.

Outro dado observado nestes testes foi que, apesar do acréscimo de tempo do método de máxima diversidade, o tempo computacional do CS não sofreu mudanças significativas. Isto se deve ao fato de ter-se utilizado um método eficiente para resolver o problema de máxima diversidade, e também ao fato de uma boa configuração inicial auxiliar o processo de busca do CS.

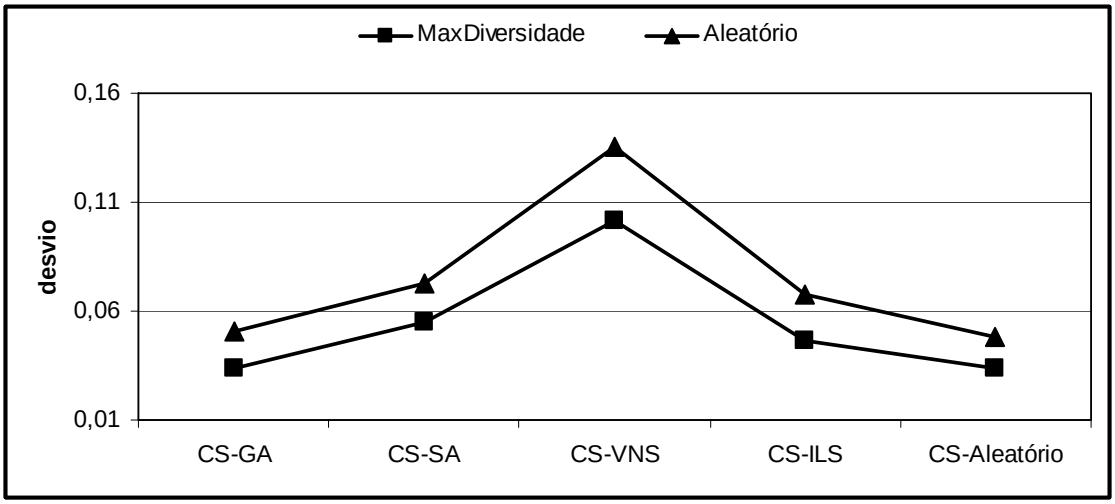

Figura 7.6 - Gráfico com os desvios dos CS's com diferentes métodos para gerar os clusters iniciais.

7.4 Desempenho do CS Aplicando Busca Local Aleatoriamente

Um teste interessante é a aplicação aleatória de busca local. Ao invés de descobrir um cluster promissor e intensificar a busca neste, é adicionada uma probabilidade de aplicar busca local no centro de um cluster quando este for ativado. Neste teste, a probabilidade de aplicar busca local foi de 5%. Desta forma, tem-se um número de execuções do componente de busca local semelhante nas duas abordagens do CS, com detecção de regiões promissoras e com busca aleatória.

Nesta análise foi utilizado o CS aplicado ao PCTSP, no qual foram executadas as instâncias até *pr152*. Os parâmetros do CS, assim como os métodos de busca local são os mesmos, a única diferença está relacionada ao momento no qual a busca local é acionada.

A Figura 7.7 apresenta o gráfico com os *gaps* (erro relativo entre a melhor solução conhecida e a melhor solução encontrada pelo método) de cada versão do CS com diferentes métodos geradores de solução. Pode-se observar que a estratégia de aplicar busca local nos clusters considerados promissores obteve resultados significativamente melhores que a estratégia de aplicar busca local aleatoriamente. Os *gaps* da primeira estratégia foram próximos a zero, enquanto, os *gaps* da estratégia aleatória são superiores a 3%.

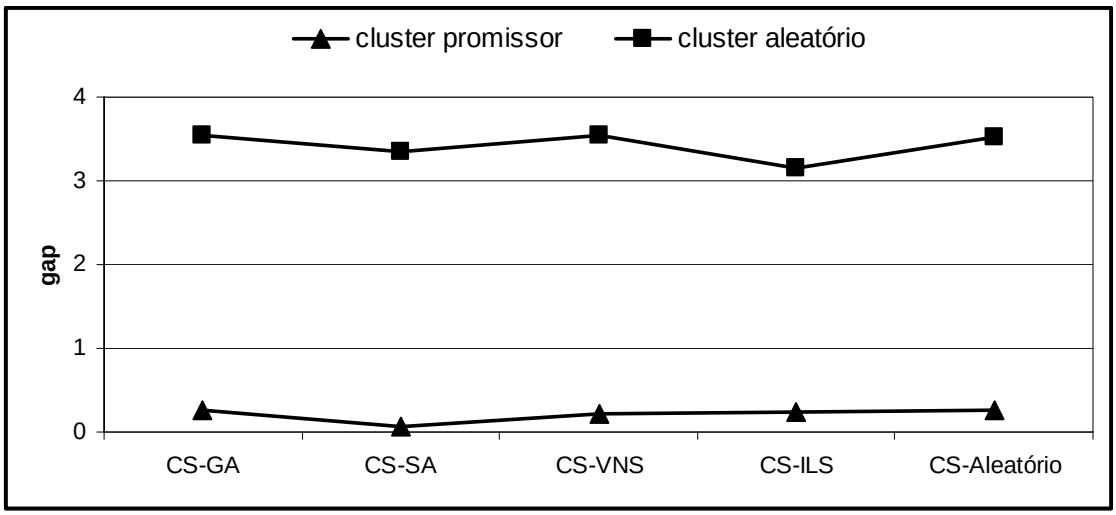

Figura 7.7 - Gráfico com os *gaps* dos CS's utilizando estratégias diferentes para aplicar busca local.

A Figura 7.8 apresenta o gráfico com os desvios do CS (erro relativo entre a melhor solução encontrada pelo CS e a média das soluções), no qual pode-se observar que, além do CS com detecção de regiões promissoras ter encontrado soluções melhores, este também foi mais robusto.

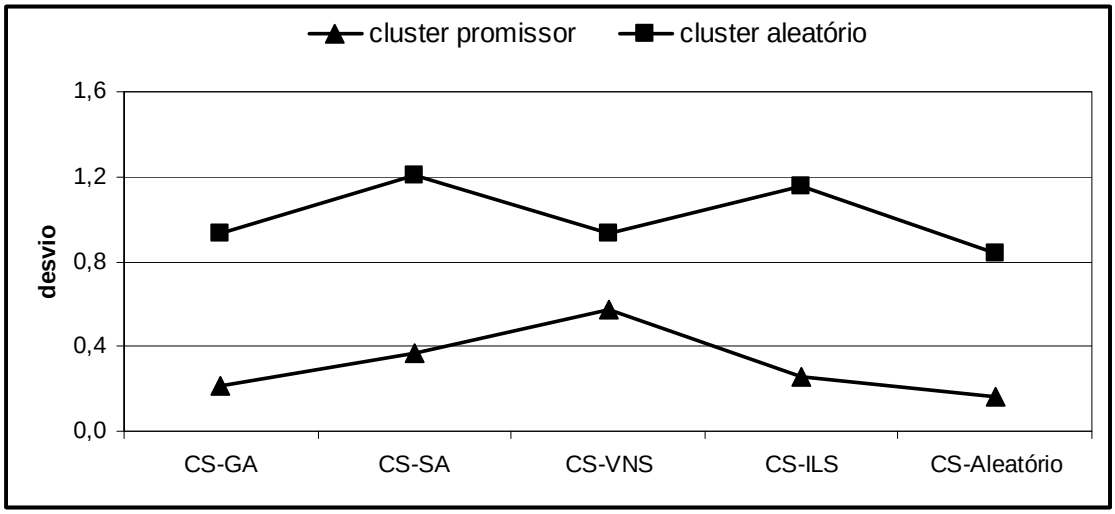

Figura 7.8 - Gráfico com os desvios dos CS's utilizando estratégias diferentes para aplicar busca local.

