MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

A RELAXAÇÃO LAGRANGEANA/SURROGATE E ALGUMAS APLICAÇÕES EM OTIMIZAÇÃO COMBINATÓRIA

Marcelo Gonçalves Narciso

Tese de Doutorado em Computação Aplicada, orientada pelo Dr. Luiz Antônio Nogueira Lorena.

INPE
São José dos Campos
Março 1998

ABSTRACT

The lagrangean relaxation has been used there is a long time, with great success, as auxiliary in the development of methods for the search of feasible solutions to Combinatorial Optimization problems. The solutions of the relaxations supply limits, that jointly with other limits of feasible solutions to the problems, provide indications of the value of those feasible solutions.

Another well-known relaxation and maid, in this context, is the surrogate relaxation. This relaxation, although it supplies better limits in general than the lagrangeana, has not frequently been used due to the inherent difficulty of solution of the relaxed problem.

This work has the objective of show as local information can improve the performance of the employment of lagrangean relaxations when applied together with subgradient methods. A *surrogate* version of the lagrangean relaxation provides a local otimization that will contemplate in all the iterations of a subgradient method. This new form of use of local information can also be seen as a new relaxation, denominated in this work of relaxation *lagrangeana/surrogate* or simply *lagsur*. This new proposal was applied to the generalized assignment problem *(PGA)* and to the traveling salesman problem *(PCV*) and the obtained results were better than obtained with the lagrangean relaxation in terms of time of execution, mainly when the instances have great dimensions. Besides winning in time, the relaxation lagsur obtained such good limits as supplied for the lagrangean relaxation.

RESUMO

A relaxação lagrangeana tem sido empregada há muito tempo, com grande sucesso, como auxiliar no desenvolvimento de métodos para a busca de soluções ótimas aos problemas da Otimização Combinatória. As soluções das relaxações fornecem limites, que conjuntamente com outros limites de soluções viáveis aos problemas, proporcionam indicações do valor dessas soluções viáveis.

Outra relaxação conhecida e empregada, neste contexto, é a relaxação *surrogate*. Esta relaxação, embora forneça em geral limites melhores que a lagrangeana, não tem sido empregada frequentemente devido à dificuldade inerente de solução do problema relaxado.

Este trabalho tem como objetivo mostrar como informações locais podem melhorar a performance do emprego de relaxações lagrangeanas quando aplicadas em conjunto com métodos subgradientes. Uma versão *surrogate* da relaxação lagrangeana proporciona uma otimização local, que irá refletir em todas as iterações de um método subgradientes. Esta nova forma de uso de informações locais pode ser vista também como uma nova relaxação, denominada neste trabalho de relaxação *lagrangeana/surrogate* ou simplesmente *lagsur*. Esta nova proposta foi aplicada ao problema generalizado de atribuição *(PGA)* e ao problema do caixeiro viajante *(PCV*) e os resultados obtidos foram melhores do que os obtidos com a relaxação lagrangeana em termos de tempo de execução, principalmente quando as instâncias têm grandes dimensões. Além de ganhar em tempo, a relaxação lagsur obteve limites tão bons quantos os fornecidos pela relaxação lagrangeana.

AGRADECIMENTOS

Agradeço a Deus, primeiramente, por tornar possível esta tese de Doutorado ser realizada e por todas as outras coisa que Ele fez por mim.

Agradeço à minha esposa por ter sido uma grande companheira para mim.

Agradeço ao meu orientador, Luiz A. N. Lorena, por tudo o que fez por mim desde o Mestrado até aqui.

Agradeço a todo pessoal do CNPTIA/EMBRAPA e CNPA/EMBRAPA que me ajudou a confeccionar esta tese e também pelo apoio moral e financeiro.

Agradeço à CAPES pelo apoio financeiro.

Agradeço a todos os estudantes da Pós-Graduação do INPE pelo apoio que Me deram durante todo o tempo em que estive aqui.

SUMÁRIO

Pág

Pág

Pág

LISTA DE TABELAS

Tabela **Pág**

Tabela **Pág**

Tabela **Pág**

LISTA DE FIGURAS

CAPÍTULO 1

1 - Introdução

Considere problemas onde um grande número de possibilidades proporcionem soluções, mas que seriam inviáveis de avaliar, ou mesmo gerar, uma a uma. Os problemas de *Otimização Combinatória* caem neste contexto. Um espaço discreto e finito de soluções representa o conjunto viável para a avaliação de uma função objetivo. Este espaço possui em geral grande dimensão tornando os problemas de Otimização Combinatória difíceis de resolver. As diversas aplicações práticas de problemas que podem ser formulados neste contexto levaram a uma crescente pesquisa de métodos para sua solução. Este trabalho se enquadra na linha de propostas para a melhora de performance do emprego da *relaxação lagrangeana* neste contexto, particularmente nos problemas formulados como de *Programação Linear Inteira zero-um.*

A relaxação lagrangeana tem sido empregada há muito tempo, com grande sucesso, como auxiliar no desenvolvimento de métodos para a busca de soluções ótimas aos problemas da área. As soluções das relaxações fornecem limites, que conjuntamente com outros limites de soluções viáveis aos problemas, proporcionam indicações do valor dessas soluções viáveis. Podem participar de métodos enumerativos, ou no desenvolvimento de heurísticas. Em geral seu emprego está relacionado ao uso de um *método dual de otimização por subgradientes*, também de consagrado uso e conhecimento para pesquisadores de métodos de otimização.

Outras relaxações conhecidas e empregadas são a de *programação linear* e a relaxação *surrogate*. Esta última, embora forneça em geral limites melhores que as anteriores, não tem sido empregada, frequentemente devido à dificuldade inerente de solução do problema relaxado. As três relaxações são amplamente descritas na literatura e seus aspectos formais necessários aos resultados deste trabalho serão descritos no segundo capítulo.

Este trabalho tem como objetivo mostrar como informações locais podem melhorar a performance do emprego de relaxações lagrangeanas quando aplicadas em conjunto com métodos subgradientes. Uma versão *surrogate* da relaxação lagrangeana proporciona uma otimização local, que irá refletir em todas as iterações de um método subgradientes. Esta nova forma de uso de informações locais pode ser vista também como uma nova relaxação, denominada neste trabalho de relaxação *lagrangeana/surrogate* ou simplesmente *lagsur*. A nova relaxação será descrita no *capítulo 3*.

O comportamento da relaxação *lagsur* para se obter limites será investigado em sua aplicação a dois problemas clássicos e importantes devido ao seu grande número de aplicações: o *problema generalizado de atribuição (PGA)* e a versão simétrica do *problema do caixeiro viajante (PCV)*. Eles serão descritos nos *capítulos 4 e 5*, respectivamente. A relaxação *lagsur* será comparada com a lagrangeana usando instâncias de *"larga escala"*, isto é, problemas com um grande número de variáveis. Com o emprego da relaxação *lagsur* obtém-se um ganho significativo em relação à relaxação lagrangeana no que se refere a tempo computacional. Além disso, a relaxação *lagsur* fornece limites tão bons quanto os fornecidos pela relaxação lagrangeana.

Embora este trabalho enfoque o *lagsur* aplicado ao (*PGA)* e ao (*PCV)*, esta nova relaxação pode ser aplicada em vários outros problemas da Otimização Combinatória, em particular sua versão *surrogate contínua* foi aplicada a vários problemas de cobertura e localização de facilidades (veja Narciso [52], Lorena e Narciso [48], Lorena e Lopes [47], Lopes [46], Senne e Lorena [61], Espejo [20] e Almiñana e Pastor [1]).

Finalmente, no *capítulo 6*, será feita uma análise desta nova proposta de relaxação *(lagsur)* dentro do contexto do (*PCV)* e do (*PGA)*. Também serão sugeridas algumas linhas de pesquisa para continuação do tema. No *apêndice A* estão mostrados em tabelas todos os resultados obtidos com os problemas (*PGA)* e (*PCV)*.

CAPÍTULO 2

2 - Relaxação de Programação Linear, Lagrangeana e Surrogate

2.1 -Introdução

Para uma melhor compreensão do que seja *relaxação*, é oportuno definir antes alguns termos que serão usados daqui em diante neste trabalho. Após a definição destes termos (programação, problema de programação linear, problema de programação inteira), será dada uma breve introdução sobre relaxação e então serão enfocadas as relaxações de programação linear, lagrangeana e *surrogate*.

O termo *programação* é usado para descrever o planejamento de um conjunto de atividades. A idéia é representar a quantidade ou nível de cada atividade como uma variável de decisão. Assim, um problema pode ser representado como um conjunto de equações e inequações que, depois de resolvidos, geram o valor das variáveis. O termo *programação matemática* é usado para descrever a minimização ou maximização de uma função objetivo de muitas variáveis, sujeita a restrições sobre tais variáveis. Tanto no desenvolvimento como nas aplicações da programação matemática, surgem casos especiais nos quais todos os custos, requisitos e outras atividades de interesse são termos estritamente proporcionais aos níveis das atividades ou à soma de tais termos, isto é, a função objetivo é uma função linear, e as restrições são equações ou inequações lineares. Tal problema é chamado de *problema de programação linear (PPL).*

Os chamados *problemas de otimização discreta* são aqueles que algumas ou todas as variáveis de decisão são restritas a assumir valores dentro de um conjunto discreto. Alguns destes problemas podem ser formulados como problemas lineares com a restrição adicional de que algumas ou todas as variáveis de decisão assumam somente valores inteiros. Tais problemas são chamados de *problemas de programação inteira (PPI)*. Em geral, problemas de programação inteira são mais difíceis de se resolver do que os problemas de programação linear. Nesta tese serão enfocados, em capítulos posteriores,

dois problemas de programação linear inteira: O *problema generalizado de atribuição (PGA)* e a versão simétrica do *problema do caixeiro viajante (PCV)*.

Feitas as considerações acima, será enfocada então a *relaxação*. A relaxação de um problema tem como finalidade tornar o problema mais fácil de se resolver. Entretanto, a solução fornecida pela relaxação geralmente não é viável para o problema original. Uma solução é *viável* para um problema se satisfaz a todas as restrições do problema. Um limitante é obtido para o valor de sua solução ótima. Se o problema for de minimização (maximização), então o valor da função objetivo da relaxação fornecerá um limite inferior (superior) para a solução ótima do problema em questão. Estas afirmações serão mostradas mais adiante.

Neste capítulo serão cobertas as relaxações de programação linear, lagrangeana e *surrogate*. Estas relaxações são muito conhecidas e amplamente divulgadas na literatura. As relaxações lagrangeana e *surrogate* serão enfocadas com maior ênfase pois formam a base da relaxação *lagsur*, a qual será descrita no *capítulo 3*.

2.2 - Propriedades de uma relaxação

Seja o *(PPI)* formulado como

$$
\begin{array}{lll}
 & v(P) = & \min\ cx \\
(P) & \text{sujeito a} & Ax = b \\
 & & Dx \leq e \\
 & & x \geq 0 \text{ e inteiro;}
\end{array}
$$

onde x é um vetor *n x 1*, b é um vetor *m x 1*, e é um vetor *k x 1*, A é uma matriz *m x n* e D é uma matriz *k x n*.

De acordo com Parker e Rardin [55], define-se relaxação de um problema de minimização *(P)* como um problema *(RP)* tal que

(i) cada solução viável de *(P)* é também viável para *(RP)*;

(ii) a função objetivo do problema *(P)*, avaliada para qualquer ponto x_v viável, é maior ou igual a função objetivo de *(RP)* para o mesmo ponto x_v.

De acordo com a afirmação *(ii)*, podemos concluir que a relaxação fornece um limite inferior para problemas de minimização. Se *(P)* for problema de maximização, a afirmação *(i)* também é válida, porém a afirmação *(ii)* seria substituída por

(iii) a função objetivo do problema de maximização *(P)*, avaliada para qualquer ponto x_v viável, é menor ou igual a função objetivo de *(RP)* para o mesmo ponto x_v.

Desta forma, para problemas de maximização, o valor da função objetivo da relaxação seria um limitante superior ao valor ótimo da função objetivo do problema *(P)*.

Lema 2.1 - Seja *(P)* definido acima e *(RP)* uma relaxação de *(P)*. Então v*(RP)* ≤ v*(P)*.

Prova. Qualquer ponto x_v viável para *(P)* é também viável para *(RP)*. Assim, se *(P)* é não viável (v*(P)* = +∞) a desigualdade acima (v*(RP)* ≤ v*(P)*) se mantém. Se v*(P)* é ilimitado (v*(P)* = -∞), então v*(RP)* também é ilimitado. Se v*(P)* tem um ótimo e este ótimo é finito, o valor da função objetivo de *(P)*, o qual é maior ou igual a função objetivo de *(RP)*, fornece um limite superior a v*(RP)* pois a solução ótima de *(P)* é viável em *(RP)*.

Lema 2.2 - Seja *(P)* definido acima e *(RP)* uma relaxação de *(P)*. Seja c*.x a função objetivo de *(RP)*. Se x* é ótimo em *(RP)*, x* é viável em *(P)* e c.x* = c*.x* então x* é ótimo em *(P)*.

Prova. Um x* que satisfaça as hipóteses do Lema 2.2 é necessariamente viável em *(P)*. Isto implica dizer que c.x* ≥ v*(P)*. Então, do item (ii) do Lema 2.1, c*.x* = v*(RP)* ≤ v*(P)* ≤ c.x* = c*.x* Segue então que x* é solução viável para *(P)* e x* é solução ótima para *(P)*.

2.3 - A Relaxação de programação linear

Se o domínio de x, no problema *(P)*, for alterado de {0, 1, ...} (x é um número inteiro não negativo) para x ≥ 0 e x pertencente aos reais, então teremos uma relaxação de *(P)*. Esta relaxação é conhecida como *relaxação de programação linear* [55]. A grande vantagem desta relaxação de *(P)* é a existência de algoritmos eficientes para a solução de problemas de programação linear.

Lema 2.3 - Seja *(RPL)* a relaxação de programação linear de *(P)*. Temos as seguintes relações entre *(P)* e *(RPL)*,

(i) v*(P)* ≥ v*(RPL)*;

(ii) Se a solução x_{RPL} do problema *(RPL)* for tal que cada componente é não-negativa e inteira, então x_{RPL} também é solução de *(P)*.

A prova deste lema pode ser facilmente verificada usando-se os *lemas 2.1 e 2.2* (vide Parker e Rardin [55]).

Para problemas *(P)* nos quais x ∈ {0,1}, o item *(ii)* do *lema 2.3* seria melhor escrito da seguinte forma: se a solução x_{RPL} do problema *(RPL)* for tal que cada componente de x_{RP} seja 0 ou 1, então x_{RPL} também é solução de *(P)*.

2.4 - Relaxação lagrangeana

2.4.1 - Introdução

A *relaxação lagrangeana* é baseada na observação de que muitos problemas de programação inteira são modelados através de um conjunto de restrições que consideradas isoladamente tornam o problema de fácil solução. As outras possíveis restrições seriam restrições complicadoras tornando o problema de difícil solução. Para explorar esta observação, cria-se um problema lagrangeano no qual as restrições complicadoras são adicionadas à função objetivo através de um vetor de multiplicadores, e em seguida eliminadas do conjunto total de restrições.

O problema lagrangeano é frequentemente usado para fornecer limites em algoritmos de enumeração implícita do tipo "*branch and bound*", constituindo-se em uma alternativa ao uso da relaxação de programação linear (ver Fisher [21], Geoffrion[26]).

O "nascimento" da relaxação lagrangeana, tal como é conhecido hoje, ocorreu em 1970 quando Held e Karp [33,34] usaram a relaxação lagrangeana para fornecer um limite inferior para o problema do caixeiro viajante *(PCV)*. Motivados pelo sucesso da relaxação lagrangeana, este método foi aplicado ainda no início da década de 70 em problemas de programação inteira em geral (Shapiro [62]).