Esses resultados mostram que não adianta apenas aplicar busca local aleatoriamente, é preciso ter uma idéia sobre o espaço de solução para que o processo de busca consiga convergir para boas soluções.

7.5 Desempenho do CS sem Alguns Componentes

Para analisar a influência que cada componente do CS exerce sobre este, foram realizados dois testes nos quais retirou-se, respectivamente, o método Recombinação por Caminho (PR) e o componente de busca local.

Nestes testes utilizou-se a abordagem do CS aplicado ao CPMP, executando as instâncias *sjc*. Todos os demais componentes e parâmetros do CS foram iguais em ambos os testes.

No primeiro teste (MH+BL) é realizado o processo de assimilação simples ao invés do PR. Nessa assimilação, uma porcentagem das novas informações contida na solução agrupada é inserida aleatoriamente no centro do cluster. Neste teste, 20% das medianas diferentes entre a solução agrupada e o centro do cluster são introduzidas no centro, sendo os pontos de demanda realocados em seguida.

O segundo teste (MH+PR) foi executado com o CS sem o componente de busca local. Assim, as soluções são apenas geradas e agrupadas por meio do PR. E, quando um cluster se torna promissor nenhuma ação é realizada.

A Figura 7.9 apresenta o gráfico, em escala logarítmica, com os resultados dos *gaps* médios dos CS's neste teste. Novamente, os *gaps* foram calculados com base na melhor solução encontrada pelo CS e a solução ótima das instâncias. Pode observar-se que, o CS encontra resultados melhores do que as versões sem um dos componentes, e que a versão sem a busca local foi a que obteve os piores resultados. Este teste mostra a importância do componente de busca local, mas também a influência que o processo de assimilação tem sobre os resultados do CS. Portanto, a combinação de ambos é o que produziu os melhores resultados com relação à qualidade das soluções.

Em termos de robustez do método, o CS também obteve resultados melhores que as outras duas versões, como pode ser visto no gráfico, em escala logarítmica, da Figura 7.10.

Com relação ao tempo computacional, é evidente que houve diminuição em virtude da retirada de um dos componentes. O componente de busca local é responsável, em média, por 60% do tempo computacional e o processo de assimilação por caminho por 30% do tempo. Portanto, a retirada de um destes componentes acarreta uma diminuição proporcional ao tempo que ele representa.

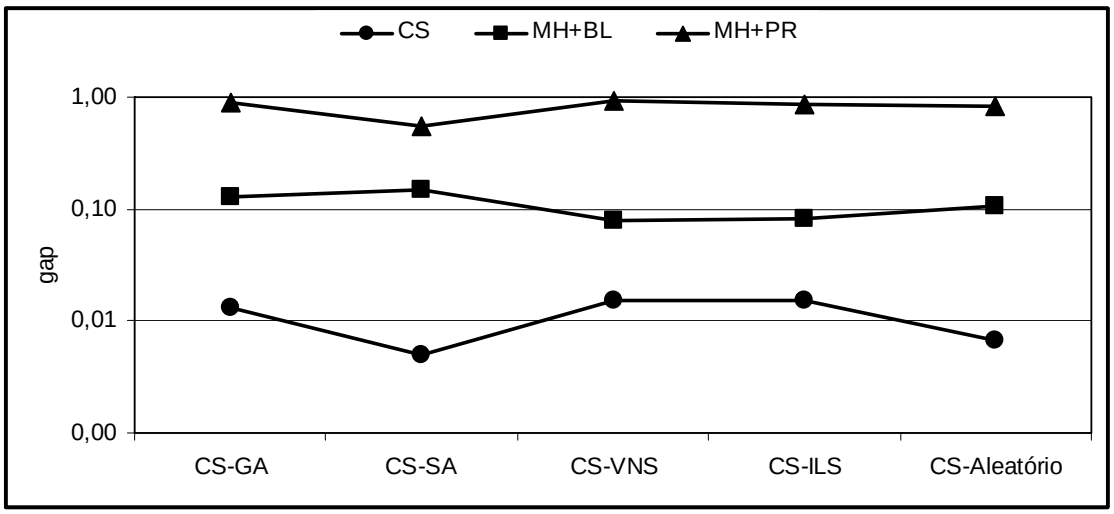

Figura 7.9 - Gráfico com os *gaps* dos CS's sem alguns componentes.

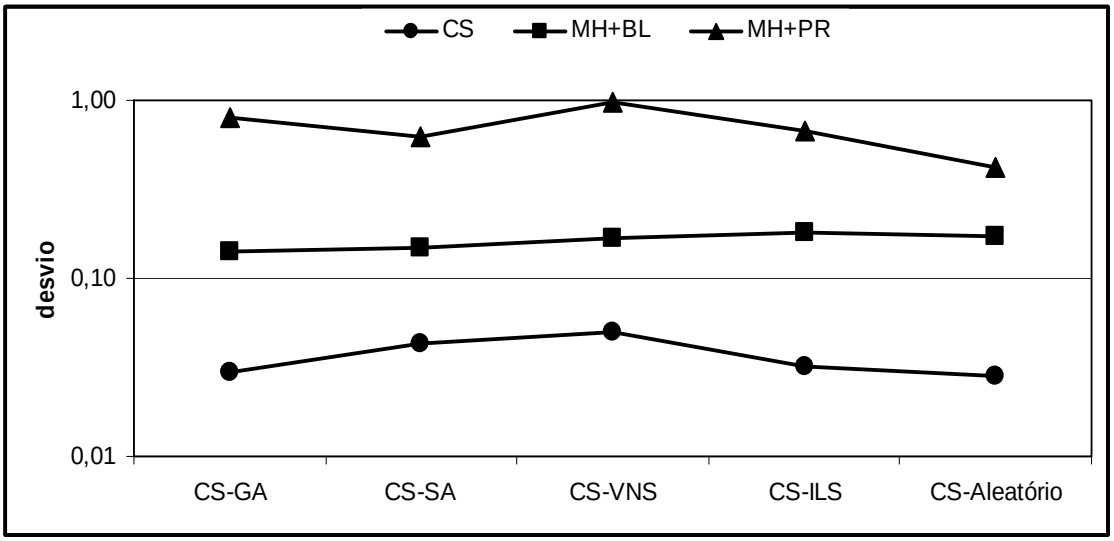

Figura 7.10 - Gráfico com os desvios dos CS's sem alguns componentes.

8 CONCLUSÕES E TRABALHOS FUTUROS

Nas últimas décadas, as meta-heurísticas têm-se apresentado como ferramentas interessantes para produzir, em tempo razoável, soluções de boa qualidade para problemas de otimização combinatória. Sendo que o conceito de meta-heurísticas híbridas vem ganhado cada vez mais importância, ao ponto que, uma escolha de uma abordagem híbrida adequada, muitas vezes utilizando conhecimento específico sobre a natureza do problema e informações sobre o espaço de busca, pode ser determinante para alcançar a melhor performance na resolução de problemas difíceis.

Todas as meta-heurísticas procuram balancear diversificação e intensificação. Isto é importante para evitar uma convergência prematura, mas também para identificar rapidamente regiões no espaço de busca com soluções de alta qualidade.

A intensificação é realizada muitas vezes por meio de heurísticas de busca local específicas para o problema abordado. Porém, essas heurísticas geralmente possuem um custo computacional muito alto, o que inviabiliza a sua utilização de forma indiscriminada.

Sendo assim, uma técnica interessante é a aplicação de busca local somente em regiões consideradas promissoras. Uma maneira de detectar essas regiões é por meio da quantidade de soluções geradas em cada região do espaço de busca. Portanto, estabelece-se que regiões nas quais muitas soluções são geradas tendem a ser regiões promissoras, isto é, regiões que contenham as melhores soluções.

Este trabalho discutiu e reorganizou um método híbrido que combina meta-heurísticas e heurísticas de busca local. Este método foi nomeado Busca por Agrupamento (CS, do inglês *Clustering Search*). A ideia é dividir o espaço de busca em subespaços e detectar regiões promissoras por mérito de frequência, ou seja, baseado na quantidade de soluções geradas em cada região. A busca é intensificada nas regiões promissoras tão logo estas sejam detectadas por meio das heurísticas.

O CS foi aplicado a diferentes problemas de otimização combinatória encontrados na literatura. A primeira abordagem resolve dois problemas de localização de facilidades: o Problema de p-Medianas Capacitado (CPMP, do inglês *Capacitated p-Median Problem*) e o Problema de Agrupamento Centrado Capacitado (CCCP, do inglês *Capacitated Centered Clustering Problem*). Na segunda abordagem foi resolvido o Problema do Caixeiro Viajante com Coleta de Prêmios (PCTSP, do inglês

Prize Collecting Traveling Salesman Problem). E por fim, foi tratado o Problema de Balanceamento e Designação de Trabalhadores em Linhas de Produção (ALWABP, do inglês *Assembly Line Worker Assignment and Balancing Problem*).

Sendo assim, as principais contribuições e as próximas etapas deste trabalho são apresentadas a seguir.

8.1 Resumo das Contribuições

As principais contribuições deste trabalho estão relacionadas às melhorias propostas para o método CS, tornando-o mais eficiente e amigável. Para tal, foram adicionados novos componentes e uma variável para controlar a eficiência da busca local.

Inicialmente foi proposto um método para criar os clusters iniciais baseado na máxima diversidade inter-clusters. O objetivo é dividir o espaço de soluções em regiões que representem diversas áreas de busca do problema. Assim, é possível identificar quais regiões são promissoras e intensificar a busca apenas nestas regiões.

Uma das metas deste trabalho era mostrar que o CS pode ser utilizado com qualquer meta-heurística, não somente com o Algoritmo Genético. Desta forma, foram implementadas quatro meta-heurísticas clássicas para gerar soluções para o processo de agrupamento. Sendo que, todas as versões do CS encontraram bons resultados. Outra modificação proposta é a utilização de um método gerador de soluções aleatórias. Apesar deste método não levar em consideração informações sobre o espaço de busca do problema, os resultados do CS com este método foram satisfatórios. A única questão levantada é com relação ao tempo de convergência do método, o qual foi maior nesta versão do CS.

Para permitir uma maior movimentação dos clusters pelo espaço de busca decidiu-se sempre atualizar o centro do cluster após o processo de assimilação. Desta forma, é possível pesquisar mais regiões no espaço de busca. Porém, essa modificação pode gerar um *loop* em torno de um ótimo local, fazendo com que o centro do cluster sempre retorne para a mesma solução. Para evitar esse problema, foi proposto adicionar uma variável para medir a eficiência do componente de busca local quando aplicado aos clusters. Esta variável foi chamada índice de ineficácia (r_i) e permite controlar se uma região é ruim e/ou já foi suficientemente explorada pelo componente de busca local do CS.

Ainda com relação ao componente de busca local, outra modificação proposta é a perturbação de um cluster no qual o índice de ineficácia for alto. Permitindo ao CS escapar de um possível ótimo local e regiões ruins, além de possibilitar a pesquisa em outras áreas do espaço de busca.

Este trabalho também contribuiu com a resolução, por meio do método CS, de diferentes problemas de otimização combinatória classificados na literatura como NP-*hard*. Todos os problemas abordados possuem um vasto campo de aplicações práticas e comerciais.

Os resultados do CS para o CPMP mostraram que o CS é um método eficiente. Sendo que, a solução ótima foi encontrada em 25 das 26 instâncias testadas, com tempos competitivos se comparados com os tempos de outros métodos encontrados na literatura. Na única instância que não foi possível obter a solução ótima, o erro relativo em relação a esta foi de 0,02%.

Utilizando o CS implementado para o CPMP foi possível resolver também o CCCP. Os resultados para esse problema foram muito bons, encontrando soluções com boa qualidade em um tempo computacional razoável. Sendo que, o CS encontra uma solução melhor que a conhecida na literatura até o momento em 20 das 25 instâncias testadas. A melhora na qualidade das soluções em relação aos resultados apresentados em Negreiros e Palhano (2006) foi de 17,26%.

No PCTSP os resultados também foram satisfatórios, apesar de não haver comparação com outros métodos da literatura. O CS conseguiu obter todas as soluções ótimas nas instâncias em que o CPLEX conseguiu provar a otimalidade, e para as instâncias nas quais o CPLEX não encontrou a solução ótima, as soluções do CS são bem melhores que as apresentadas pelo CPLEX. Além disso, o tempo computacional do CS para esse problema foi pequeno, principalmente por causa do fato das heurísticas utilizadas no componente de busca local (2-Opt, Inserir Vértice e Retirar Vértice) terem sido implementadas sem fazer chamadas à função objetivo do PCTSP.

Os resultados do CS para o ALWABP não foram tão satisfatórios. Mesmo assim, o CS com a meta-heurística Recozimento Simulado (SA) obteve bons resultados, encontrando a solução ótima para as instâncias menores e soluções melhores que as obtidas pelo CPLEX para as instâncias maiores. As demais versões do CS não

tiveram sucesso nas instâncias maiores. Além disso, o erro relativo entre a melhor solução encontrada pelo CS e a solução média ficou muito alto em todas as versões do CS com os diferentes geradores de soluções.

Por fim, outra contribuição deste trabalho é a realização de diversos testes que permitiram compreender melhor o funcionamento do CS e seu comportamento perante os vários fatores que podem influenciar no processo de busca.

Por meio destes testes pôde-se observar que o número de clusters influencia a eficiência do componente de busca local. Sendo que, quanto maior o número de clusters maior é a proporção de sucesso da busca local nos clusters.