A lista de aplicações da relaxação lagrangeana é muito extensa, assim como a amplitude do tipo de problemas que são abordados usando esta técnica. Uma relação de algumas das aplicações pode ser encontrada em Fisher [21], e incluem problemas tais como o do caixeiro viajante, problemas de localização, problema generalizado de atribuição, problemas de cobertura e particionamento de conjuntos. Outros exemplos de aplicações estão amplamente descritos na literatura. Neste trabalho veremos a aplicação desta relaxação ao problema generalizado de atribuição e no problema simétrico do caixeiro viajante.

2.4.2 - Formulação da relaxação lagrangeana

Vamos assumir as restrições de *(P)* particionadas em dois conjuntos: Ax = b e Dx≤e. Suponha que Dx≤e sejam as restrições complicadoras. Para facilitar a resolução de *(P)*, vamos adicionar a restrição Dx≤e, modificada por um vetor u≥0, à função objetivo, isto é,

$$
\begin{array}{lll}
 & v(L_u) = min & \{cx - u(e - Dx)\} \\
(L_u) & \text{sujeito a} & Ax = b \\
 & & x \geq 0 \text{ e inteiro,}
\end{array}
$$

onde $u = (u_1, u_2, ..., u_k)$ é um vetor de multiplicadores com componentes não-negativos. (L_u) é uma relaxação lagrangeana de *(P)*.

Por conveniência, vamos assumir que (L_u) tenha solução viável e que o conjunto J = { x | Ax = b, x ≥ 0 e inteiro } de soluções viáveis para (L_u) seja finito. Então (L_u) é finito para todo u ≥ 0. É bem conhecido que $v(L_u) \leq v(P)$ (Fisher[21]). Isto é fácil mostrar. Assumindo x* como solução ótima para *(P)*, observa-se que $v(L_u) \leq cx^* - u(e - Dx^*) \leq v(P)$. A primeira desigualdade na relação acima segue da definição de $v(L_u)$. A Segunda desigualdade segue do fato que $cx^* - u(e - Dx^*) \leq cx^* = v(P)$, visto que $u \geq 0$.

Em geral, não é possível garantir um determinado valor de u para o qual $v(L_u) = v(P)$. O fato de que $v(L_u) \leq v(P)$ permite que $v(L_u)$ seja usado em substituição ao valor da relaxação de programação linear, fornecendo limites inferiores para *(P)* em algoritmos exatos tipo "*branch and bound*".

2.4.3 - Exemplo de formulação da relaxação lagrangeana para o (PGA)

Para exemplificar o uso da relaxação lagrangeana, vamos considerar o problema generalizado de atribuição *(PGA).* Este problema pode ser visto como maximizar o lucro p_{ij} (benefício, rendimento, etc.) de se atribuir n itens a m mochilas (n > m) tal que cada item seja atribuído a apenas uma mochila, levando-se em conta as restrições de peso w_{ij} de cada item e a capacidade b_i de cada mochila. A formulação matemática do *(PGA)*, para o caso de maximização, é a seguinte:

$$v(PGA) = max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}.x_{ij}$$

(PGA)

$$\text{sujeito a} \quad \sum_{j=1}^{n} w_{ij}.x_{ij} \leq b_i, \ i \in M = \{1,...,m\} \qquad (2.1)$$

$$\sum_{i=1}^{m} x_{ij} = 1, \ j \in N = \{1,...,n\} \qquad (2.2)$$

$$x_{ij} \in \{0,1\}, \ i \in M, \ j \in N \qquad (2.3)$$

As restrições (2.1) impõem que os pesos dos itens escolhidos não devem exceder à capacidade de cada mochila e as restrições (2.2) impõem que cada item deve ser atribuído a somente uma mochila.

Duas relaxações lagrangeanas podem ser diretamente identificadas, relaxando-se as restrições de capacidades (2.1) ou relaxando-se as restrições de atribuição (2.2). Vamos considerar a primeira relaxação.

Seja λ um vetor de multiplicadores $\lambda_i \geq 0$ e $i \in M$. A relaxação lagrangeana será

$$v(L_{\lambda}PGA) = max \sum_{i=1}^{m} \sum_{j=1}^{n} \{p_{ij}.x_{ij} + \lambda_i.(b_i - w_{ij}.x_{ij})\}.$$

($L_{\lambda}PGA$)

$$\text{sujeito a} \quad (2.2) \text{ e } (2.3).$$

2.4.4 - Dual lagrangeano

Seja o problema (L_u) tal como definido anteriormente no item 2.4.2. O *dual lagrangeano* de (L_u), chamado doravante de (DL_u), determina para um conjunto de vetores u diferentes, o maior valor de $v(L_u)$. O dual de (L_u) seria portanto

$$v(DL_u) = max\ v(L_u),\ u \geq 0,$$

onde $u = (u_1, u_2, ..., u_k)$ é um vetor de multiplicadores com componentes reais não negativos. Vale a pena ressaltar que se as restrições relaxadas do problema original *(P)* forem igualdades, então u pode ser irrestrito de sinal.

A resolução do dual pode ser obtida aplicando um método de otimização de subgradientes. Este método será descrito mais adiante.

A seguir, vamos apresentar alguns resultados importantes envolvendo o dual lagrangeano, a relaxação lagrangeana, o problema primal *(P)* e a relaxação de programação linear. Através destes teoremas, podemos verificar características importantes da relaxação lagrangeana . O material apresentado aqui baseia-se no livro de Parker and Rardin [55], onde podem ser encontrados mais detalhes, incluindo provas de teoremas e provas de convergência de algoritmos.

Teorema 2.1. (Dualidade lagrangeana fraca) Sejam *(P)*, (L_u) e (DL_u) definidos anteriormente. Então, para todo $u \geq 0$, $v(L_u) \leq v(P)$. Consequentemente, $v(DL_u) \leq v(P)$.

Teorema 2.2. (Dualidade lagrangeana forte) Sejam *(P)*, (L_u) e (DL_u) definidos anteriormente. Se x* resolve (L_{u^*}) para algum $u^* \geq 0$ e também $Dx^* \leq e$ e $u^*.(e - Dx^*) = 0$ então x* resolve *(P)*.

Teorema 2.3. (Caracterização de $v(DL_u)$) Seja J o conjunto $\{ x \mid x \geq 0$ e inteiro e $Ax=b\}$. Sejam *(P)*, (L_u) e (DL_u) definidos anteriormente. Então

$$v(DL_u) = \min \; cx \quad Ax = b \quad Dx \leq e \quad x \in [\mathbf{J}]$$

onde [J] denota a envoltória convexa de pontos em J.

Corolário 2.1. (Dual lagrangeano versus relaxação de programação linear) Sejam *(P)*, (L_u) e (DL_u) definidos anteriormente e *(RPL)* a relaxação de programação linear de *(P)*. Então $v(DL_u) \geq v(RPL)$.

Quando [J] = J*, isto é, quando subproblemas (L_u) puderem ser resolvidos pela relaxação de programação linear, diz-se que J tem a *propriedade de integralidade*. Um resultado importante pode ser visto no corolário a seguir.

Corolário 2.2. (Propriedade de integralidade) Sejam *(P)*, (L_u) e (DL_u) definidos anteriormente. Se J tem a *propriedade de integralidade*, então $v(DL_u) = v(RPL)$.

Teorema 2.4. (Concavidade linear por partes da função dual lagrangeana) Sejam *(P)*, (L_u) e (DL_u) definidos anteriormente. Então $v(L_u)$ é uma função côncava linear por partes em u.

2.4.4.1 – Resolução do dual lagrangeano

Um dos métodos de solução de problemas não-lineares é a busca por gradientes. A idéia deste método é, partindo-se de um determinado ponto, avançar ao longo do gradiente da função até um ponto em que não se consiga mais um progresso significativo. Calcula-se, então, o gradiente neste ponto e a busca continua. Se o problema a ser resolvido é de maximização e a função a ser maximizada é côncava, tal processo permite uma aproximação do ótimo. Do *teorema 2.4*, tem-se que $v(L_u)$ em função de u é côncava. A

dificuldade aqui é que, como esta função é linear por partes, não é possível garantir que ela seja sempre diferenciável.

Para tratar a falta de diferenciabilidade é necessário generalizar o conceito de gradiente. Um vetor s é um *subgradiente* de uma função (côncava) $\theta(u)$ no ponto u^* se, para todo u, $\theta(u^*)+s(u-u^*) \geq \theta(u)$.

Teorema 2.5. Seja (L_u) como definido acima, com J não vazio e finito. Seja $S = \{x \in J \mid x \text{ resolve } (L_u)\}$. Então a coleção de subgradientes de $v(L_u)$ para qualquer u tem a forma:

$$\{\sum_{x \in S} \beta_x(e - Dx) \mid \sum_{x \in S} \beta_x = 1,\ \beta_x \geq 0,\ \text{para todo } x \in S\}.$$

O teorema a seguir estabelece que tais subgradientes aproximam (no sentido euclidiano de distância) as soluções de um problema à solução ótima.

Teorema 2.6. Seja $\theta(u)$ uma função côncava e contínua de u. Seja, para um dado α, o conjunto $B(\alpha) = \{ u \mid \theta(u) \geq \alpha \}$. Então, dados α, $u^* \notin B(\alpha)$ e s (subgradiente de θ no ponto u^*), existe $\delta \in \Re$ tal que:

$| u - (u^* + \beta s) | < | u - u^*|$, para todo $u \in B(\alpha)$ e $\beta \in (0, \delta)$.

Com base nos resultados acima, pode-se estabelecer o seguinte algoritmo para a solução de (DL_u):

Algoritmo de subgradientes

Seja $u \geq 0$,

Fazer $k \leftarrow 1$, $v \leftarrow -\infty$,

Enquanto {não para as condições de parada} ***fazer***

Resolver a relaxação lagrangeana (L_u). Seja $x \in J$ a solução ótima obtida.

Se $v < v(L_u) = c\,x + u\,(e - D\,x)$ ***então*** $v \leftarrow v(L_u)$.

Fazer $u \leftarrow u + p_k(e - Dx)$, onde p_k é o tamanho de passo,

Fazer $u_j \leftarrow \max(0, u_j)$ para todo j,

Testar condições de parada,

Fazer $k \leftarrow k+1$,

fim_enquanto.

Deve-se observar neste algoritmo que é necessário manter o valor v da melhor solução obtida anteriormente porque os passos do método subgradientes não garantem melhorias em v(L_u). As condições de parada poderiam ser $(e - Dx) \geq 0$ e $u.(e - Dx) = 0$, onde neste caso $v(L_u) = v(DL_u) = v(P)$, *ou* o tamanho do passo ficou muito pequeno *ou* um número pré determinado de iterações foi atingido. Outra observação refere-se aos tamanhos dos passos p_k, que devem satisfazer as condições:

(i) $p_k > 0$ para todo k, e
$\lim_{k \to \infty} p_k . \|(e - Dx)\| = 0;$

(ii) $\sum_{k=1}^{\infty} p_k . \|(e - Dx)\| = \infty.$

Esta condições foram propostas por Ermol'ev [19], para garantir a convergência do algoritmo. Poljak [56] propôs que o cálculo do passo P_k fosse feito da seguinte maneira

$$p_k \leftarrow q_k.(v(DL_u) - v(L_u))/\|(e - D x)\|^2,$$

onde $0<\gamma< q_k \leq 2 - \gamma, \ \gamma \in \Re$.

Held, Wolfe e Crowder [35] validaram outros tamanhos de passos sugeridos por Held e Karp [34] . Na expressão de p_k proposta por Poljak [56] , o valor de v(DL_u) é substituído pelo melhor limite superior conhecido (melhor solução viável para *(P)* gerada ou conhecida no momento da atualização do passo) e o intervalo de variação de q_k substituído por $0 \leq q_k \leq 2$, começando com q_k inicial igual a 2. Esta proposta, embora largamente usada, não satisfaz a condição (ii) de Ermol'ev [19] . Nos problemas enfocados nesta tese (*PGA* e *PCV*), o passo usado foi o proposto por Poljak [56] com a substituição de v(DL_u) pelo melhor limite superior conhecido, conforme a proposta de Held e Karp [33,34], bem como a variação do valor de q_k ($0 \leq q_k \leq 2$). Isto foi feito devido ao fato de que, para a maioria das instâncias utilizadas, não se conheciam os valores das soluções ótimas. A determinação do tamanho do passo e provas de convergência do método são tópicos que têm recebido considerável atenção, o que pode ser consultado em Allen et al. [2], Held et al.[35], Bazaraa e Sherali [6] e Goffin [30].

Outros métodos para resolver (DL_u) estão baseados em variantes do método simplex, e algoritmos especializados em métodos de ajuste de multiplicadores, que aproveitam a estrutura de uma aplicação particular (ver Fisher [23]). Estes métodos estão fora do escopo deste trabalho e não serão mostrados.

2.5 - Relaxação surrogate

2.5.1 - Introdução

A relaxação *surrogate* foi introduzida na programação matemática por Glover [27,28]. Basicamente, esta relaxação consiste em tomar um conjunto de restrições do problema original *(P)* e usando um vetor de multiplicadores transformar estas restrições em apenas uma restrição. Esta nova restrição é chamada de restrição *surrogate*.

Desde a sua introdução por Glover [27,28], a relaxação *surrogate* tem sido proposta por vários autores para o uso na solução de problemas não convexos, especialmente em problemas de programação inteira. Um tratamento teórico abrangente da dualidade *surrogate* em programação matemática é dado por Greenberg e Pierskalla [31]. Glover [29], em 1975, resumiu esses resultados e apresentou um tratamento unificado da teoria da dualidade *surrogate*. Um estudo da relação entre o dual lagrangeano e o dual *surrogate*, para o caso de problemas inteiros, foi feito por Karwan e Rardin [40].

Procedimentos de busca para multiplicadores *surrogate* foram desenvolvidos por Banerjee[5], Glover [27,28,29], Karwan e Rardin [40] , Dyer [18] , Sarin et al.[60].

Mais recentemente, Lorena, Freville e Plateau [49] aplicaram a relaxação *surrogate contínua* (relaxa-se também a restrição de integralidade) ao problema multidimensional da mochila 0-1 com vantagens sobre a relaxação lagrangeana, principalmente em relação à estabilidade na convergência da sequência dos valores relaxados. No trabalho de Lorena e

Lopes [47] a relaxação *surrogate contínua* foi aplicada com sucesso no problema de cobertura de conjuntos *(PCC)*, obtendo maior estabilidade na sequência de valores relaxados e também resultados melhores em relação a tempo computacional em comparação com a relaxação lagrangeana. Em Lorena e Narciso [48], esses resultados foram confirmados com o uso da relaxação *surrogate contínua* no problema generalizado de atribuição *(PGA)*. Os trabalhos de Lorena e Lopes [47] e Lorena e Narciso [48] serão mostrados, resumidamente, neste capítulo para exemplificar a aplicação da relaxação *surrogate*.

2.5.2 - Formulação da relaxação surrogate

Seja o problema *(P)* definido na *seção 2.2*. Considera-se um vetor de multiplicadores $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k)$, onde $\lambda_i \geq 0$, i = 1,..,k. Uma relaxação *surrogate* referente a *(P)* pode ser formulada como

$$
\begin{array}{lll}
 & v(S_l) = \quad min & cx \\
(S_l) & \text{sujeito a} & Ax = b \\
 & & \lambda Dx - \lambda e \leq 0, \\
 & & x \geq 0 \text{ e inteiro.}
\end{array}
$$

Desta forma, as k restrições de $Dx \leq e$ são substituídas pela restrição $\lambda Dx - \lambda e \leq 0$. Quando as restrições de integralidade são também relaxadas, obtem-se a *relaxação surrogate contínua (Scont$_l$)*, que no caso de *(S$_l$)* produz um problema de programação linear.

De um modo geral os problemas resultantes das relaxação *surrogate* são mais difíceis de resolver que os da relaxação lagrangeana. O *dual surrogate (DS$_l$)* é definido como *max* $\{v(S_l)\}$, $\lambda \geq 0$. Sua solução fornece limites melhores ou iguais ao dual lagrangeano. Este fato foi demonstrado por Greenberg e Pierskala [31] e Glover [27], e será formalizado a seguir. Novamente aqui, as provas dos teoremas que não forem incluídas podem ser encontradas no livro de Parker e Rardin [55].