Outro dado importante está relacionado ao tempo computacional do CS com diferentes valores de limitante λ, mostrando que o tempo computacional diminui a medida que aumenta-se o valor de λ. Porém, a qualidade das soluções fica pior com valores altos de λ. Portanto é necessário realizar um balanceamento entre tempo e qualidade na calibragem dos parâmetros do CS.

É possível notar também que não adianta aplicar busca local aleatoriamente, sendo necessário ter informações sobre o espaço de busca para que o CS convirja para boas soluções. Além disso, foi mostrada a importância do processo de assimilação e do componente de busca local para o CS.

As modificações propostas associadas a uma maior clareza em relação às funções de cada componente do CS fez deste um método de aprendizado e uso fáceis e intuitivos. Assim, acredita-se que este trabalho seja uma boa fonte de pesquisa para futuras implementações do CS aplicado a outros problemas de otimização.

No Apêndice A são listados os trabalhos oriundos desta pesquisa que foram publicados e/ou apresentados em eventos científicos nacionais e internacionais.

8.2 Trabalhos Futuros

O método híbrido apresentado neste trabalho abrange um campo de pesquisa que visa resolver problemas de otimização por meio da divisão do espaço de busca em regiões, realizando uma intensificação da busca somente em regiões consideradas promissoras. Embora os resultados obtidos tenham sido satisfatórios para os problemas de otimização combinatória testados, melhorias podem ser aplicadas.

O CS tem obtido sucesso quando aplicado a diversos problemas de otimização combinatória encontrados na literatura, tais como, problemas de sequenciamento de padrões (OLIVEIRA; LORENA, 2006), programação *Flow Shop* (FILHO et al., 2007a; FILHO et al., 2007b), escalonamento de jogos para torneios esportivos (BIAJOLI; LORENA, 2007), localização-alocação de máxima cobertura probabilístico (CORREA et al., 2007), localização de concentradores (*hubs*) (ALMEIDA; SENNE, 2008), entre outros. Espera-se que este trabalho possa incentivar ainda mais a aplicação do CS em outros problemas de otimização combinatória.

Na parte teórica, alguns testes ainda podem ser feitos no CS. Um possível alvo de estudos é a utilização de outras medidas de distância (por exemplo, distância de Levenshtein) para definir qual o cluster mais similar. Outros estudos também se faz necessário quanto à definição de um cluster se tornar promissor, e em relação a utilização de novas técnicas para criar os clusters iniciais.

Novas heurísticas de busca local podem ser implementadas, levando em consideração mais informações específicas sobre as características dos problemas abordados, para melhorar os resultados obtidos pelo CS. E, com o intuito de verificar a qualidade das soluções obtidas para o PCTSP e para o ALWABP, pode-se inserir as melhores soluções obtidas pelo CS no *software* CPLEX, como soluções iniciais para os modelos matemáticos apresentados neste trabalho. Assim, espera-se que o CPLEX consiga convergir para a solução ótima ou provar que a solução obtida pelo CS é ótima.

Outra melhora que pode ser aplicada ao método CS é a utilização de penalidades variáveis no cálculo da função objetivo dos problemas. O valor do peso que reflete a penalidade imposta se uma restrição for violada é aumentado enquanto o método não melhora a solução corrente. E, o valor do peso é diminuído quando o método encontrar uma solução melhor. Esta estratégia ajudaria o CS manter a diversidade nas soluções.

Uma alternativa interessante para que instâncias maiores sejam resolvidas em pouco tempo de processamento é a paralelização do CS. Esta pode ocorrer desde a paralelização simples do método gerador de soluções para o processo de agrupamento (ver Alba (2005) para maiores detalhes) até paralelizar os componentes do CS. Neste último caso, cada processador ficaria responsável por uma parte do processo de agrupamento. Além disso, os clusters podem ser divididos entre os processadores, sendo que, cada processador fica responsável pela intensificação da busca em um

cluster quando este for considerado promissor.

Em relação ao CS aplicado ao ALWABP, que não obteve resultados satisfatórios, uma melhoria possível é adotar novas estruturas de vizinhanças que não gerem tantas soluções inviáveis. Assim, acredita-se que o CS será mais robusto. Outra opção é estudar novas heurísticas de busca local mais eficientes.

Ainda sobre o ALWABP, seria interessante realizar uma implementação real deste problema. Tal objetivo conta com apoio do Professor Cristobal Miralles da Universidade Politécnica de Valência, que possui experiência em aplicações reais em centros de trabalho para deficientes situados na Espanha.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBA, E. **Parallel metaheuristics**: a new class of algorithms. New Jersey: Wiley-Interscience, 2005. 574 p. 179

ALMEIDA, W. G. de; SENNE, E. L. F. Uma aplicação da busca por agrupamentos ao problema de localização de concentradores não-capacitado. In: SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 40., 2008, João Pessoa. **Anais...** Rio de Janeiro: SOBRAPO, 2008. p. 1906–1915. 179

AUSIELLO, G.; BONIFACI, V.; LAURA, L. **The online prize-collecting traveling salesman problem**. Roma: Università di Roma La Sapienza, 2006. 15 p. 106

AWERBUCH, B.; AZAR, Y.; BLUM, A.; VEMPALA, S. New approximation guarantees for minimumweight k-trees and prize-collecting salesmen. **SIAM Journal on Computing**, v. 28, n. 1, p. 254–262, 1998. 102, 105

BALAS, E. The prize collecting traveling salesman problem. **Networks**, v. 19, p. 621–636, 1989. 31, 100, 101, 105, 106, 138

BALAS, E.; MARTIN, G. **ROLL-A-ROUND**: software package for scheduling the rounds of a rolling mill. Pittsburgh, 1985. 104 p. 105

BARTHOLDI, J.; EISENSTEEIN, D. A production line that balances itself. **Operations Research**, v. 44, n. 1, p. 21–34, 1996. 145

BAUM, E. B. **Iterated descent**: a better algorithm for local search in combinatorial optimization problems. Pasadena: Caltech, 1986. 41

BAXTER, J. Local optima avoidance in depot location. **Journal of the Operational Research Society**, v. 32, p. 815–819, 1981. 41

BECKER, R. W.; LAGO, G. A global optimization algorithm. In: ALLERTON CONFERENCE ON CIRCUITS AND SYSTEMS THEORY, 8., 1970, Illinois. **Proceedings...** [s.l.]: IEEE, 1970. p. 3–12. 45

BIAJOLI, F. L.; LORENA, L. A. N. Clustering search approach for the traveling tournament problem. In: GELBUKH, A.; MORALES, A. K. (Ed.). **Advances in Artificial Intelligence**. Berlin / Heidelberg: Springer, 2007, (Lecture Notes in Computer Science, v. 4827). p. 83–93. MICAI 2007. 179

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization overview and conceptual comparison. **ACM Computing Surveys**, v. 35, n. 3, p. 268–308, 2003. 33

_____. Hybrid metaheuristics: an introduction. In: BLUM, C.; AGUILERA, M. J. B.; ROLI, A.; SAMPELS, M. (Ed.). **Hybrid metaheuristics: studies in computational intelligence**. Berlin/Heidelberg: Springer, 2008. p. 1–30. 27, 29, 35, 43