Teorema 2.7. (Dualidade surrogate fraca) Sejam *(P)*, (S_l) e (DS_l) definidos anteriormente. Então, para todo $\lambda \geq 0$, $v(S_l) \leq v(P)$. Consequentemente, $v(DS_l) \leq v(P)$.

Prova. Cada (S_l) é obviamente uma relaxação de *(P)*, conforme o *lema 2.1*, pois cada solução viável para *(P)* é viável para (S_l). Segue então que

$v(DS_l) = max_{\lambda \geq 0}\ v(S_l) \leq v(P)$.

Teorema 2.8. (Dualidade surrogate forte) Sejam *(P)*, (S_l) e (DS_l) definidos anteriormente. Então, se para todo $\lambda^* \geq 0$, existe um x^* que resolve (S_{l^*}) e satisfaz $Dx^* \leq e$, x^* resolve *(P)* e $v(DS_l) = v(P)$.

Prova. Como (S_l) e *(P)* têm a mesma função objetivo, e (S_l) é uma relaxação de *(P)*, segue do *lema 2.2* que x^* resolve *(P)*. Além disso, a dualidade fraca implica $cx^* = v(S_l) \leq v(DS_l) \leq v(P) = cx^*$. Desta forma, $v(DS_l) = v(P)$.

Teorema 2.9. (Dual surrogate versus dual lagrangeano) Seja o problema *(P)* definido sobre um conjunto não vazio T. Sejam *(P)*, (S_l) e (DS_l) definidos anteriormente. Então $v(DL_u) \leq v(DS_l)$. Além disso, se $v(DL_u) = v(DS_l)$, então, para qualquer u^* que resolva (DL_u), deve existir $x^* \in T$ tal que $u^*(e - Dx^*) = 0$.

2.5.3 - Aplicação da relaxação surrogate contínua

Para ilustrar a aplicação da relaxação *surrogate contínua*, vamos tomar como exemplos os problemas de cobertura de conjuntos *(PCC)* e o generalizado de atribuição *(PGA)*. O trabalho de Lorena e Lopes [47] sobre o *(PCC)* e o de Lorena e Narciso [48] sobre o *(PGA)*, mostraram como a relaxação *surrogate contínua* pode ser usada com vantagens sobre a relaxação lagrangeana. Inicialmente, vamos descrever o *(PCC)* e, em seguida, o *(PGA)*, enfocando a relaxação *surrogate*.

2.5.3.1 - Aplicação da relaxação surrogate para o (PCC)

Muitos problemas da vida real podem ser modelados como *(PCC)*. Como exemplos, podemos citar: recuperação de informações (Day [15]), localização de facilidades (Garfinkel [25]), balanceamento de linhas de montagem (Salverson [59]), projeto de circuitos digitais (Quine[57]) e roteamento (Foster [24]). Outras áreas de aplicação podem ser encontradas em Balas e Padberg [4].

Lopes [46] desenvolveu um algoritmo heurístico para o *(PCC)* baseado em uma relaxação *surrogate contínua*, usada em conjunto com um método de subgradientes, e comparou os seus resultados com a heurística lagrangeana de Beasley [7] para o *(PCC)*. Veremos posteriormente que a aplicação do método subgradientes com o *surrogate contínuo* foi realizada em uma relaxação lagrangeana correspondente, denominada relaxação *lagrangeana/surrogate*, que será melhor descrita no *capítulo 3*.

A heurística de Beasley consiste em resolver o dual lagrangeano para fornecer um limite inferior à solução ótima. Além disso, a heurística de Beasley fornece um limite superior, dado por uma heurística, a qual gera a solução viável para o *(PCC)* a partir da solução não viável obtida pela relaxação.

Para descrever a heurística desenvolvida por Lopes para o *(PCC)*, é conveniente definir a relaxação surrogate para o *(PCC)*.

Seja o *(PCC)* abaixo

$$
\begin{array}{lllll}
 & v(PCC) = & \text{Min} & cx & \\
(PCC) & \text{sujeito a} & & Ax \geq e & \\
 & & & x \in \{0,1\}^n. &
\end{array}
$$

onde **c** é um vetor de custos com *n* componentes, **A** é uma matriz de elementos iguais a 0 ou 1 e cuja dimensão é *m x n*, **e** é um vetor *n x 1* e com todos os elementos iguais a 1 e x é um vetor com *n* componentes.

A relaxação *surrogate* associada ao *(PCC)* é

$$
\begin{array}{lll}
 & v(S_l) = \ min & cx \\
(S_l) & \text{sujeito a} & \lambda Ax \geq \lambda e \\
 & & x \in \{0,1\}^n.
\end{array}
$$

onde $\lambda_i \geq 0$ e o vetor λ é não nulo.

Lopes [46], em seu trabalho, considerou a relaxação *surrogate contínua* do *(PCC)*, isto é, a restrição $x \in \{0,1\}^n$ foi relaxada para $x \in [0,1]^n$. Desta forma, a relaxação *surrogate contínua* do PCC é dada por

$$
\begin{array}{lll}
 & v(Scont_l) = & min \quad cx \\
(Scont_l) & \text{sujeito a} & \lambda Ax \geq \lambda e \\
 & & x \in [0,1]^n.
\end{array}
$$

O problema (S_l) é um problema da mochila 0-1. A aplicação de algoritmos do tipo "*branch and bound"* pode resolver problemas da mochila 0-1 com milhares de variáveis (veja Martello e Toth [50]), mas sua versão contínua possui algoritmos de tempo polinomial para solução. A relaxação *surrogate contínua* leva vantagem em tempo quando são considerados vários valores diferentes de λ. A sequência de valores para λ foi obtida através de um método subgradientes, mostrando sua relação com a solução de um dual lagrangeano, onde a relaxação *lagrangeana/surrogate* é usada. Este dual não foi completamente explicitado no trabalho de Lopes [46] (ou em Lorena e Lopes [47]). Porém sua comparação com o dual lagrangeano de Beasley mostrou uma sequencia estável de valores, convergindo mais rapidamente e com aproximadamente metade do tempo usado pela heurística de Beasley para as mesmas instâncias.

2.5.3.2 - Relaxação surrogate aplicada ao (PGA)

A relaxação *surrogate* aplicada ao *(PGA)*, definido na *seção 2.4.3*, considerando a restrição das capacidades (2.1), tem a seguinte formulação matemática

$$v(S_l) = \max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}.x_{ij}$$

$$(S_l) \quad \text{sujeito a} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_i.w_{ij}.x_{ij} \leq \sum_{i=1}^{m} \lambda_i.c_i$$

$$\sum_{i=1}^{m} x_{ij} = 1,\ j \in N=\{1,...,n\}$$

$$x_{ij} \in \{0,1\},\ i \in M,\ j \in N.$$

$$\lambda_i \geq 0 \text{ e o vetor } \lambda \text{ é não nulo.}$$

(S_l) pode ser visto como um problema da mochila de múltipla escolha [17]. Sendo *(S_l)* um problema da classe NP-hard [23], como na aplicação ao *(PCC)*, *$(Scont_l)$* foi considerada, relaxando-se a *restrição de integralidade*, isto é, a restrição $x_{ij} \in \{0,1\}$ foi modificada para $x_{ij} \in [0,1]$. Desta forma, tem-se o problema linear da mochila de múltipla escolha. Alguns algoritmos eficientes são conhecidos para resolver este problema [17]. Dudzinski e Waluckiewicz [17] descrevem o algoritmo que foi usado para resolver o problema da mochila de múltipla escolha linear no trabalho de Lorena e Narciso [48]. Para uma melhor compreensão deste algoritmo e sua conexão com a relaxação *lagrangeana/surrogate*, será descrita a seguir a proposta de Dudzinski e Waluckiewicz [17].

O problema dual da relaxação de programação linear de *(Scont $_l$)* pode ser escrito como

$$v(DScont_l) = min \quad (\sum_{i=1}^{m} \lambda_i.c_i).t + \sum_{i=1}^{m} r_i$$

(DScont $_l$) sujeito a $\sum_{j=1}^{n} (\lambda_i.w_{ij}).t + r_i \geq \sum_{j=1}^{n} p_{ij}$, $i \in M$

$t \geq 0$ e t é um número real.

O conjunto de restrições de *(DScont $_l$)* pode ser reescrito como

$$r_i \geq \sum_{j=1}^{n} p_{ij} - \sum_{j=1}^{n} (\lambda_i.w_{ij}).t \text{ , } i \in M.$$

Levando-se em conta que a função objetivo em *(DScont $_l$)* é de minimização, então os menores valores de r_i possíveis ocorrem quando

$$r_i = max \sum_{j=1}^{n} p_{ij} - \sum_{j=1}^{n} (\lambda_i.w_{ij}).t \text{ , } i \in M.$$

Substituindo-se r_i na função objetivo em *(DScont $_l$)*, temos

$$v(DScont_l) = min \{(\sum_{i=1}^{m} \lambda_i.c_i).t + max \sum_{i=1}^{m} \sum_{j=1}^{n} [p_{ij} - (\lambda_i.w_{ij}).t]\}$$

Desta forma, *(DScont $_l$)* pode ser escrito como uma minimização de uma função linear por partes, isto é,

$$v(DScont_l) = min \quad (\sum_{i=1}^{m} \lambda_i.c_i).t + \sum_{i=1}^{m} f_i(t),$$

onde $f_i(t) = max \sum_{j=1}^{n} [p_{ij} - (\lambda_i.w_{ij}).t]$, $i \in M$.

A função objetivo de *(DScont $_l$)* é função de t. A *figura 2.1*, a seguir, apresenta uma representação gráfica do problema:

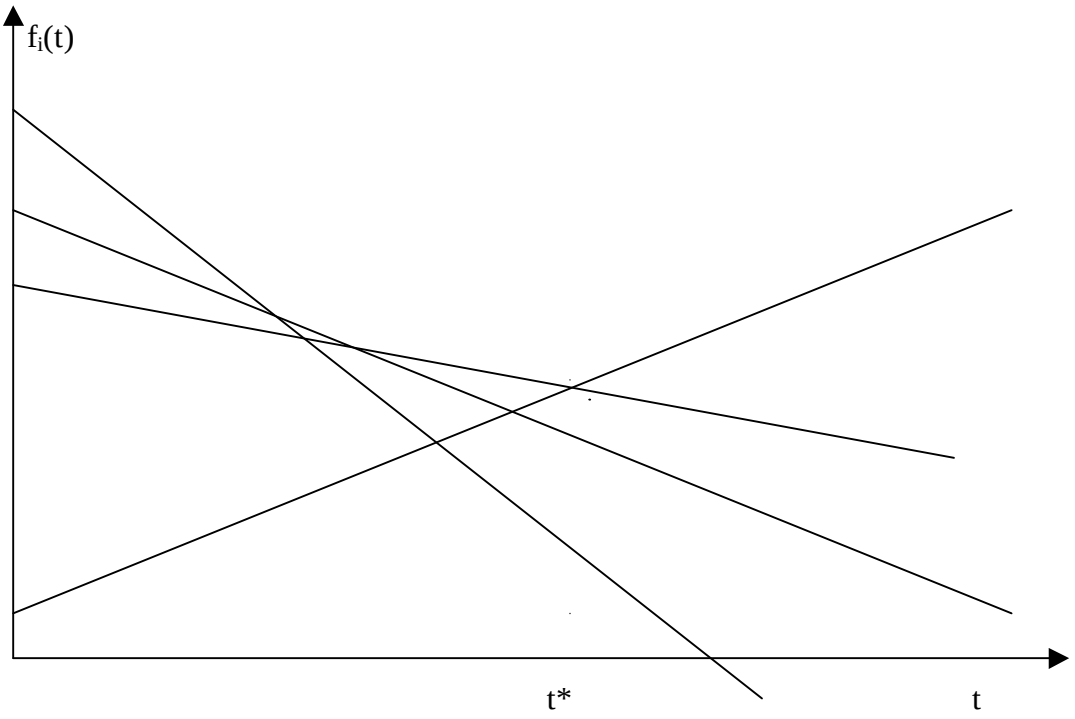

Figura 2.1: Comportamento da função a ser otimizada no dual *surrogate contínuo*

Da *figura 1*, pode-se observar que existe um ponto de mínimo. Seja t* este ponto. Seja $v(t) = \sum f_i(t)$. No ponto de mínimo t*, a derivada a esquerda de v(t*) é negativa. A derivada a direita de v(t*) é positiva. Simbolizando a derivada a esquerda de v(t*) como sendo $v'(t^*)^-$ e a derivada a direita de t* como sendo $v'(t^*)^+$, tem-se que

$$v'(t^*)^- = max \sum_{i=1}^{m} (\lambda_i.w_{ij})$$

$$v'(t^*)^+ = min \sum_{i=1}^{m} (\lambda_i.w_{ij})$$

onde $(\lambda_i.w_{ij})$ é tal que $p_{ij} - (\lambda_i.w_{ij}).t^* = f_i(t^*)$.

Uma vez que, para um dado t = t*, $v'(t^*)^- < 0$ e $v'(t^*)^+ > 0$, o valor de t* é dado por $t^* = (p_{rj^*} - p_{sj^*}) / (\lambda_r.w_{rj^*} - \lambda_s.w_{sj^*})$, onde r, s ∈ M e j* ∈ N.

A solução ótima de *(Scont $_l$)* é dada por:

$x_{ij} = 1$ para os valores de i e j que satisfazem $f_i(t^*) = p_{ij} - (\lambda_i.w_{ij}).t^*$
$x_{rj^*} = q$, $x_{sj^*} = 1 - q$
$x_{rj} = 0$ para os demais valores de i e j.

O valor de q é dado por

$$q = [(\sum_{i=1}^{m} \lambda_i.c_i) - \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq j^*}}^{n} (\lambda_i.w_{ij})] / (\lambda_r.w_{rj^*} - \lambda_s.w_{sj^*})$$

Observa-se que dois valores da solução são fracionários ($x_{rj^*} = q$, $x_{sj^*} = 1 - q$). Caso o valor de q seja 0, então a solução obtida também é solução para a relaxação *surrogate* não contínua.

O dual *surrogate contínuo* foi resolvido fazendo-se uma busca de t=t*, com um valor fixo do vetor λ. Reescrevendo sua função objetivo, tem-se que

$$v(DScont_l) = min \{ max \sum_{i=1}^{m} [\sum_{j=1}^{n} p_{ij} + (c_i - \sum_{j=1}^{n} w_{ij}).\lambda_i.t]\}$$

Uma relaxação lagrangeana pode ser identificada desta expressão, onde o multiplicador lagrangeano seria **l.t.** Esta relaxação será identificada no próximo capítulo como a relaxação *lagrangeana/surrogate*.

Algumas observações a mais podem ser feitas a título de resumo da aplicação do *surrogate contínuo* aos problemas *(PCC)* e *(PGA):*

(i) O vetor λ está fixo na formulação de (S_l). A princípio a relaxação *surrogate contínua* $(Scont_l)$ foi usada com o objetivo de se ter um problema de mais fácil solução;

(ii) Com a variação do vetor λ em um método de otimização por subgradientes, o problema $(Scont_l)$ foi resolvido várias vezes, para diferentes valores de λ;

(iii) O uso do método subgradientes como introduzido na *seção 2.4.4.1* tem seu fundamento teórico de convergência para a relaxação lagrangeana. Portanto, o *surrogate contínuo* é adequado ao uso do método subgradientes pela identificação realizada acima de uma relaxação lagrangeana implícita;

(iv) Outra constatação interessante é a de que para os problemas testados e relaxações consideradas, o conjunto de restrições em questão possui a *propriedade de integralidade*, proporcionando, conforme o *corolário 2.2*, a igualdade dos limites ótimos dos problemas $(DScont_l)$ e um possível dual lagrangeano na variável unidimensional t.