BOCCIA, M.; SFORZA, A.; STERLE, C.; VASILYEV, I. A cut and branch approach for the capacitated p-median problem based on fenchel cutting planes. **Journal of Mathematical Modelling and Algorithms**, v. 7, n. 1, p. 43–58, 2008. 66, 76, 78, 84

BOYD, E. A. Generating fenchel cutting planes for knapsack polyhedra. **SIAM Journal on Optimization**, v. 3, n. 4, p. 734–750, 1993. 66

BRIMBERG, J.; HANSEN, P.; SALHI, N. M. S. A survey of solution methods for the continuous location-allocation problem. **International Journal of Operations Research**, v. 5, n. 1, p. 1–12, 2008. 87

BéRUBé, J.-F.; GENDREAU, M.; POTVIN, J.-Y. An exact ϵ-constraint method for bi-objective combinatorial optimization problems: application to the traveling salesman problem with profits. **European Journal of Operational Research**, v. 194, n. 1, p. 39–50, abril 2008. 107, 116

CESELLI, A. Two exact algorithms for the capacitated p-median problem. **4OR: A Quarterly Journal of Operations Research**, v. 1, n. 4, p. 319–340, 2003. 66

CHAVES, A. A.; BIAJOLI, F. L.; MINE, O. M.; SOUZA, M. J. F. Modelagens exata e heurística para resolução de uma generalização do problema do caixeiro viajante. In: SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 36., 2004, São João Del Rei. **Anais...** Rio de Janeiro: SOBRAPO, 2004. p. 1367–1378. 106

_____. Metaheurísticas híbridas para resolução do problema do caixeiro viajante com coleta de prêmios. **Produção**, v. 17, n. 2, p. 263–272, 2007. 106

CHAVES, A. A.; LORENA, L. A. N. Aplicação do algoritmo clustering search aos traveling salesman problems with profits. In: SIMPóSIO BRASILEIRO DE

PESQUISA OPERACIONAL, 39., 2007, Fortaleza. **Anais...** Rio de Janeiro: SOBRAPO, 2007. p. 1472–1483. 102

CHELOUAH, R.; SIARRY, P. Genetic and nelder-mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. **European Journal of Operational Research**, v. 148, n. 2, p. 335–348, 2003. 46

COROMINAS, A.; PASTOR., R.; PLANS, J. Línea de montaje con tiempos dependientes del tipo de operario. In: CONGRESO NACIONAL DE ESTADÍSTICA E INVESTIGACIóN OPERATIVA, 27., 2003. **Anais...** Lleida: Universitat de Lleida, 2003. p. 2789–2795. 145

CORREA, F. A.; CHAVES, A. A.; LORENA, L. A. N. Heurística híbrida com detecção de regiões promissoras aplicada ao problema probabilístico de localização-alocação de máxima cobertura. In: SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 39., 2007, Fortaleza. **Anais...** Rio de Janeiro: SOBRAPO, 2007. p. 1553–1565. 179

COSTA, A. M.; MIRALLES, C. Rotação de tarefas em linhas de produção com trabalhadores deficientes. In: SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 40., 2008, João Pessoa. **Anais...** Rio de Janeiro: SOBRAPO, 2008. p. 143–152. 146

CROES, G. A method for solving traveling salesman problems. **Operations Research**, v. 6, p. 791–812, 1958. 113

DANTZIG, G.; FULKERSON, R.; JOHNSON, S. Solution of a large-scale traveling salesman problem. **Operations Research**, v. 2, p. 393–410, 1954. 28, 99

DELL'AMICO, M.; MAFFIOLI, F.; SCIOMANCHEN, A. A lagrangian heuristic for the prize collecting traveling salesman problem. **Anals of Operations Research**, v. 81, n. 1, p. 289–306, 1998. 105, 116

DELL'AMICO, M.; MAFFIOLI, F.; VäRBRAND, P. On prize collecting tours and the asymetric traveling salesman problem. **International Transactions in Operational Research**, v. 2, n. 3, p. 297–308, 1995. 101, 105

DIAZ, J. A.; FERNANDEZ, E. Hybrid scatter search and path relinking for the capacitated p-median problem. **European Journal of Operational Research**, v. 169, n. 39, p. 570–585, 2006. 66

DOERR, K.; KLASTORIN, T. D.; MAGAZINE, M. J. Synchronous unpaced flow lines with worker differences and overtime cost. **Management Science**, v. 46, n. 3, p. 421–435, 2000. 145

EIBEN, A. E.; SCHIPPERS, C. A. On evolutionary exploration and exploitation. **Fundamenta Informaticae**, v. 35, n. 1-4, p. 35–50, 1998. 34

FEILLET, D.; DEJAX, P.; GENDREAU, M. Traveling salesman problems with profits. **Transportation Science**, v. 2, n. 39, p. 188–205, 2005. 31, 99, 106

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, p. 109–133, 1995. 109

FILHO, G. R.; NAGANO, M. S.; LORENA, L. A. N. Evolutionary clustering search for flowtime minimization in permutation flow shop. In: BARTZ-BEIELSTEIN, T.; AGUILERA, M. J. B.; BLUM, C.; NAUJOKS, B.; ROLI, A.; RUDOLPH, G.; SAMPELS, M. (Ed.). **Hybrid Metaheuristics**. Berlin / Heidelberg: Springer, 2007, (Lecture Notes in Computer Science, v. 4771). p. 69–81. 179

_____. Hybrid evolutionary algorithm for flowtime minimisation in no-wait flowshop scheduling. In: GELBUKH, A.; MORALES, A. K. (Ed.). **MICAI 2007: Advances in Artificial Intelligence**. Berlin / Heidelberg: Springer, 2007, (Lecture Notes in Computer Science, v. 4827). p. 1099–1109. 179

FISCHETTI, M.; TOTH, P. An additive approach for the optimal solution of the prize collecting traveling salesman problem. In: GOLDEN, B. L.; ASSAD, A. A. (Ed.). **Vehicle Routing: Methods and Studies**. North-Holland: Elsevier Science, 1988. p. 319–343. 105

FLESZAR, K.; HINDI, D. S. An effective vns for the capacitated p-median problem. **European Journal of Operational Research**, v. 191, n. 3, p. 612–622, 2008. 66, 78, 84

FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. **Artificial intelligence through simulated evolution**. New York: John Wiley, 1966. 162 p. 36

FORGY, E. W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. **Biometrics**, v. 21, n. 3, p. 768–769, 1965. 87

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability**: A guide to the theory of np-completeness. New York: W. H. Freeman & Co., 1990. 338 p. 29, 63

GEL, E. S.; HOPP, W. J.; OYEN, M. P. V. Factors affecting opportunity of worksharing as a dynamic line balancing mechanism. **IIE Transactions**, v. 34, n. 10, p. 847–863, 2002. 145

GENSCH, D. H. An industrial application of the traveling salesman subtour problem. **AIIE Transactions**, v. 10, n. 4, p. 362–370, 1978. 101