2.5.3.2.1 - Resultados Relaxação surrogate contínua aplicada ao (PGA)

A relaxação *surrogate contínua* obteve melhores resultados (em termos de tempo) do que a relaxação lagrangeana, na média de todos os problemas testados. Foram testados problemas das classes A, B, C e D. Estas classes de problemas estão disponíveis na OR-Library [8] e também estão descritas no *apêndice A*. A grande dificuldade da relaxação *surrogate contínua* foi a sua implementação computacional, a qual foi bem mais complexa que a implementação da relaxação lagrangeana. Maiores detalhes podem ser vistos em Lorena e Narciso [48] e Narciso [52]. As figuras 2.2, 2.3 e 2.4 a seguir mostram o comportamento do dual lagrangeano e o comportamento do dual *surrogate contínuo* para uma instância conhecida por B10x60. Em cada um dos gráficos estão dispostos os pontos relativos à relaxação e os pontos da solução viável obtidos em cada iteração. Observe que o comportamento do dual *surrogate contínuo* é bem mais estável e também converge em um

menor número de iterações do que o dual lagrangeano. Estes gráficos exemplificam o comportamento dos duais lagrangeano e *surrogate contínuo* para as instâncias da classe B. Este comportamento foi observado também para as instâncias das classes A, C e D.

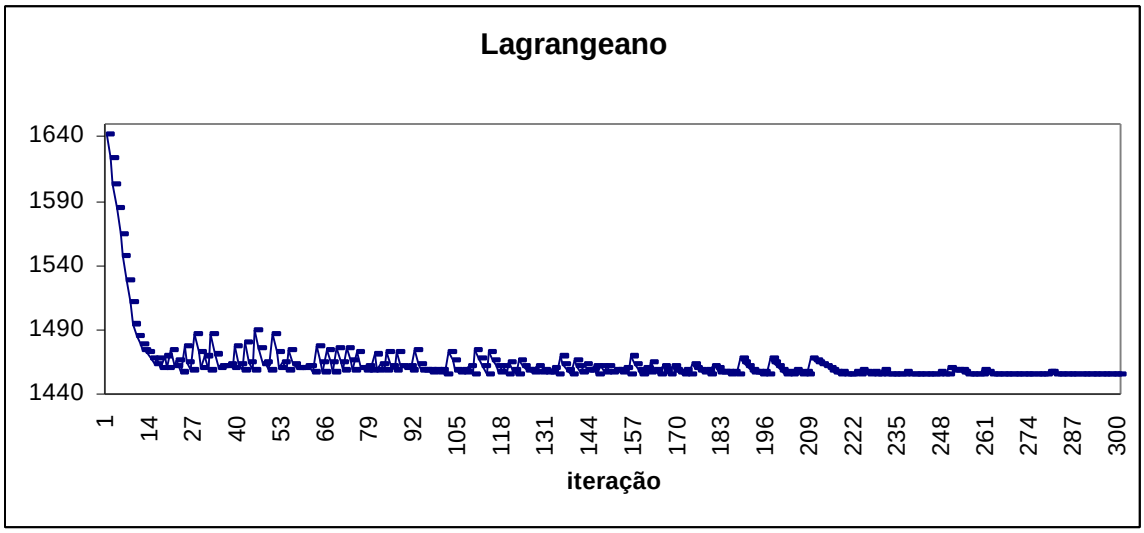

Figura 2.2 - comportamento do dual lagrangeano

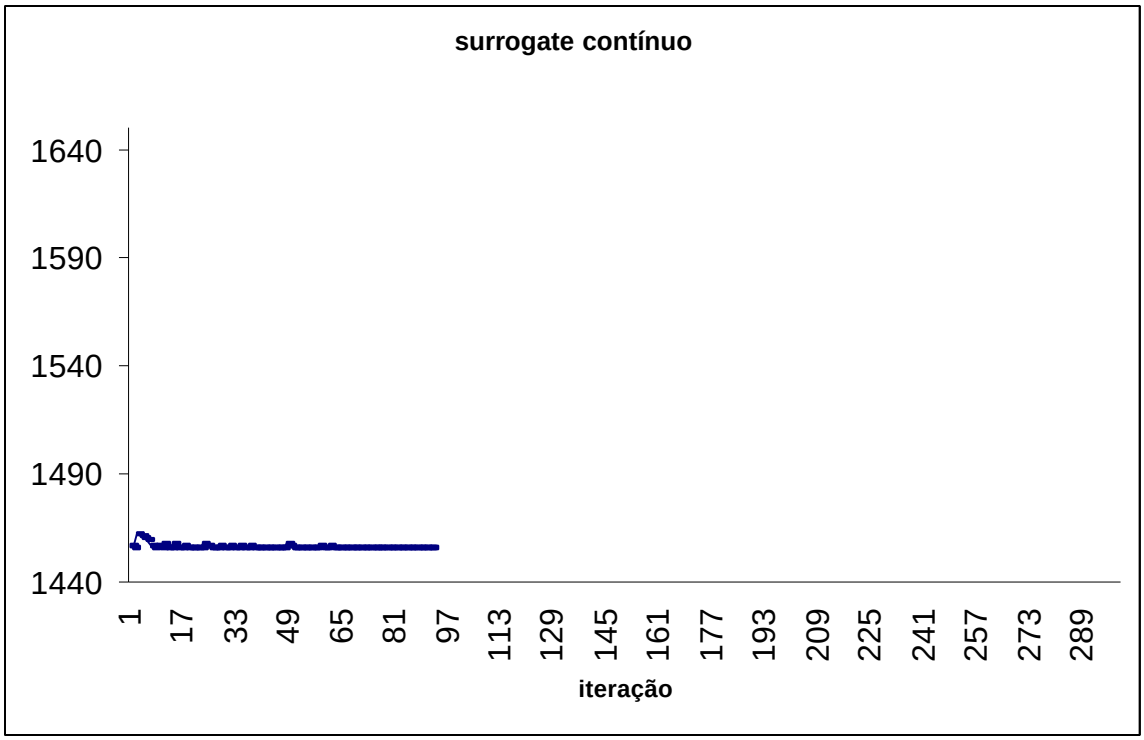

Figura 2.3 - comportamento do dual surrogate contínuo

Para uma melhor visualização da figura 2.3, está abaixo a figura 2.4 com o eixo da ordenada modificado.

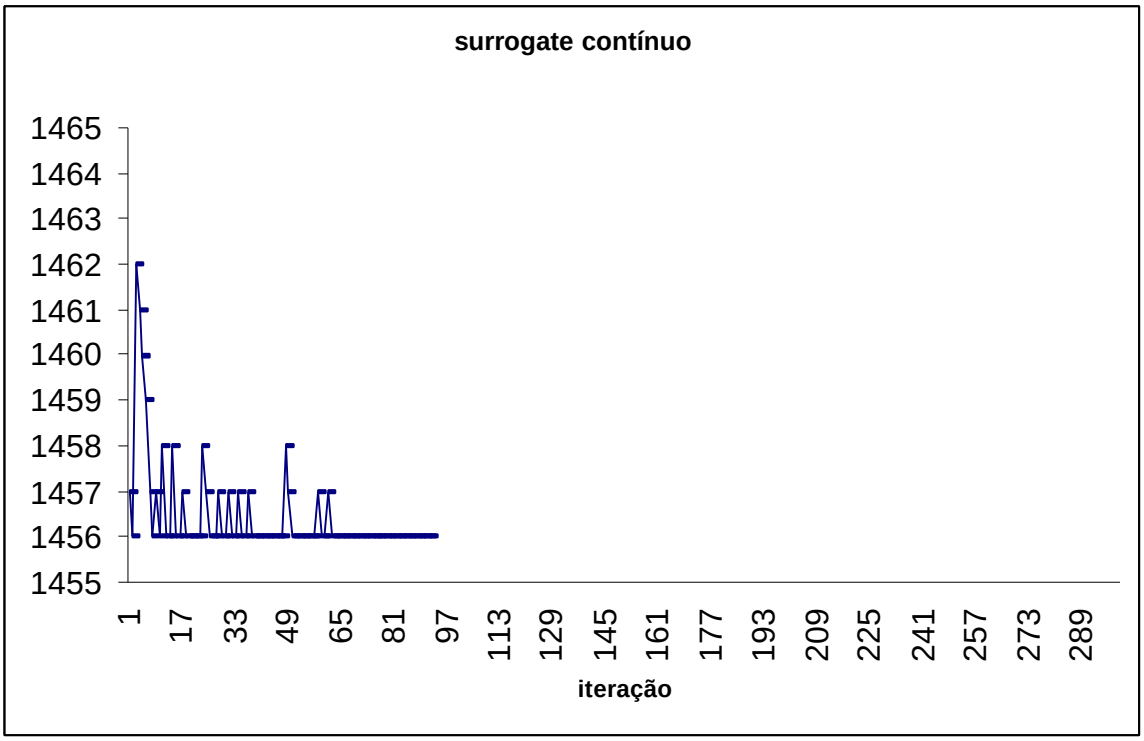

Figura 2.4 - comportamento do dual surrogate contínuo

CAPÍTULO 3

3 - Relaxação lagrangeana/surrogate (lagsur)

3.1 - Introdução

A relaxação *lagrangeana/surrogate (lagsur)* é uma nova proposta para problemas de Otimização Combinatória. As relaxações lagrangeana e surrogate são combinadas com objetivo de conseguir melhores tempos computacionais na aplicação de heurísticas subgradientes.

Para um dado problema de Otimização Combinatória, inicialmente é derivada uma relaxação surrogate de um conjunto adequado de restrições. Uma relaxação lagrangeana da restrição surrogate é então obtida. Vale a pena mencionar que neste caso o multiplicador lagrangeano é uni-dimensional.

Considerando-se a aplicação de um método subgradientes com a relaxação lagrangeana usual e a *lagsur*, dado um mesmo multiplicador inicial para as duas relaxações, a relaxação *lagsur* nos proporciona um conjunto maior de informações locais que a relaxação usual lagrangeana. Uma otimização local do multiplicador lagrangeano da relaxação *lagsur* é capaz de proporcionar limites locais de melhor qualidade.

A relaxação *lagsur* foi inspirada nos trabalhos de Lorena e Lopes [47] e Lorena e Narciso [48] observando-se os resultados da aplicação das relaxações *surrogate contínua* e lagrangeana para o *(PCC)* e *(PGA)*, respectivamente. Foi constatado que na aplicação de um método subgradientes, usando o mesmo multiplicador inicial, sobre os mesmos dados, a relaxação *surrogate contínua* proporcionava uma sequência mais estável que a correspondente da relaxação lagrangeana, convergindo mais rapidamente. Além disso, os

limites obtidos com a aplicação da relaxação *surrogate contínua* eram tão bons quanto os obtidos com a relaxação lagrangeana.

Para os problemas estudados, isto é, o problema de cobertura de conjuntos e o problema generalizado de atribuição, as relaxações lagrangeanas consideradas possuiam a *propriedade de integralidade*, existindo uma coincidência dos limites obtidos com o dual da relaxação surrogate contínua e o dual lagrangeano. Mais adiante neste capítulo, com a definição formal da relaxação *lagsur*, iremos mostrar que neste caso, com o uso da relaxação *surrogate contínua*, estamos usando a informação local (otimização) em todos os passos do correspondente método subgradientes.

A relaxação *lagsur* fornece um método geral para trabalhar com relaxações que satisfaçam ou não a *propriedade de integralidade* e, em princípio, pode ser usada com uma classe de métodos de subgradientes, conforme será mostrado mais adiante neste capítulo. A direção do subgradiente é, em geral, diferente da fornecida pela direção correspondente dada pela relaxação lagrangeana.

Conforme pode ser constatado nos resultados relativos às instâncias do *(PGA)* e do *(PCV)* (*capítulos 4, 5 e Apêndice A*), a relaxação *lagsur* é apropriada para problemas com um grande número de variáveis. Quanto maior for o número de variáveis do problema, melhor será o desempenho da relaxação *lagsur* em relação à relaxação lagrangeana no que se refere a tempo de execução. Esta comparação da relaxação lagrangeana com a relaxação *lagsur* é adequada visto que a relaxação *lagsur* é também uma relaxação lagrangeana.

3.2 - Formulação matemática da relaxação lagsur

Para expor matematicamente a relaxação *lagsur*, vamos considerar novamente o problema *(P)*, tal como foi definido na relaxação no *capítulo 2*.

$$
\begin{array}{lll}
 & v(P) = & \textit{min}\ cx \\
(P) & \text{sujeito a} & Ax = b \\
 & & Dx \leq e \\
 & & x \geq 0 \text{ e inteiro.}
\end{array}
$$

Considere ainda a relaxação surrogate (S_l) de (P) em relação ao conjunto de restrições $Dx \leq e$, considerando-se um vetor de multiplicadores $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k)$, onde $\lambda_i \geq 0$, i = 1,..,k.

$$
\begin{array}{lll}
 & v(S_l) = & \textit{min}\ cx \\
(S_l) & \text{sujeito a} & Ax = b \\
 & & \lambda Dx - \lambda e \leq 0, \\
 & & x \geq 0 \text{ e inteiro.}
\end{array}
$$

Desta forma, as k restrições de $Dx \leq e$ foram substituídas pela restrição $\lambda Dx - \lambda e \leq 0$. Seja o conjunto J definido por $J = \{ x \geq 0 \text{ e inteiro} \mid Ax = b \}$. Relaxando a *restrição de integralidade*, isto é, $x \geq 0$ e x é um número real, estão teremos a relaxação *surrogate contínua* $(Scont_l)$ e $v(Scont_l P) \leq v(S_l)$.

Seja agora relaxar a restrição *surrogate* de (S_l) no modo lagrangeano. Visto que a restrição *surrogate* possui apenas uma dimensão, precisa-se de um multiplicador de apenas uma dimensão. Seja $t \geq 0$ o multiplicador lagrangeano. Desta forma, após relaxar a restrição *surrogate*, temos a relaxação *lagsur* com a seguinte formulação:

$$
\begin{array}{lll}
 & v(L_tS_l) = & \textit{min}\ \ cx - t.(\lambda Dx - \lambda e) \\
(L_tS_l) & \text{sujeito a} & x \in J
\end{array}
$$

Reescrevendo a função objetivo, temos:

$$
\begin{array}{lll}
 & v(L_tS_l) = & \textit{min}\ \ (c - t.\lambda.D).x + t.\lambda.e \\
(L_tS_l) & \text{sujeito a} & x \in J
\end{array}
$$

Visto que (L_tS_l) é uma relaxação de (S_λ), então temos: $v(L_tS_l) \leq v(S_l) \leq v(P)$.

O dual *lagsur* é dado por

$$(DL_tS_l) \qquad v(DL_tS_l) = max\ v(L_tS_l) \quad \text{sujeito a} \quad t.\lambda \geq 0.$$

É imediato que, fazendo-se $u = t.\lambda$, o problema (DL_tS_l) é o dual lagrangeano (DL_u) definido na *seção 2.4.4*. Desta forma, os limites inferiores coincidem para o dual *lagsur* e para o dual lagrangeano.

Um dual local também pode ser obtido usando-se o *lagsur*. Para um multiplicador λ, fixado em um valor λ^*, o melhor valor de t para o qual $v(L_tS_{l^*})$ seja máximo é dado por

$$(DL_tS_l^*) \qquad v(DL_tS_l^*) = max\ v(L_tS_{l^*}) \quad \text{sujeito a} \quad t \geq 0.$$

Se o conjunto J satisfaz a *propriedade de integralidade*, então $v(DL_tS_l^*) = v(Scont_{l^*})$ (corolário 2.2). Em geral, temos $v(L_tS_l) \leq v(DL_tS_l^*) \leq v(DL_tS_l) = v(DL_u) \leq v(P)$.

Uma característica interessante da relaxação *lagsur* é que, usando-se t = 1, tem-se a relaxação lagrangeana usual para a restrição $Dx \leq$ e referente ao multiplicador λ. Desta forma, para t = 1 , temos:

$$(L_1S_l) \qquad v(L_1S_l) = min\ (c - \lambda.D).x + \lambda.e \quad \text{sujeito a} \quad x \in J$$

O problema (L_1S_l) em nossa notação poderia ser escrito ainda como (L_l) (veja *seção 2.4.2*). Handler e Zang [32] e Minoux [51] propuseram uma forma exata de resolver $(DL_tS_l^*)$, e consequentemente obter o melhor valor local do multiplicador t (dado que $v(L_tS_l) \leq v(DL_tS_l^*)$). O custo computacional para se obter o melhor t, conforme a proposta de Handler e Zang é alto. Entretanto, nem sempre será necessário se obter o melhor valor de t, bastando obter um valor de t que leve a um resultado melhor do que para t = 1.

Vamos considerar a figura abaixo, a qual ilustra o comportamento de $v(L_t S_{l^*})$ em função de t para λ^* fixo.

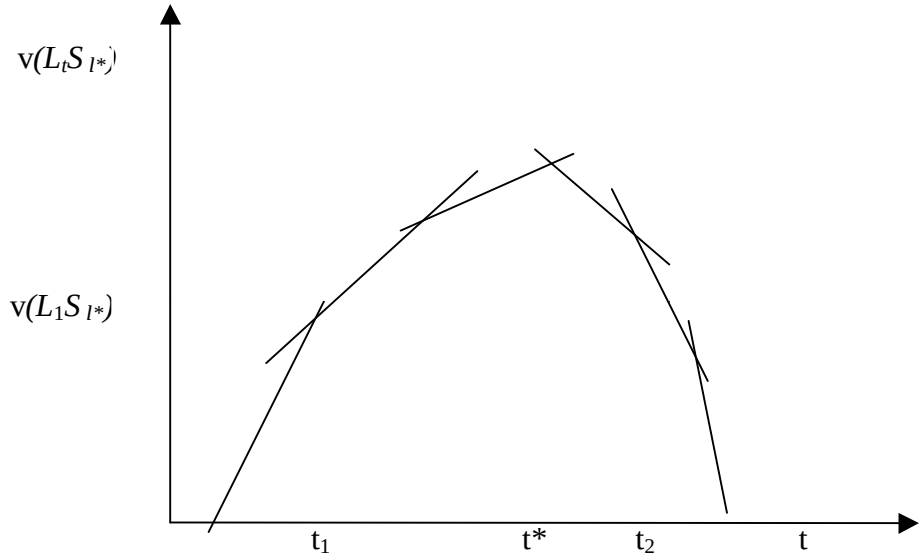

Figura 3.1 - Comportamento da função objetivo da relaxação *lagsur*

Na *figura 3.1,* tem-se que a função a ser otimizada na relaxação *lagsur* é côncava e linear por partes (teorema 2.4). Além disso, o valor de t = t* que torna $v(L_t S_{l^*})$ máximo situa-se no intervalo $t_1 < t < t_2$. Supondo $t_1 = 1$, se obtivermos um valor de t no intervalo $t_1 = 1 < t < t_2$, então $v(L_t S_{l^*}) > v(L_1 S_{l^*})$. O mesmo raciocínio pode ser repetido quando $t_2 = 1$. Um algoritmo de busca unidimensional poderia ser empregado aqui, e a obtenção do valor de t nos intervalos considerados teria um custo computacional menor que o da resolução de $(DL_t S_l^*)$.

O dual *lagsur* $(DL_t S_l)$ pode ser resolvido usando-se o método subgradientes. Localmente, a relaxação *lagsur* pode fornecer melhores limites do que a correspondente relaxação lagrangeana (L_l). As regras de convergência do método subgradientes, definidas no *teorema 2.6* do capítulo 2, bem como os resultados de Ermol'ev [19], Poljak [56], Held e Karp [33,34], também valem para o *lagsur*.

3.3 - Aplicação do algoritmo de subgradientes

Para a aplicação do algoritmo de subgradientes apresentado na *seção 2.4.4.1*, alguns de seus passos foram reescritos para que o mesmo algoritmo fosse usado com as duas relaxações, a lagrangeana e a *lagsur*. Depois de resolvida a respectiva relaxação, sua solução é tornada viável ao problema *(P)* aplicando uma heurística convenientemente elaborada visando a aplicação em estudo. Os limites superior e inferior são atualizados e usados na fórmula de atualização do passo na direção do subgradiente. Alguns critérios de parada adicionais aos apresentados no algoritmo da *seção 2.4.4.1* podem ser derivados. O algoritmo de subgradientes foi então reescrito como:

Algoritmo de subgradientes - versão II
Seja $\lambda \geq 0$,
Fazer lb $\leftarrow -\infty$ e ub $\leftarrow +\infty$
Enquanto {não para as condições de parada} ***fazer***
  Resolver *(Rel $_l$)*. **Seja** $x_\lambda \in J$ a solução ótima obtida.
  Obter uma solução viável x_v para *(P)* a partir de x_λ e o valor c. x_v,
  Fazer lb $\leftarrow$ *máx* (lb, v*(Rel $_l$)*),
    ub $\leftarrow$ *min* (ub, c.x_v).
  Atualizar a direção do subgradiente g^l, o tamanho do passo p, e o multiplicador λ;
  Testar condições de parada,
fim_enquanto.

Deste algoritmo, algumas considerações devem ser feitas:

(i) A relaxação *(Rel $_l$)* pode ser uma das duas relaxações: *(L $_l$)* (lagrangeana) ou *(L$_t$S $_l$)* (*lagsur*). O uso da relaxação lagrangeana pode ser considerado nesse caso como a *lagsur*, onde o multiplicador t foi fixado no valor 1 em todas as iterações do algoritmo de subgradientes - versão II. A relaxação *lagsur* envolve ainda a aplicação de um algoritmo para o cálculo do multiplicador t. Este algoritmo pode ser um de busca unidimensional ou eventualmente aquele que calcula o melhor multiplicador t de uma forma exata (como foi feito no caso das aplicações apresentadas considerando o *(PCC)* e o *(PGA)*). No caso da

aplicação do *lagsur* ao *(PGA)*, que será descrito no *capítulo 4*, um algoritmo de busca unidimensional foi aplicado durante algumas iterações do algoritmo subgradientes - versão II. Já no caso da aplicação ao *(PCV)* o método de busca unidimensional foi aplicado apenas na primeira iteração do algoritmo.

(ii) A atualização do multiplicador λ e do tamanho do passo é feita da seguinte maneira:

$$\lambda_i \leftarrow max \{0, \lambda_i + p.g_i^{\lambda} \}, i = 1,2,...,k,$$

onde g^{λ} é o subgradiente selecionado e $p > 0$ é o tamanho do passo. O tamanho do passo mais usado é $p = \pi.(ub - lb)/|| g^{\lambda} ||^2$. O valor de π é um número real e $0<\pi\leq 2$. O valor de π é variado durante as iterações do método dos subgradientes. Inicialmente, π começa com o valor 2. Após um determinado critério ter sido atingido, π passa a ser $\pi/2$ (o valor atual é dividido pela metade).

No caso da relaxação *lagsur*, o multiplicador $t_m.\lambda$ (onde t_m é o melhor valor do parametro t obtido por busca unidimensional ou método exato) está sendo atualizado indiretamente. Suponha t_m calculado na primeira iteração do algoritmo de subgradientes - versão II, e então fixado nas demais iterações. Desta forma, uma correspondente regra de atualização para os multiplicadores $t_m.\lambda$ é obtida substituindo-se p por $t_m.p$ (t_m é um multiplicador escalar). Por outro lado, se t_m é calculado em cada iteração (ou em algumas iterações) do algoritmo de subgradientes - versão II, um efeito semelhante poderia ser constatado.

(iii) Critérios de parada:

- Número de iterações > valor convenientemente escolhido, ou
- $\pi \leq 0.005$, ou
- ub - lb < 1, ou
- a parte inteira de lb não mudar após 30 iterações consecutivas.

(iv) A direção do vetor de subgradientes obtida na solução do problema $(L_t S_l)$, é em geral diferente da correspondente direção obtida pela solução do problema (L_l).

Iniciando o método subgradientes com o mesmo multiplicador inicial λ_0 , diferentes sequências de limites são obtidas para cada relaxação.

(v) Outros métodos subgradientes apareceram na literatura [10,11,14,41,42]. Mais elaborados, eles aumentam os tempos computacionais computando direções descendentes [14], ou combinando subgradientes de iterações anteriores [10,11], ou realizando projeções em conjuntos convexos [41,42,44]. Resultados experimentais com alguns destes métodos mostraram uma melhora no desempenho quando comparados ao método subgradientes [41,44]. O método subgradientes ainda é largamente usado para se obter o dual da relaxação lagrangeana.

3.3.1 - Comentários sobre o algoritmo de subgradientes

Este algoritmo tem uma série de fatores que influenciam o desempenho. Estes fatores são:

- *O tipo de relaxação do problema*. Como exemplo, considere o *(PGA)*, e três maneiras diferentes de se usar a relaxação lagrangeana (vide próximo capítulo). Observa-se que, para uma mesma instância, o tempo para se resolver o dual de cada tipo diferente de relaxação lagrangeana é diferente e os valores dos limites obtidos também são diferentes.
- *Os valores iniciais usados para os multiplicadores.* A convergência mais rápida ou lenta do algoritmo de subgradientes está ligada também aos valores iniciais dos multiplicadores λ. Entretanto, não é possível prever que valor de λ inicial seria eficaz para cada instância;

- *O tamanho do passo*. O controle do tamanho do passo é importante, pois pode influenciar no comportamento da relaxação, principalmente no que se refere a estabilidade e convergência;

- *As condições de parada do algoritmo*. Elas devem ser convenientemente escolhidas de tal forma que, ao término do algoritmo, a sequência de valores fornecidos pela relaxação já tenha se estabilizado.

- *Controle do valor do parâmetro "p".* Há casos em que o parâmetro "π" do tamanho passo "p" rapidamente diminui. Isto pode comprometer a qualidade do resultado do dual da relaxação em questão. Como um dos critérios de parada é o valor de $\pi < 0.005$, então o algoritmo pode parar antes do algoritmo de subgradientes conseguir um valor razoável. Desta forma, é necessário controlar o decaimento de "π" para que este não fique pequeno demais, pois a solução final resultante pode não ser a melhor que o algoritmo de subgradientes possa obter.

CAPÍTULO 4

4. Problema Generalizado de Atribuição

4.1 - Introdução

O problema de se maximizar o lucro (benefício, rendimento, etc.) ao se atribuir n tarefas a m agentes, n > m, tal que cada tarefa seja atribuída a apenas um único agente, levando-se em conta as restrições de capacidade de cada agente, é conhecido na literatura como problema generalizado de atribuição *(PGA)*. Muitos problemas da vida real podem ser modelados como um *(PGA)*. Podemos citar como exemplos, problemas de investimento de capitais [16], alocação de espaço de memória [12], projeto de redes de comunicação com restrições de capacidades para cada nó da rede [16], atribuição de tarefas de desenvolvimento de software para programadores [3], atribuição de tarefas a computadores em uma rede [3], subproblemas de roteamento de veículos [22] , e outros. Como uma consequência lógica de sua aplicação, um grande esforço tem sido realizado no desenvolvimento de algoritmos para este problema [39].

Este problema é NP-hard [12,23] e poucas heurísticas existem na literatura para a resolução deste problema de forma a encontrar boas soluções viáveis. Dentre as que mais se destacam podemos citar as de Klastorin [43], Martello e Toth [50] e Lorena e Narciso [48].

Em Lorena e Narciso [48] temos alguns resultados da relaxação lagrangeana e também da relaxação *surrogate contínua* aplicadas ao *(PGA)*. Com instâncias do *(PGA)*, obtidas da OR-Library [8], foram comparadas as duas relaxações (*surrogate contínua* e *lagrangeana*) e a relaxação *surrogate contínua* mostrou-se ser melhor tanto em tempo computacional como em limites de relaxação. A seqüência de limites de relaxação obtida com a aplicação do algoritmo de subgradientes - versão II, mostrou-se mais estável no caso da relaxação *surrogate contínua*.

A aplicação do trabalho de Lorena e Narciso se restringiu à relaxação lagrangeana do *(PGA)* apresentada na *sessão 2.4.3*. Neste caso particular de relaxação das restrições de capacidades, o problema resultante apresenta um conjunto de restrições que satisfaz a *propriedade de integralidade*, onde foi usado um algoritmo exato de resolução do dual *(DL_tS_l*)* (resumido na *sessão 2.5.3.2*).

Nesta sessão, serão usadas três relaxações diferentes para o *(PGA)*. O desempenho da relaxação lagrangeana e da relaxação *lagsur* será mostrado para um conjunto maior de instâncias para o *(PGA)*, incluindo instâncias de grande porte, todas obtidas da OR-Library [8]. Duas das relaxações não apresentam a *propriedade de integralidade* e consequentemente seus limites são melhores que os do trabalho de Lorena e Narciso [48]. Além disso, um algoritmo de busca unidimensional foi usado para o cálculo do melhor parâmetro *lagsur* em algumas iterações do algoritmo subgradientes-versão II. A heurística construtiva, que torna a solução da relaxação viavel, foi melhorada. Os resultados foram melhores que os do trabalho de Lorena e Narciso [48], tanto em limites de relaxações quanto nos de soluções viáveis, e estão apresentados em um artigo recente (Narciso e Lorena [53]).

4.2 - Relaxações lagrangeanas e lagsur

O *(PGA)* foi formulado matematicamente na *sessão 2.4.3*, e será relembrado aqui para melhor continuidade do texto:

$$v(PGA) = max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}.x_{ij}$$

(PGA)

$$\text{sujeito a} \quad \sum_{j=1}^{n} w_{ij}.x_{ij} \le b_i, \quad i \in M = \{1,...,m\} \qquad (4.1)$$

$$\sum_{i=1}^{m} x_{ij} = 1, \; j \in N = \{1,...,n\} \qquad (4.2)$$

$$x_{ij} \in \{0,1\}, \; i \in M, \; j \in N \qquad (4.3)$$

As restrições (4.1) impõem que os pesos dos itens escolhidos não devem exceder à capacidade de cada mochila e as restrições (4.2) impõem que cada item deve ser atribuído a somente uma mochila.

A relaxação lagrangeana aplicada ao *(PGA)* pode ser vista inicialmente de duas maneiras, isto é, relaxando-se as restrições das capacidades (4.1) ou relaxando-se as restrições de atribuição (4.2). Vamos considerar inicialmente a relaxação das restrições (4.1), reproduzindo a relaxação usada em Lorena e Narciso [48] e também apresentada na *sessão 2.4.3.*

Seja λ um vetor de multiplicadores $\lambda_i \geq 0$ e $i \in M$. A primeira relaxação lagrangeana será

$$v(L_lcPGA) = max \sum_{i=1}^{m} \sum_{j=1}^{n} \{(p_{ij} - \lambda_i . w_{ij}).x_{ij}\} + \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_i . c_{ij}$$

(L_lcPGA)

sujeito a (4.2) e (4.3);

e para um multiplicador multiplicador $t \geq 0$, a correspondente *lagsur*

$$v(L_tS_lcPGA) = max \sum_{i=1}^{m} \sum_{j=1}^{n} \{(p_{ij} - t.\lambda_i . w_{ij}).x_{ij}\} + \sum_{i=1}^{m} \sum_{j=1}^{n} t.\lambda_i . c_{ij}$$

(L_tS_lcPGA)

sujeito a (4.2) e (4.3)

Para a segunda relaxação consideram-se outros multiplicadores $\lambda_j \in \Re$ e $j \in N$.

$$v(L_{\lambda}aPGA) = max \sum_{i=1}^{m} \sum_{j=1}^{n} \{(p_{ij} + \lambda_i).x_{ij}\} - \sum_{j=1}^{n} \lambda_j$$

$(L_{\lambda}aPGA)$

sujeito a (4.1) e (4.3);

e a correspondente *lagsur* (para t irrestrito neste caso)

$$v(L_tS\lambda aPGA) = max \sum_{i=1}^{m} \sum_{j=1}^{n} \{(p_{ij} + t.\lambda_i).x_{ij}\} - \sum_{j=1}^{n} t.\lambda_j$$

$(L_tS\lambda aPGA)$

sujeito a (4.1), (4.3)

Para a resolução de $(L_{\lambda}aPGA)$, basta verificar que, para cada $i \in M$, existe um problema da mochila a ser resolvido. Desta forma, podemos decompor o problema como:

$$v(L_{\lambda}aPGA)_i = max \sum_{j=1}^{n} \{p_{ij}.x_{ij} + \lambda_j.(1 - . x_{ij})\}$$

$(L_{\lambda}aPGA)_i$

sujeito a (4.1) e (4.3)

O valor de $v(L\lambda aPGA)$ seria o somatório dos valores dos m problemas da mochila, isto é,

$$v(L_{\lambda}aPGA) = \sum_{i=1}^{m} v(L_{\lambda}aPGA)_i$$

Uma interpretação semelhante poderá ser dada no caso da relaxação *lagsur*, considerando-se t fixo.