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers and Operations Research**, v. 5, p. 553–549, 1986. 33

_____. Tabu search and adaptive memory programing: advances, applications and challenges. In: BARR, R. S.; HELGASON., R. V.; KENNINGTON, J. L. (Ed.). **Interfaces in Computer Science and Operations Research**. Norwell: Kluwer, 1996. p. 1–75. 59

GLOVER, F.; LAGUNA, M. **Tabu Search**. Boston: Kluwer, 1997. 408 p. 34

GOLDBERG, D. E. **Genetic algorithms in Search**: optimization and machine learning. Addison-Wesley: Berkeley, 1989. 223 p. 36

GOLDEN, B. L.; LEVY, L.; VOHRA, R. The orienteering problem. **Naval Research Logistics**, v. 34, n. 3, p. 307–318, 1987. 101

GOMES, L. M.; DINIZ, V. B.; MARTINHON, C. A. An hybrid grasp+vnd metaheuristic for the prize collecting traveling salesman problem. In: SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 32., 2000, Viçosa. **Anais...** Rio de Janeiro: SOBRAPO, 2000. p. 1657–1665. 106, 113

GUTJAHR, A. L.; NEMHAUSER, G. L. An algorithm for the line balancing problem. **Management Science**, v. 11, n. 2, p. 308–315, 1964. 141

HAKIMI, S. Optimum location of switching centers and the absolute centers and the medians of a graph. **Operations Research**, v. 12, p. 450–459, 1964. 63

_____. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. **Operations Research**, v. 13, p. 462–475, 1965. 63

HAMMING, R. W. Error detecting and error correcting codes. **Bell System Technical Journal**, v. 26, n. 2, p. 147–160, 1950. 50

HANSEN, P.; MLADENOVIC, N. **A tutorial on variable neighborhood search**. Montreal: HEC Montreal and GERAD, 2003. 40, 62

_____. Variable neighborhood search. In: GLOVER, F.; KOCHENBERGER, G. A. (Ed.). **Handbook of Metaheuristics**. New York: Springer, 2003, (International Series in Operations Research & Management Science, v. 57). p. 145–184. 41

HENDERSON, D.; JACOBSON, S.; JOHNSON, A. The theory and practice of simulated annealing. In: GLOVER, F.; KOCHENBERGER, G. A. (Ed.). **Handbook of Metaheuristics**. New York: Springer, 2003, (International Series in Operations Research & Management Science, v. 57). p. 287–319. 38, 39

HOFFMANN, T. R. Assembly line balancing: a set of challenging problems. **International Journal of Production Research**, v. 28, p. 1807–1815, 1990. 153

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Michigan: University of Michigan Press, 1975. 211 p. 35, 36, 37, 70

HOPP, W. J.; TEKIN, E.; OYEN, M. P. V. **Benefits of skill chaining in production lines with cross trained workers**. WorkSmart Laboratory, 2001. 145

ILOG. **Ilog Cplex 10.0**. França, 2006. 478 p. 155

_____. **Ilog Cplex 11.0**. França, 2007. 140 p. 100, 116

JAIME, L. R.; CARMO, J. C. **A inserção da pessoa com deficiência no mundo do trabalho**: o resgate de um direito de cidadania. São Paulo: Ed. dos Autores, 2005. 204 p. 139

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM Computing Surveys**, v. 31, n. 3, p. 264–323, 1999. 44

JELASITY, M.; ORTIGOSA, P. M.; GARCÍA, I. Uego, an abstract clustering technique for multimodal global optimization. **Journal of Heuristics**, v. 7, n. 3, p. 215–233, 2001. 46

JOHNSON, D. S. Local optimization and the traveling salesman problem. In: INTERNATIONAL COLLOQUIUM ON AUTOMATA, LANGUAGES AND PROGRAMMING, 17., 1990. **Proceedings...** London: Springer-Verlag, 1990. p. 446–461. 41

JONG, K. A. D. **Analysis of the behavior of a class of genetic adaptive systems**. 271 p. Tese (Doutorado em Filosofia) — University of Michigan, 1975. 36, 38

JOZEFOWIEZ, N.; GLOVER, F.; LAGUNA, M. Multi-objective meta-heuristics for the traveling salesman problem with profits. **Journal of Mathematical Modelling and Algorithms**, v. 7, n. 2, p. 177–195, 2008. 107

KATAOKA, S.; MORITO, S. An algorithm for the single constraint maximum collection problem. **Journal of the Operations Research Society of Japan**, v. 31, n. 4, p. 515–530, 1988. 101

KELLER, C. P.; GOODCHILD, M. The multiobjective vending problem: a generalization of the traveling salesman problem. **Environment and Planning B: Planning and Design**, v. 15, p. 447–460, 1988. 101

KELLY, J. P. **Meta-Heuristics**: theory and applications. Norwell: Kluwer Academic Publishers, 1996. 704 p. 29

KIRKPATRICK, S.; GELLAT, D. C.; VECCHI, M. P. Optimization by simulated annealing. **Science**, v. 220, n. 4598, p. 671–680, 1983. 38

KUO, C.-C.; GLOVER, F.; DHIR, K. S. Analyzing and modeling the maximum diversity problem by zero-one programming. **Decision Sciences**, v. 24, n. 6, p. 1171–1185, 1993. 54

LAPORTE, G.; MARTELLO, S. The selective traveling salesman problem. **Discrete Applied Mathematics**, v. 26, p. 193–207, 1990. 101

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling salesman problem. **Operational Research**, v. 21, n. 2, p. 498–516, 1973. 33

LORENA, L.; SENNE, E. A column generation approach to capacitated p-median problems. **Computers and Operations Research**, v. 31, n. 6, p. 863–876, 2004. 66

LORENA, L. A. N.; SENNE, E. L. F. Local search heuristics for capacitated p-median problems. **Networks and Spatial Economics**, v. 3, n. 4, p. 407–419, 2003. 66, 71, 73, 74, 75, 76, 89

LOURENçO, H.; MARTIN, O.; STüTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. A. (Ed.). **Handbook of Metaheuristics**. New York: Springer, 2003, (International Series in Operations Research & Management Science, v. 57). p. 320–353. 41, 42, 43

MANSOOR, E. M. Assembly line balancing: a heuristic algorithm for variable operator performance levels. **Journal of Industrial Engineering**, v. 19, p. 618–628, 1968. 145

MARTIN, O.; OTTO, S. W.; FELTEN, E. W. Large-step markov chains for the traveling salesman problem. **Complex System**, v. 5, n. 3, p. 299–326, 1991. 41

MARTIN, O. C.; OTTO, S. W. **Combining simulated annealing with local search heuristics**. Oregon Graduate Institute School of Science & Engineering, 1993. 41

MARTINEZ-ESTUDILLO, A.; HERVAS-MARTINEZ, C.; MARTINEZ-ESTUDILLO, F.; GARCIA-PEDRAJAS, N. Hybridization of evolutionary algorithms and local search by means of a clustering method. **IEEE Transactions on Systems, Man, and Cybernetics, Part B**, v. 36, n. 3, p. 534–545, 2005. 46