Das duas relaxações diferentes apresentadas até agora temos a seguinte relação: $v(L\lambda aPGA) \leq v(L\lambda cPGA)$ (*propriedade de integralidade*). Nos resultados obtidos com

instâncias do (*PGA*), descritos no *apêndice A*, isto pode ser visto claramente. Entretanto, a relaxação *(L ¡aPGA)* é de mais difícil solução que a relaxação *(L ¡cPGA),* visto que m problemas da mochila deverão ser resolvidos para se determinar o valor de *(L ¡aPGA).* Para o caso de *(L ¡cPGA),* o problema se resume o de escolher, para cada j, o maior valor de (p_{ij} - $\lambda_i w_{ij}$), para i variando de 1 a m (idem para a correspondente *lagsur* com t fixo) e, desta forma, este problema é resolvido de forma exata. Vale a pena mencionar que o problema da mochila deve ser resolvido de forma exata. Para resolver de forma exata os m problemas da mochila, têm-se várias opções de algoritmos na literatura. O algoritmo usado para se determinar os m problemas de forma exata foi o proposto por Horowitz e Sahni [50].

Em 1986, Jornsten e Nasberg [38] propuseram uma nova relaxação lagrangeana que combina as relaxações *(L ¡aPGA)* e *(L ¡cPGA).* Esta nova relaxação esta descrita a seguir.

Seja o seguinte problema:

$$v(JN) = max \quad \alpha \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} \cdot x_{ij} + \beta \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} \cdot y_{ij}$$

(*JN*)

sujeito a (4.1)

$$\sum_{i=1}^{m} y_{ij} = 1, \; j \in N = \{1,...,n\} \qquad (4.4)$$

$$y_{ij} = x_{ij}, \text{ para } i = \{1,...,m\} \text{ e } j = \{1,...,n\} \qquad (4.5)$$

$$x_{ij} \in \{0,1\}, \; i \in M, \; j \in N \qquad (4.6)$$

$$y_{ij} \in \{0,1\}, \; i \in M, \; j \in N \qquad (4.7)$$

Na expressão da função objetivo, α e β são positivos. Pode-se afirmar também que v(*JN*) = (α+β).v(PGA). Se α+β=1, então tem-se que v(*JN*)=(α+β).v(*PGA*)=1.v(*PGA*) = v(*PGA*). No *apêndice A*, e resumidamente neste capítulo, são apresentados uma série de

resultados obtidos com a proposta de Jornsten e Nasberg. Para estes resultados, os valores de α e β são iguais a 0.5.

Para um multiplicador λ (cujos componentes λ_{ij} são números reais), relaxando as restrições (4.5) conforme a relaxação lagrangeana, teremos:

$$v(L_\lambda JN) = max \left[\alpha \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}.x_{ij} + \beta \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}.y_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_{ij}(y_{ij} - x_{ij})\right]$$

$(L_\lambda JN)$ sujeito a (4.1), (4.4), (4.6) e (4.7).

Considerando ainda um multiplicador t (irrestrito em sinal), tem-se a correspondente relaxação *lagsur*

$$v(L_t S_\lambda JN) = max \left[\alpha \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}.x_{ij} + \beta \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}.y_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{n} t.\lambda_{ij}(y_{ij} - x_{ij})\right]$$

$(L_t S_\lambda JN)$ sujeito a (4.1), (4.4), (4.6) e (4.7).

O problema $(L_\lambda JN)$ podem ser separado em dois outros problemas, $(L_\lambda XJN)$ e $(L_\lambda YJN)$. O primeiro está descrito a seguir.

$$v(L_\lambda XJN) = max \sum_{i=1}^{m} \sum_{j=1}^{n} (\alpha.p_{ij} - \lambda_{ij}).x_{ij}$$

$(L_\lambda XJN)$ sujeito a (4.1) e (4.6)

O segundo problema tem a seguinte formulação matemática:

$$v(L_\lambda YJN) = max \sum_{i=1}^{m} \sum_{j=1}^{n} (\beta.p_{ij} + \lambda_{ij}).y_{ij}$$

$(L_\lambda YJN)$ sujeito a (4.4) e (4.7)

Observe que o primeiro problema apresenta o conjunto de restrições variáveis x_{ij} e o segundo com as variáveis y_{ij}. Além disso, temos que v*(L ¡JN)* = v*(L ¡YJN)* + v*(L ¡XJN)*, conforme descrito em Jornsten e Nasberg [38]. Esta relaxação lagrangeana é também conhecida como decomposição lagrangeana [38].

Os problemas acima são semelhantes aos descritos nas relaxações anteriores e são resolvidos da mesma forma descrita para essas relaxações.

A relaxação *(L ¡JN)* tem a característica de obter limites melhores que as relaxações *(L ¡aPGA)* e *(L ¡cPGA)*. Entretanto, a convergência é bem mais lenta. Isto foi comprovado nos resultados obtidos com os problemas testes para o *(PGA)*. Maiores informações sobre a relaxação descrita acima pode ser vista em Jornsten e Nasberg [38].

Cada uma destas relaxações foi comparada com a respectiva relaxação *lagsur* e os resultados computacionais estão apresentados na *seção 4.6* e no *apêndice A*.

4.3 - Busca unidimensional

Nas três relaxações *lagsur* propostas para o *(PGA)*, uma busca local foi empregada para se aproximar o melhor valor de t. Para um dado λ fixo, temos que a função objetivo da relaxação *lagsur* é linear por partes. A figura 4.1 ilustra, qualitativamente, o comportamento da função v*(L$_t$S ¡PGA)* versus t. Observe que a função v*(L$_t$S ¡PGA)* é linear por partes (teorema 2.4).

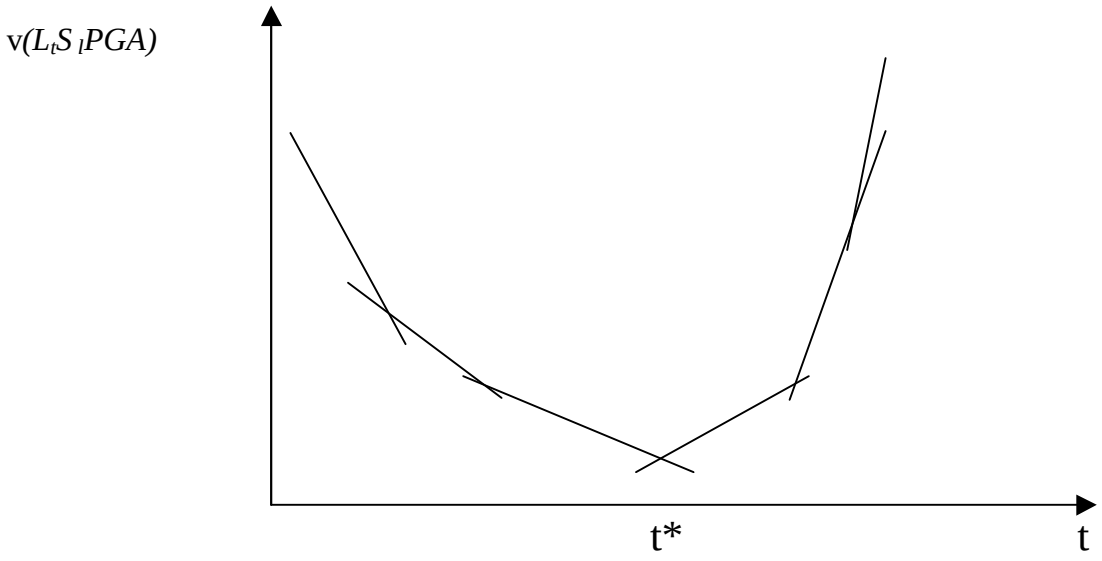

Figura 4.1 - comportamento da função v(L_tS_lPGA)

Não existe uma expressão analítica que obtenha o melhor valor de t diretamente tal que minimize v(L_tS_lPGA). No *capítulo 3* foram citados Minoux [51] e Handler e Zang [32], os quais propuseram métodos para se obter esse valor. Entretanto, estes métodos consomem muito esforço computacional. Desta forma, como foi descrito no *capítulo 3*, busca-se uma forma alternativa de se obter um valor de t tal que v(L_tS_lPGA) < v(L_1S_lPGA) e este valor de t seja o mais próximo possível do melhor valor de t em poucas iterações. Existem na literatura uma série de propostas para se obter o valor de t quando se conhece o intervalo no qual ele se encontra. Entretanto, para o caso do *lagsur*, há de se levar em conta o fato de que não se sabe previamente um intervalo no qual t esteja contido. Desta forma, é necessário obter t através de um método que seja independente dos valores de um intervalo. Uma proposta de se obter um valor de t é descrita a seguir:

1 . Defina um valor de t inicial t = t_0. Calcule o valor do *lagsur* para este t=t_0.

2 . Se a inclinação da função v(L_tS_lPGA) for negativa, isto indica que o valor de t* é maior que t. Desta forma, um possível limite inferior do intervalo no qual t*

está é $t_1 = t$. Se a inclinação da função $v(L_t S_l PGA)$ for positiva, isto indica que o valor de t* é menor que t. Desta forma, o limite superior do intervalo no qual t* está é $t_2 = t$.

3 . Conforme a avaliação do passo 2, o valor de t deverá sofrer um acréscimo ou um decréscimo para uma nova avaliação do *lagsur*. Após calcular este novo valor para t, então voltar para o passo 2 caso o critério de parada não seja satisfeito. Em cada iteração deste método, verificar qual é o menor valor de $v(L_t S_l PGA)$ e guardá-lo. Quando o critério de parada for atendido, tem-se então o menor valor de $v(L_t S_l PGA)$ e o respectivo valor de t.

Em relação ao critério de parada, para o *(PGA)*, levou-se em consideração:

- o limite máximo de iterações e
- a diferença entre t_2 e t_1 ser menor que δ, δ é definido conforme o grau de precisão que se queira obter para se obter t*.

Para o *(PGA)*, usando uma série de instâncias, observou-se que, para uma média de 11 iterações, a diferença entre t_2 e t_1 tornava-se menor que 0.05. Entretanto, isto custa uma quantidade considerável de tempo. Desta forma, para os testes feitos para o *(PGA)*, o limite máximo de iterações para se obter o melhor valor de t foi 5 e o valor de δ foi 0.10.

Feitas as considerações acima, pode-se então descrever o algoritmo de busca do melhor t que foi usado para o *(PGA)*.

Algoritmo de busca do melhor t

Dados *incremento = 0.10;*
k_max = 5 (número máximo de iterações);
t_1 := -¥ (limite inferior para o melhor t);
t := incremento (multiplicador lagrangeano/surrogate atual);
t_2 := ¥ (limite superior para o melhor t);
v := (melhor limite inferior atual). Inicialmente, v* := +¥ ;*
k := 0 (número de iterações atual);
delta = 0.10;

Repita
Fazer *k := k + 1;*
Se *k > k_max* ***ou*** *(t_2 - t_1) < delta* ***então pare;***
Senão resolva *($L_t S_l PGA$)*
Fim_Se
Se *$v(L_t S_l PGA)$ < v** ***então***
Fazer *melhor_t = t,*
v := $v(L_t S_l PGA)$;*
Fim_Se
Calcule *w^l (w^l é a inclinação da função lagsur);*
Se *w^l > 0* ***então***
t_2 = t;
t = t - incremento;
Se *t_1 ¹ -¥ (t_1 já tem valor definido)* ***então***
incremento = (t_2 - t_1)/2.0;
t = t + incremento;
Fim_Se

senão
t_1 = t;
*incremento = incremento*2;*
t = t + incremento;
Se *t ³ t_2* ***então***
t = t - incremento;
incremento = (t_2 - t_1)/2.0;
t = t + incremento;

Fim_Se

Fim_Se
Até *(condições de parada).*

O valor da inclinação ω^{λ} *para* cada um dos tres casos é o seguinte:

1) para quando a restrição relaxada for a das capacidades

$$\omega^{\lambda} = \left(\sum_{i=1}^{m} b_i.\lambda_i - \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_i.w_{ij}.xij \right)$$

2) para quando a restrição relaxada for a da atribuição

$$\omega^{\lambda} = \left(\sum_{j=1}^{n} \lambda_j - \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_i.x_{ij} \right)$$

3) para quando a restrição relaxada for a proposta por Jornsten e Nasberg

$$\omega^{\lambda} = \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_{ij}.(y_{ij} - x_{ij})$$

4.4 -Heurística usada para fornecer solução viável ao (PGA)

Em Lorena e Narciso [48] foi descrita uma heurística que era usada para se resolver o *(PGA)* e fornecer uma solução viável. Esta heurística, na média, com instâncias conhecidas como classe C, as quais têm solução ótima conhecida, forneceu resultados muito próximos da solução ótima para cada instância considerada. O maior desvio em relação à solução ótima foi de 0.5%. Para as classes A e B, esta heurística praticamente obteve todas as soluções ótimas. A idéia desta heurística é, partindo-se da solução obtida através de uma relaxação, tentar torná-la viável através de uma série de critérios. Esta heurística foi melhorada (adicionando mais uma etapa, a qual está descrita no algoritmo a seguir como etapa 5) e usada para se obter a solução viável para cada instância do *(PGA)*.

Para cada restrição relaxada, a heurística tem um determinado critério. Considerando-se a relaxação das capacidades, tem-se as seguintes etapas do algoritmo

1 . Verificar quais foram as restrições de capacidade que não foram atendidas;

2 . Nas restrições de capacidades não atendidas pela solução fornecida pela relaxação retirar, em ordem crescente, p_{ij} tal que x_{ij} seja igual a 1 até que as restrições de capacidades voltem a ser atendidas. Após isto, algumas restrições de atribuição ficam sem ser atendidas.

3 . Inserir, em cada coluna j, na qual a restrição de atribuição não seja atendida, o maior valor de p_{ij} possível tal que a restrição das capacidades não sejam violadas;

4 . Caso consiga viabilizar todas as restrições de atribuição sem violar as restrições de capacidade, então melhorar a solução conforme a proposta de Martello e Toth [50] modificada (vide em Lorena e Narciso [48]).

5 . Se não conseguir viabilizar, repetir os critérios de 1 a 4 com alteração no critério 2. A alteração seria, nas restrições de capacidades não atendidas, retirar, em ordem crescente de p_{ij}/w_{ij} tal que $x_{ij} = 1$, p_{ij} até que as restrições de capacidade sejam atendidas;

6 . Se não conseguir viabilizar, repetir os critérios de 1 a 4 com alteração no critério 2. A alteração seria, nas restrições de capacidades não atendidas,

retirar, em ordem decrescente de w_{ij} tal que $x_{ij} = 1$, p_{ij} até que as restrições de capacidade sejam atendidas;

7 . Se não conseguir viabilizar, então usar uma heurística qualquer para obter a solução viável tal que a solução seja função do multiplicador λ.

O algoritmo para os primeiros 4 critérios acima, quando a restrição relaxada é a das capacidades, é o seguinte:

Algoritmo solução viável para o PGA

Fazer $z_{ij} := x^{l}_{ij}$; $i = 1,....,m$ *(z recebe a solução* x^{l} *da relaxação)*
  $knap_i := b_i - w_{ij}.x_{ij}$; $j = 1,....,n$
Repita para $i = 1, 2,....., m$
  Se $knap_i < 0$ ***então***
    $j^* :=$ *índice de min* $\{ p_{ij} \mid z_{ij} = 1 \}$
    $K := \{ k \mid (knap_q - w_{ij*}) \geq 0;\ q = 1, 2, ..., m;\ q \neq i \}$
    Se $K \neq \emptyset$ ***então***
      $i^* :=$ *índice de max* $\{ p_{kj*} \mid k \in K \}$;
      $z_{ij*} := 0$;
      $z_{i*j*} := 1$;
      $knap_i := knap_i + w_{ij*}$;
      $knap_{i*} := knap_{i*} - w_{i*j*}$;
    fim_Se
  fim_Se
fim_Repita

Repita para $i = 1, 2,....., m$
  Se $knap_i < 0$ ***então*** *vá para o passo 5;*
  { Os 4 primeiros passos do algoritmo não encontraram uma solução}
  fim_Se
fim_Repita
Melhorar *a solução viável com a Segunda parte do algoritmo*
    Proposto por Martello e Toth modificado
    (vide Lorena e Narciso[48] e Narciso[52])
Parar

Quando a restrição relaxada for a da atribuição, o procedimento é análogo. As etapas do algoritmo são as seguintes:

1 . Verificar quais foram as restrições de atribuição que não foram atendidas;

2 . Nas restrições de atribuição não atendidas pela solução fornecida pela relaxação, tem-se duas situações: $\sum x_{ij} = 0$ ou $\sum x_{ij} > 1$ para $i \in M$ e um dado $j^* \in N$.

Se $\sum x_{ij^*} > 1$, retirar, em cada coluna j^* e em ordem crescente, p_{ij^*} tal que x_{ij^*} seja igual a 1 até que a restrição volte a ser atendida ($\sum x_{ij^*} = 1$). Cada vez que p_{ij^*} é retirado, x_{ij^*} muda o valor de 1 para 0.

Se $\sum x_{ij^*} = 0$, então inserir, em ordem decrescente, um valor de p_{ij^*} tal que as restrições de capacidades não sejam violadas e, se houver sucesso, mudar o valor de x_{ij^*} de 0 para 1. Após verificar todas as restrições de atribuição não atendidas e tentar atendê-las, verificar se todas as restrições de atribuição e das capacidades foram atendidas, e então melhorar a solução obtida usando a heurística de Martello e Toth modificada.

Após isto, parar. Se alguma restrição não foi atendida, então ir para o passo 3.

3 . Repetir o passo 2, alterando p_{ij^*} para p_{ij^*}/w_{ij^*}. Desta forma, temos que:

Se $\sum x_{ij^*} > 1$, verificar, em cada coluna j^* e em ordem crescente de p_{ij^*}/w_{ij^*}, o valor de p_{ij^*} tal que x_{ij^*} seja igual a 1. Se $x_{ij^*} = 1$, retirar p_{ij^*} (e consequentemente, fazer $x_{ij} = 0$) até que a restrição volte a ser atendida ($\sum x_{ij^*} = 1$).

Se $\sum x_{ij^*} = 0$, então inserir, em ordem decrescente de p_{ij^*}/w_{ij^*}, um valor de p_{ij^*} tal que as restrições de capacidades não sejam violadas e, se

houver sucesso, mudar o valor de x_{ij^*} de 0 para 1. Após verificar todas as restrições de atribuição não atendidas e tentar atendê-las, verificar se todas as restrições de atribuição e das capacidades foram atendidas, e então melhorar a solução obtida usando a heurística de Martello e Toth modificada. Após isto, parar. Se alguma restrição não foi atendida, então ir para o passo 4.

4 . Repetir o passo 2, alterando p_{ij^*} para w_{ij^*}. Desta forma, temos que:

Se $\sum x_{ij^*} > 1$, verificar, em cada coluna j* e em ordem decrescente de w_{ij^*}, o valor de p_{ij^*} tal que x_{ij^*} seja igual a 1. Se $x_{ij^*} = 1$, retirar p_{ij^*} (e consequentemente, fazer $x_{ij} = 0$) até que a restrição volte a ser atendida ($\sum x_{ij^*} = 1$).

Se $\sum x_{ij^*} = 0$, então inserir, em ordem crescente de w_{ij^*}, um valor de p_{ij^*} tal que as restrições de capacidades não sejam violadas e, se houver sucesso, mudar o valor de x_{ij^*} de 0 para 1. Após verificar todas as restrições de atribuição não atendidas e tentar atendê-las, verificar se todas as restrições de atribuição e das capacidades foram atendidas, e então melhorar a solução obtida usando a heurística de Martello e Toth modificada. Após isto, parar. Se alguma restrição não foi atendida, então ir para o passo 5.

5 . Se não conseguir viabilizar, então usar uma heurística qualquer para obter a solução viável tal que a solução seja função do multiplicador λ.

Para a relaxação proposta por Jornsten e Nasberg, para se obter a solução viável a partir da solução fornecida pela relaxação, foi usada a heurística que leva em conta a solução fornecida pela relaxação da atribuição pois esta tem um valor mais próximo da solução ótima ($v(L\lambda aPGA) \leq v(L\lambda cPGA)$, conforme visto anteriormente).

4.5 -Resultados

Para verificar o comportamento das relaxações *lagsur* e lagrangeana com o *(PGA)* foram usadas instâncias obtidas da OR-Library [8]. Os resultados obtidos estão dispostos, nesta seção, em duas tabelas. A primeira tabela contém instâncias consideradas de pequena e média escala. A Segunda tabela contém instâncias consideradas larga escala. A seguir, estão descritas as duas tabelas com a média dos resultados obtidos.

A tabela 1 apresenta os resultados computacionais usando um conjunto de 12 tipos diferentes de problemas. Cada tipo de problemas possui uma dimensão m x n e 5 instâncias diferentes. Os valores de m x n são: (5 x 15), (5 x 20), (5 x 25), (5 x 30), (8 x 24), (8 x 32), (8 x 40), (8 x 48), (10 x 30), (10 x 40), (10 x 50) e (10 x 60). Apenas a classe C de problemas da OR-Library foi usada por ter todas as soluções ótimas conhecidas e os resultados estão dispostos como uma média, para cada problema, das 5 instâncias diferentes. Os resultados computacionais foram obtidos usando-se um PC486DX2 e todos os algoritmos foram codificados em linguagem C. A notação em cada coluna é a seguinte:

tempo := tempo final, em segundos, obtido pela heurística;
iter. := número de vezes que a relaxação lagrangeana foi resolvida;
gap1 := ((solução ótima) - lb) / (solução ótima) ;
gap2 := (ub - (solução ótima)) / (solução ótima) ;

gap2 = 5%,...,0.5% := tempo computacional para o gap2 obter 5%, 4%, 3%, 2%, 1% and 0.5%.

Em gap1, lb significa limite inferior e em gap2, ub significa limite superior, tal como definido no algoritmo de subgradientes, versão II. Quando as porcentagens do gap2 são alcançadas, elas refletem o desempenho da relaxação sem a influência dos testes de parada. Os números entre parênteses representam o número de problemas que a correspondente relaxação alcançou a dada porcentagem de gap2.

Vale a pena mencionar que, para a *relaxação lagsur,* o melhor valor de t foi calculado até que fosse repetido em 5 iterações seguidas. A partir daí foi fixado nesse último valor. Para a relaxação lagrangeana, t foi fixado para 1 em todas as iterações.

Relaxação	tempo (seg.)	iter.	gap1 $x10^{-3}$	gap2 $x10^{-3}$	gap2 = 5%	gap2 = 4%	gap2 = 3%	gap2 = 2%	gap2 = 1%	gap2 =0.5%
L_1S_λcPGA	1.99	300	3.87	11.16	0.10 (12)	0.11 (12)	0.14 (12)	0.16 (10)	0.50 (7)	0.89 (1)
L_tS_λcPGA	1.12	196	5.28	11.24	0.09 (12)	0.09 (12)	0.10 (12)	0.10 (10)	0.16 (7)	0.26 (1)
L_1S_λaPGA	12.32	192	0.84	1.76	2.59 (12)	2.81 (12)	3.09 (12)	3.48 (12)	4.30 (12)	6.09 (12)
L_tS_λaPGA	15.18	248	0.90	1.82	2.24 (12)	2.38 (12)	2.61 (12)	3.31 (12)	7.52 (12)	8.65 (12)
L_1S_λJN	46.45	444	0.95	4.76	4.12 (12)	6.18 (12)	9.72 (12)	16.76 (12)	21.32 (11)	24.29 (8)
L_tS_λJN	28.62	370	0.37	2.38	2.11 (12)	2.98 (12)	4.76 (12)	9.60 (12)	10.58 (12)	10.60 (12)

Tabela 4.1 - resultados computacionais para instâncias pequenas (classe C de problemas).

A tabela 4.2, a seguir, mostra a média dos resultados obtidos para os problemas considerados de larga escala. Foram utilizados 24 tipos de problemas com dimensões diferentes. Estes problemas são conhecidos como problemas de classes A, B, C, e D. As dimensões de cada um destes problemas são: (5 x 100), (5 x 200), (10 x 100), (10 x 200), (20 x 100) e (20 x 200). Os números em cada campo da tabela 4.2 representam os valores médios obtido. Estes testes foram executados em uma máquina SUN SPARC 5, com compilador C.

Relaxação	tempo (seg.)	iter.	gap1 $x10^{-3}$	gap2 $x10^{-3}$	gap2 = 5%	gap2 = 4%	gap2 = 3%	gap2 = 2%	gap2 = 1%	gap2 =0.5%
$L_1S\lambda$cPGA	9.15 (22)	232	1.58	2.23 (22)	1.50 (22)	1.55 (22)	1.60 (22)	1.66 (22)	1.83 (21)	2.04 (19)
$L_tS\lambda$cPGA	2.60 (24)	148	2.48	2.45 (24)	0.80 (24)	0.80 (24)	0.80 (24)	0.80 (24)	0.86 (23)	0.88 (21)
$L_1S\lambda$aPGA	184.54 (17)	534	0.44	2.24 (17)	35.88 (17)	37.6 (17)	39.63 (17)	44.64 (16)	48.55 (16)	52.37 (16)
$L_tS\lambda$aPGA	115.83 (23)	433	0.30	1.00 (23)	14.15 (23)	14.76 (23)	15.51 (23)	17.43 (23)	19.20 (23)	22.53 (22)
$L_1S\lambda$JN	248.44 (13)	600	2.50	16.12 (13)	35.95 (13)	42.72 (13)	24.57 (9)	30.60 (8)	23.59 (5)	19.52 (4)
$L_tS\lambda$JN	278.83 (21)	541	1.18	4.94 (21)	9.80 (21)	11.41 (20)	8.06 (19)	9.90 (18)	8.39 (15)	9.61 (13)

Tabela 4.2 - Resultados computacionais para problemas de larga escala.

Os resultados acima, para problemas de larga escala, foram obtidos com um limite de 600 iterações. Para computar o gap1 e o gap2 era necessário saber o valor da solução

ótima de cada instância. Entretanto, para as instâncias usadas, não se conheciam as soluções ótimas. As soluções ótimas foram determinadas antes de se executar os testes. Para isso, cada instância foi usada nas duas relaxações (*lagsur e lagrangeana*) para cada uma das 3 propostas descritas anteriormente (Jornsten e Nasberg, relaxação da restrição de atribuição e relaxação da restrição das capacidades), totalizando 6 testes com a mesma instância. Para cada instância, usou-se o critério de parada *(ub - lb)* < 1 e um número máximo de iterações igual a 2000. Quando se obtia *(ub - lb)* < 1, então a solução ótima era obtida. Quando não se obtia *(ub - lb)* < 1, era usada a melhor solução viável encontrada para os 6 testes com a instância. As soluções obtidas para cada instância estão abaixo e os valores marcados com (*) são soluções ótimas.

A5x100 ® 4456 (*)	*A5x200 ® 8788 (*)*	*A10x100 ® 4700 (*)*
A10x200 ® 9413 (*)	*A20x100 ® 4857 (*)*	*A20x200 ® 9666 (*)*
B5x100 ® 4008	*B5x200 ® 8502*	*B10x100 ® 4633 (*)*
B10x200 ® 9255	*B20x100 ® 4817 (*)*	*B20x200 ® 9670*
C5x100 ® 4411 (*)	*C5x200 ® 8347*	*C10x100 ® 4528*
C10x200 ® 9247	*C20x100 ® 4784*	*C20x200 ® 9611*
D5x100 ® 9147 (*)	*D5x200 ® 18750 (*)*	*D10x100 ® 10349 (*)*
D10x200 ® 20562 (*)	*D20x100 ® 10839 (*)*	*D20x200 ® 21733 (*)*

A seguir, para ilustrar o comportamento do dual das relaxações lagrangeana e *lagsur*, relaxando-se as restrições das capacidades, atribuição e a da proposta de Jornsten e Nasberg, tem-se os gráficos abaixo. A instância usada para as três diferentes formas de relaxação foi a de classe C, com dimensões 5 x 100.