MELO, V. A.; MARTINHON, C. A. Metaheurísticas híbridas para o problema do caixeiro viajante com coleta de prêmios. In: SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 36., 2004, São João Del Rei. **Anais...** Rio de Janeiro: SOBRAPO, 2004. p. 1295–1306. 106

MELO, V. V. de; DELBEM, A. C. B.; JúNIOR, D. L. P.; FEDERSON, F. M. Discovering promising regions to help global numerical optimization algorithms. In: GELBUKH, A.; MORALES, A. F. K. (Ed.). **Advances in Artificial Intelligence**. Berlin: Springer, 2007, (Lecture Notes in Computer Science, v. 4827). p. 72–82. MICAI 2007. 47

METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, E. Equation of state calculations by fast computing machines. **The Journal of Chemical Physics**, v. 21, p. 1087–1092, 1953. 38

MIRALLES, C.; GARCÍA-SABATER, J.; ANDRÉS, C.; CARDÓS, M. Advantages of assembly lines in sheltered work centres for disabled: a case study. **International Journal of Production Economics**, v. 110, n. 1-2, p. 187–197, 2007. 140, 141, 145, 153

_____. Branch and bound procedures for solving the assembly line worker assignment and balancing problem: application to sheltered work centres for disabled. **Discrete Applied Mathematics**, v. 156, n. 3, p. 352–367, 2008. 31, 141, 143, 145

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. **Computers and Operations Research**, v. 24, n. 11, p. 1097–1100, 1997. 40, 62

MULVEY, J. M.; BECK, M. P. Solving capacitated clustering problems. **European Journal of Operational Research**, v. 18, n. 3, p. 339–348, 1984. 63

NEGREIROS, M. J.; PALHANO, A. W. The capacitated centred clustering problem. **Computers and Operations Research**, v. 33, n. 6, p. 1639–1663, 2006. 31, 63, 85, 87, 89, 90, 91, 97, 177

NEMHAUSER, G. L.; WOLSEY, L. A. **Integer and combinatorial optimization**. New York: Wiley-Interscience, 1988. 763 p. 29

OLIVEIRA, A. C. M. **Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuo e discreto**. 200 p. Tese (Doutorado em Computação Aplicada) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2004. 30, 49, 58

OLIVEIRA, A. C. M.; LORENA, L. A. N. Detecting promising areas by evolutionary clustering search. In: BAZZAN, A. L. C.; LABIDI, S. (Ed.). **Advances in Artificial Intelligence - SBIA 2004**. Berlin / Heidelberg: Springer, 2004, (Lecture Notes in Artificial Intelligence). p. 385–394. 30, 46

_____. Pattern sequencing problems by clustering search. In: SICHMAN, J. S.; COELHO, H.; REZENDE, S. O. (Ed.). **Advances in Artificial Intelligence - IBERAMIA-SBIA 2006**. Berlin / Heidelberg: Springer, 2006, (Lecture Notes in Computer Science, v. 4140). p. 218–227. 179

OLIVER, I.; SMITH, D.; HOLLAND, J. A study of permutation crossover operators on the traveling salesman problem. In: INTERNATIONAL

CONFERENCE ON GENETIC ALGORITHMS, 2., 1987. **Proceedings...** Cambridge: Lawrence Erlbaum Associates, 1987. p. 224–230. 110, 149

OSMAN, I. H.; CHRISTOFIDES, N. Capacitated clustering problems by hybrid simulated annealing and tabu search. **International Transactions in Operational Research**, v. 1, n. 3, p. 317–336, 1994. 31, 65, 75

PALHANO, A. W. C.; NEGREIROS, M. J.; LAPORTE, G. A constrained k-median procedure for the capacitated centered clustering problem. In: CONGRESO LATINO IBERO AMERICANO DE INVESTIGACIóN DE OPERACIONES, 14., 2008, Cartagena de Indias, Colombia. **Proceedings...** [S.l.]: [s.n.], 2008. 87, 90, 97

PEREIRA, M. A.; SENNE, E. L. F. A column generation method for the capacitated centred clustering problem. In: ALIO/EURO WORKSHOP ON APPLIED COMBINATORIAL OPTIMIZATION, 6., 2008, Buenos Aires. **Proceedings...** Buenos Aires: Universidad de Bueno Aires, 2008. p. 1–6. 85, 87, 90, 97

RAIDL, G. R. A unified view on hybrid metaheuristics. In: HYBRID METAHEURISTICS, 3., 2006, Gran Canaria. **Proceedings...** Secaucus: Lecture Notes in Computer Science, Springer, 2006. p. 1–12. 35

RECHENBERG, I. **Evolutionsstrategie**: optimierung technischer systeme nach prinzipien der biologischen evolution. Stuttgart, 1973. 36

REEVES, C. Genetic algorithms. In: GLOVER, F.; KOCHENBERGER, G. A. (Ed.). **Handbook of Metaheuristics**. New York: Springer, 2003, (International Series in Operations Research & Management Science, v. 57). p. 55–82. 36

REINELT, G. Tsplib: a traveling salesman problem library. **ORSA Journal on Computing**, v. 3, n. 4, p. 376–384, 1991. 116

REKIEK, B.; DOLGUI, A.; DELCHAMBRE, A.; BRATCU, A. State of art of optimization methods for assembly line design. **Annual Reviews in Control**, v. 26, n. 2, p. 163–174, 2002. 145

RESENDE, M.; RIBEIRO, C. Grasp with path-relinking: recent advances and applications. In: IBARAKI, T.; NONOBE, K.; YAGIURA, M. (Ed.). **Metaheuristics: Progress as Real Problem Solvers**. New York: Springer, 2005, (Computer Science Interfaces Series). p. 29–63. 59

SCHEUERER, S.; WENDOLSKY, R. A scatter search heuristic for the capacitated clustering problem. **European Journal of Operational Research**, v. 169, n. 2, p. 533–547, 2006. 66, 78, 84

SCHOEN, F. Two-phase methods for global optimization. In: PARDALOS, P. M.; ROMEIJN, H. E. (Ed.). **Handbook of Global Optimization**. Norwell: Kluver Academic Publishers, 2002, (Nonconvex Optimization and Its Applications, v. 62). p. 151–177. 45

SCHOLL, A.; BECKER, C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. **European Journal of Operational Research**, v. 168, n. 3, p. 666–693, 2006. 141

SILVA, G. C. da; OCHI, L. S.; MARTINS, S. L. Proposta e avaliação de heurísticas grasp para o problema da diversidade máxima. **Pesquisa Operacional**, v. 26, n. 2, p. 321–360, 2006. 54, 55

SMITH, S. S.-F. Using multiple genetic operators to reduce premature convergence in genetic assembly planning. **Computers in Industry**, v. 54, n. 1, p. 35–49, 2004. 34

STüTZLE, T. **Iterated local search for the quadratic assignment problem**. TU Darmstadt, 1999. 41

STüZLE, T. **Local search algorithms for combinatorial problems**: analysis, improvements, and new applications. Tese (Doutorado em Computação Aplicada) — Darmstadt University of Technology, 1998. 43

SYSWERDA, G. Uniform crossover in genetic algorithms. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 3., 1989, Virginia. **Proceedings...** George Mason University: [s.n.], 1989. p. 2–9. 148

TAN, K.; CHIAMA, S.; MAMUNA, A.; GOHA, C. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. **European Journal of Operational Research**, 2008. Doi:10.1016/j.ejor.2008.07.025. 34

TANG, L.; WANG, X. An iterated local search heuristic for the capacitated prize-collecting travelling salesman problem. **Journal of the Operational Research Society**, v. 59, n. 5, p. 590–599, 2008. 106

TORRES, R. D.; BRITO, J. A. M. Problemas de coleta de prêmios seletiva. In: SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 35., 2003, Natal. **Anais...** Rio de Janeiro: SOBRAPO, 2003. p. 1359–1371. 104, 106, 138

TöRN, A. A. Global optimization as a combination of global and local search. In: COMPUTER SIMULATION VERSUS ANALYTICAL SOLUTIONS FOR BUSINESS AND ECONOMIC MODELS, 17., 1973, Göteborg. **Proceedings...** Göteborg: [s.n.], 1973. p. 191–206. 45

TöRN, A. A.; VIITANEN, S. Topographical global optimization using pre-sampled points. **Journal of Global Optimization**, v. 5, n. 3, p. 267–276, 1994. 45

TSILIGIRIDES, T. Heuristic methods applied to orienteering. **Journal of Operational Research Society**, v. 35, n. 9, p. 797–809, 1984. 101

WEI, L.; ZHAO, M. A niche hybrid genetic algorithm for global optimization of continuous multimodal functions. **Applied Mathematics and Computation**, v. 160, n. 3, p. 649–661, 2005. 47

WEISE, T. **Global optimization algorithms**: theory and application. Kassel: University of Kassel, 2008. 820 p. Disponível em: www.it-weise.de/projects/book.pdf. Acesso em: 20 dez. 2008. 28

WESOLOWSKY, G. The weber problem: history and perspectives. **Location Science**, v. 1, p. 5–23, 1993. 87

A APÊNDICE - PUBLICAÇÕES

Lista-se a seguir os trabalhos oriundos desta pesquisa que foram publicados e/ou apresentados em eventos científicos.

A.1 Trabalhos Publicados em Periódicos

Título:	*Clustering Search Algorithm for the Capacitated Centred Clustering Problem*
Autores:	Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena
Periódico:	*Computers & Operations Research*
Editores:	Christian Blum
Editora:	Elsevier
ISSN:	0305-0548
doi:	10.1016/j.cor.2008.09.011

Título:	*Hybrid Heuristics for the Probabilistic Maximal Covering Location-Allocation Problem*
Autores:	Francisco de A. Corrêa, Antonio A. Chaves e Luiz A. N. Lorena
Periódico:	*Operational Research: An International Journal*
Editor:	Constantin Zopounidis
Editora:	Springer
ISSN:	1109-2858
volume:	7
número:	3
páginas:	323-343
ano:	2007

A.2 Trabalhos Apresentados em Eventos Internacionais

Título:	*Hybrid Metaheuristic for the Prize Collectin Travelling Salesman Problem*
Autores:	Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena
Evento:	8^{th} European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP)

Local/Data: Napoli (Itália) / 26 a 28 de março de 2008
Periódico: *Lecture Notes in Computers Science*
Editores: J. van Hemert and C. Cotta
Editora: Springer
ISSN: 0302-9743
Volume: 4972
Páginas: 123-134
Ano: 2008

Título: *Clustering Search Heuristic for the Capacitated p-Median Problem*
Autores: Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena
Evento: 2^{nd} International Workshop on Hybrid Artificial Intelligence Systems (HAIS)
Local/Data: Salamanca (Espanha) / 12 e 13 de novembro de 2007
Periódico: *Advances in Soft Computing*
Editores: Emilio Corchado, Juan M. Corchado e Ajith Abraham
Editora: Springer
ISSN: 1615-3871
Volume: 44
Páginas: 136-143
Ano: 2008

Título: *Clustering Search Approach for the Assembly Line Worker Assignment and Balancing Problem*
Autores: Antonio A. Chaves, Cristobal Miralles e Luiz A. Nogueira Lorena
Evento: 37^{th} International Conference on Computers and Industrial Engineering (CIE)
Local/Data: Alexandria (Egito) / 20 a 23 de outubro de 2007
Editores: M. H. Elwany e A. B. Eltawil
Páginas: 1469-1478
Ano: 2007

Título: Heurística Híbrida com Busca por Agrupamentos aplicada aos *TSP's with Profits*
Autores: Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena

Evento: XII Escuela Latinoamericana de Verano de Investigación Operativa
Local/Data: Petrópolis (Brasil) / 02 a 09 de fevereiro de 2007

Título: *A Preprocessing Phase for the Evolutionary Clustering Search*
Autores: Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena
Evento: I Workshop on Computational Intelligence (WCI)
Local/Data: Ribeirão Preto (Brasil) / 23 a 27 de outubro de 2006
Ano: 2006

Título: *Hybrid Algorithms with Detection of Promising Areas for the Prize Collecting Traveling Salesman Problem*
Autores: Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena
Evento: 5^{th} International Conference on Hybrid Intelligent Systems (HIS)
Local/Data: Rio de Janeiro (Brasil) / 06 a 09 de novembro de 2005
Editora: IEEE Computer Society
Editores: Nadia Nedjah, Luiza Mourelle, Ajith Abraham e Mário Köopen
ISBN: 0-7695-2457-5
Páginas: 49-54
Ano: 2005

A.3 Trabalhos Apresentados em Eventos Nacionais

Título: Uma Metaheurística Híbrida Aplicada ao Problema de Balanceamento e Designação de Trabalhadores em Linha de Produção
Autores: Antonio A. Chaves, Luiz A. N. Lorena e Cristobal Miralles
Evento: XL Simpósio Brasileiro de Pesquisa Operacional (SBPO)
Local/Data: João Pessoa (Brasil) / 02 a 05 de setembro de 2008
Páginas: 1479 - 1490

Título: Aplicação do Algoritmo *Clustering Search* aos *Traveling Salesman Problems with Profits*
Autores: Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena
Evento: XXXIX Simpósio Brasileiro de Pesquisa Operacional (SBPO)

Local/Data: Fortaleza (Brasil) / 28 a 31 de agosto de 2007
Páginas: 1472 - 1483

Título: Heurística Híbrida com Detecção de Regiões Promissoras Aplicada ao Problema Probabilístico de Localização-Alocação de Máxima Cobertura
Autores: Francisco de A. Corrêa, Antonio A. Chaves e Luiz A. N. Lorena
Evento: XXXIX Simpósio Brasileiro de Pesquisa Operacional (SBPO)
Local/Data: Fortaleza (Brasil) / 28 a 31 de agosto de 2007
Páginas: 1553 - 1565

Título: Algoritmos Híbridos para uma generalização do Problema do Caixeiro Viajante
Autores: Antonio Augusto Chaves e Luiz Antonio Nogueira Lorena
Evento: VIII Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM)
Local/Data: Rio de Janeiro (Brasil) / 16 e 17 de agosto de 2005

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programas de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. Aceitam-se tanto programas fonte quanto os executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.