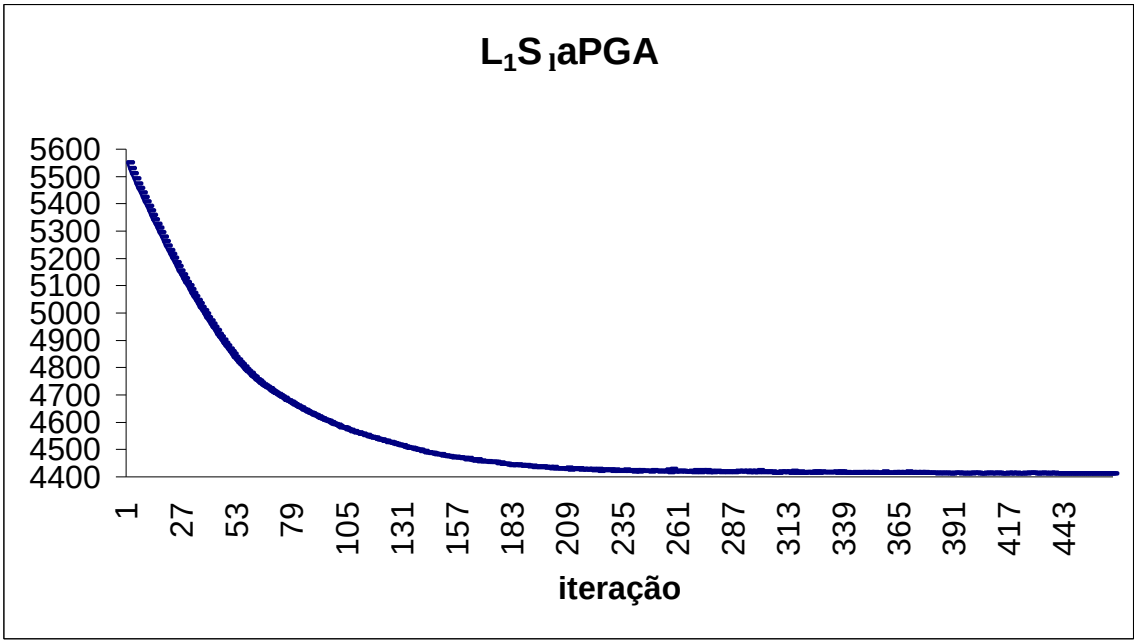

Figura 4.2 - Gráfico $L_1S_\lambda aPGA$ versus iteração

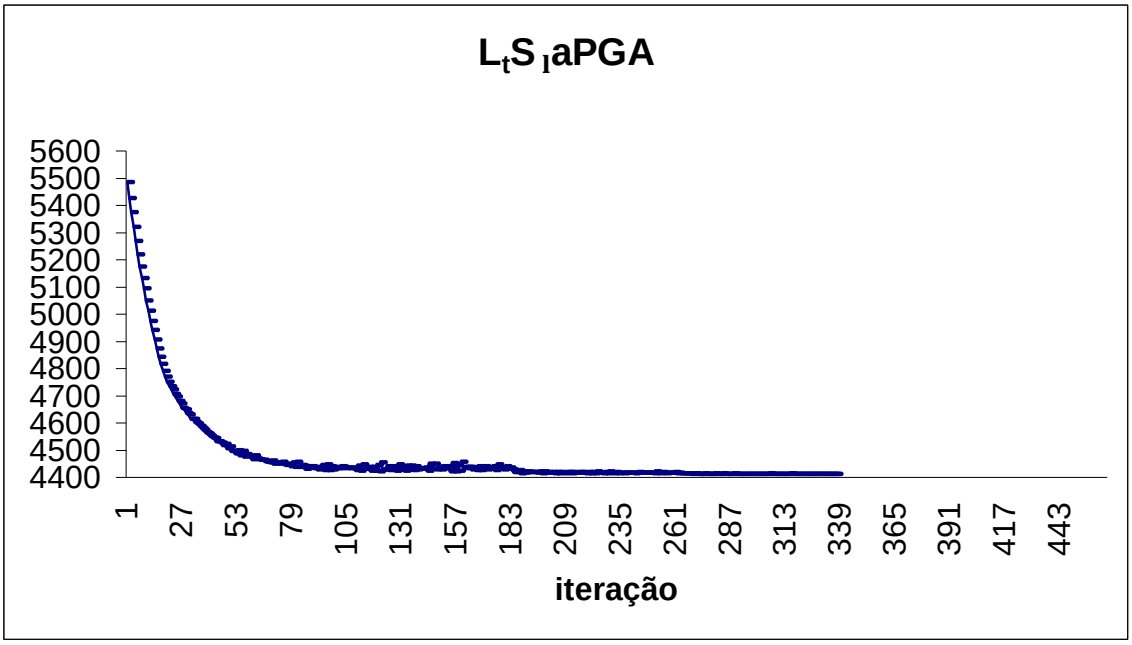

Figura 4.3 - Gráfico $L_tS_\lambda aPGA$ versus iteração

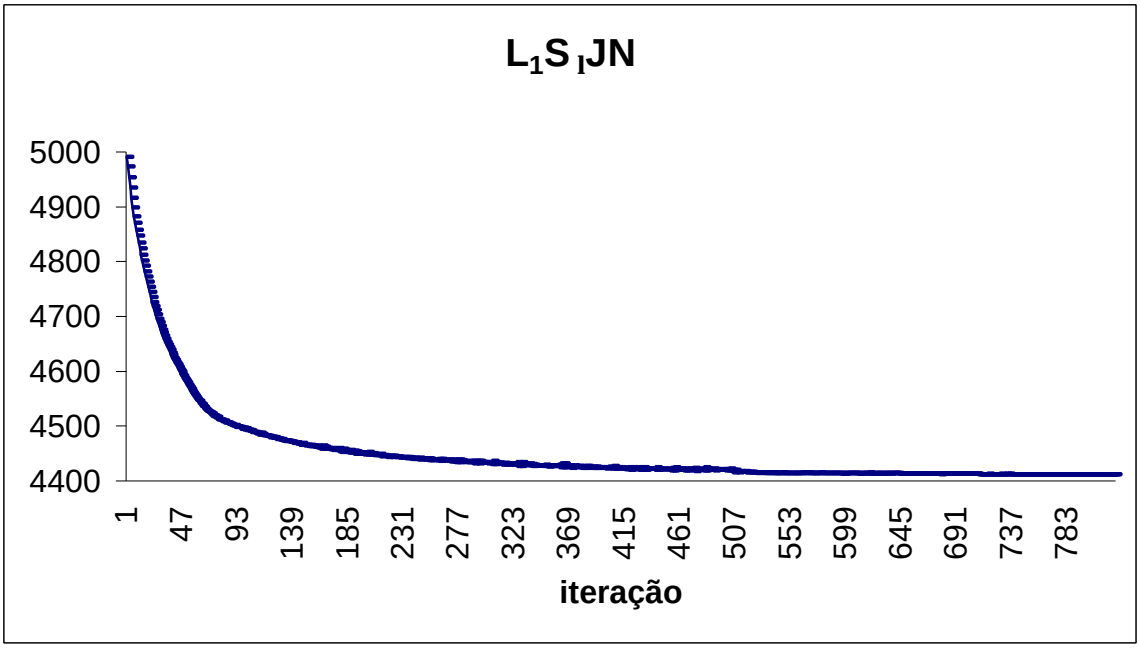

Figura 4.4 - Gráfico $L_1S_\lambda JN$ versus iteração

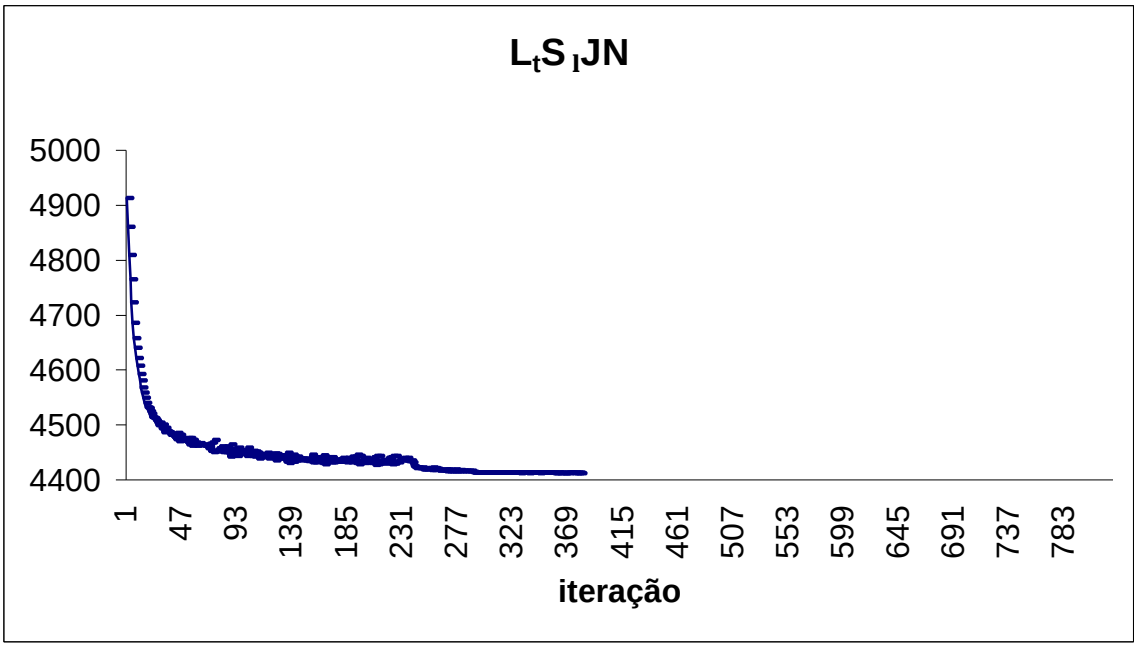

Figura 4.5 - Gráfico $L_tS_\lambda JN$ versus iteração

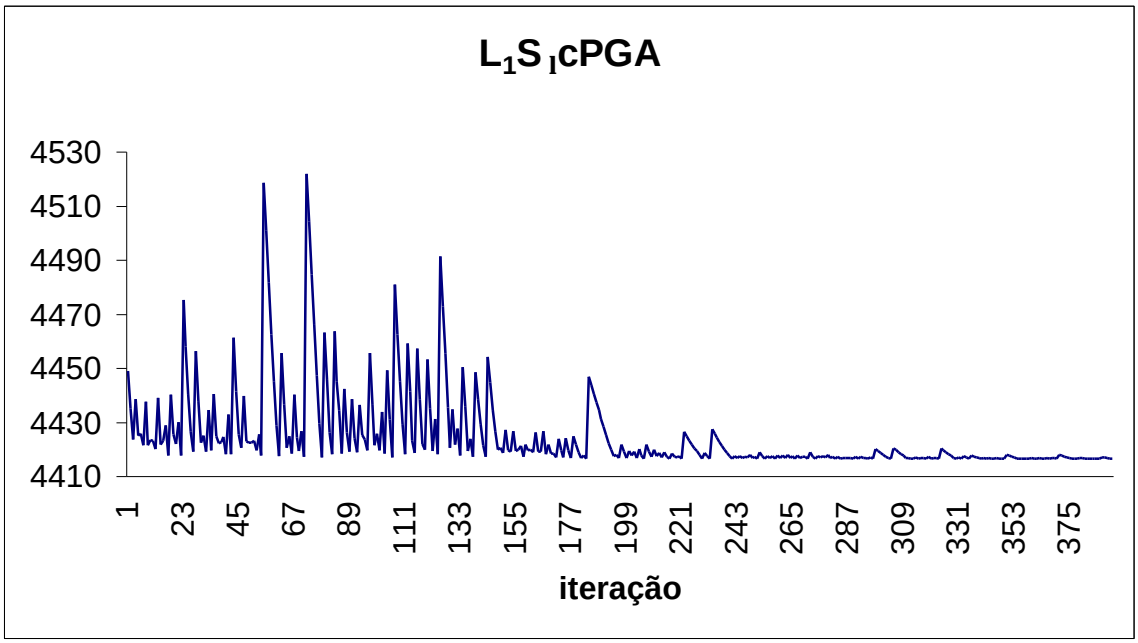

Figura 4.6 - Gráfico L_1S_λcPGA versus iteração

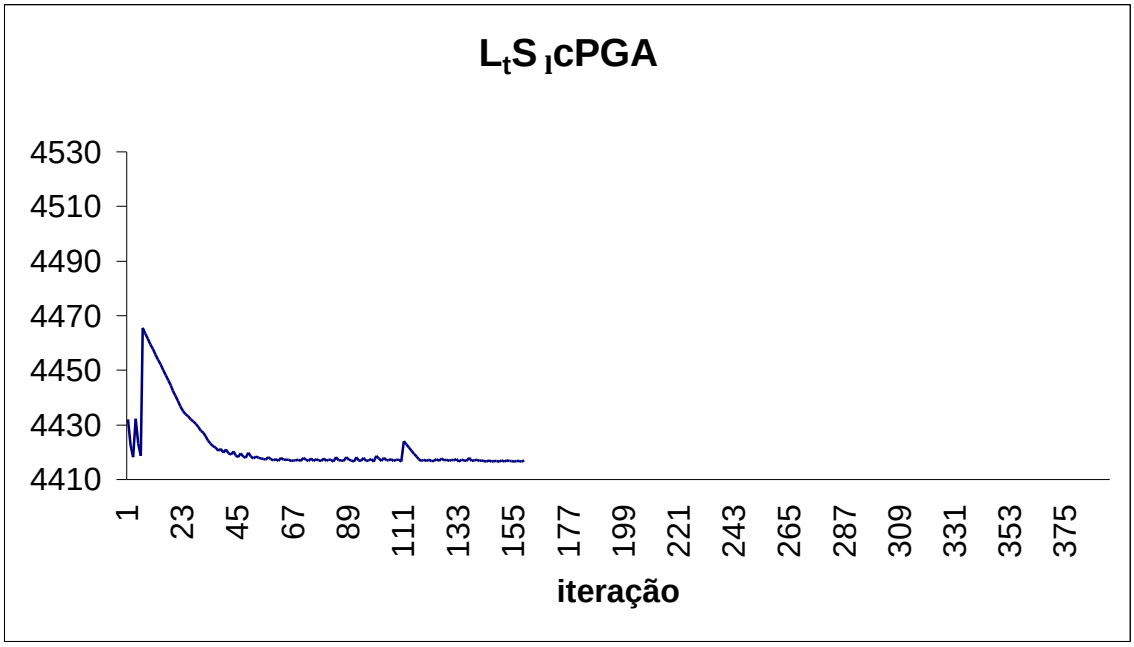

Figura 4.7 - Gráfico L_tS_λcPGA versus iteração

4.6 - Comentários sobre os resultados obtidos para o (PGA)

Em relação a 600 iterações, a relaxação L_tS_λaPGA obteve melhores resultados do que as outras duas (L_tS_λcPGA e L_tS_λJN) para gap1 e gap2. Entretanto, se o limite máximo de iterações fosse maior, a relaxação L_tS_λJN obteria melhores resultados. Isto pode ser visto no *Apêndice A*, para as tabelas contruídas para o L_tS_λJN para 1200 iterações. Nestas tabelas, nota-se claramente que a relaxação L_tS_λJN obtém um valor melhor para gap1 e gap2. A desvantagem desta relaxação proposta por Jornsten e Nasberg é a convergência lenta. Observou-se, portanto, que, conforme a restrição a ser relaxada, os resultados podem mudar significativamente.

Os testes com instâncias maiores refletem de uma forma mais intensa o ganho em tempo computacional da relaxação *lagsur ((L_tS_laPGA), (L_tS_lcPGA)* e *(L_tS_lJN))* em relação à relaxação lagrangeana *(*L_1S_λaPGA, L_1S_λcPGA e L_1S_λJN*)*. Isto também ocorreu com os valores dos limites obtidos, principalmente com os problemas mais difíceis (classes C e D).

Para os problemas da classe D, considerados mais difíceis, a relaxação *lagsur* mostrou ser mais eficaz para se encontrar limites superiores do que a relaxação lagrangeana.

A relaxação *lagsur*, dentro de um mesmo número máximo de iterações, encontrou mais valores do que a relaxação lagrangeana para todos os testes realizados para os problemas maiores (classes A, B, C e D) e em um tempo menor, na média. Isto pode ser comprovado mais claramente nos resultados dispostos no *Apêndice A*.

Nos gráficos apresentados, verifica-se que, para as relaxações *(*L_1S_λaPGA*)* e *(*L_1S_λJN*)*, as sequências de limites lagrangeanos oscilam pouco e ainda menos que as correspondentes sequências do *lagsur* . Portanto, neste caso, o modo de emprego do *lagsur* não estabilizou suas sequências de limites, refletindo o fato de que a busca unidimensional para o melhor t foi realizada em apenas algumas iterações (iniciais) do algoritmo de subgradientes, versão II. Mesmo assim, o uso da informação local do *lagsur* foi suficiente para o ganho computacional obtido.

No caso da relaxação *(*L_1S_lcPGA*)*, verificou-se grande oscilação na sequência de valores lagrangeanos. Nesse caso, a relaxação *lagsur (*L_tS_lcPGA*)* atenuou um pouco esta oscilação e também um obteve um ganho computacional significativo.

O algoritmo empregado para a busca unidimensional do melhor t poderia ser mais elaborado, mas o objetivo de ganho de tempo computacional seria prejudicado neste caso, pois um algoritmo elaborado exigiria mais avaliações de v*(*L_tS_lPGA*)*. Quando o melhor t repetiu e foi fixado nas iterações seguintes, o efeito da busca local se propagou nas próximas iterações e se mostrou benéfico em relação à convergência do *lagsur*.

CAPÍTULO 5

5 - Problema do Caixeiro Viajante Simétrico

5.1. Introdução

Suponha que um caixeiro, a partir de sua cidade de origem, precise visitar um conjunto de n - 1 cidades. Cada cidade deverá ser visitada somente uma vez. Após visitar todas as n - 1 cidades, então o caixeiro deverá voltar a sua cidade inicial. O caixeiro poderá escolher a primeira cidade a ser visitada, a segunda, e assim por diante. Entretanto, caso o caixeiro queira economizar tempo e energia (ou ainda gasolina, recursos, etc.), então ele deve procurar o menor caminho tal que todas as cidades sejam visitadas somente uma vez e então retornar para a sua cidade de origem. Se o caixeiro sabe a distância entre as cidades, então ele pode encontrar o caminho mais curto para a sua viagem simplesmente por comparar todas as maneiras possíveis de se visitar as n - 1 cidades e voltar. Entretanto, admitindo-se que exista sempre um caminho ligando uma cidade a outra, à medida em que o número de cidades vai crescendo, o número de maneiras de se visitar todas as cidades cresce também. O problema de se determinar o caminho mais curto tal que todas as cidades sejam visitadas uma vez é conhecido como *Problema do Caixeiro Viajante (PCV)* e é considerado um problema clássico da Otimização Combinatória.

Se um caixeiro deve visitar n cidades, contando com a cidade de origem, então o número de maneiras de se visitar todas as n cidades é dado por (n-1)!. Isto pode ser visto da seguinte maneira. Se o caixeiro está na cidade 1, então para ir à cidade 2, exitem (n-1) maneiras, pois restam (n-1) cidades. Para ir da cidade 2 para a cidade 3, existem (n-2) maneiras, visto que restam (n-2) cidades. Para ir da cidade 3 para a cidade 4 existem (n-3) maneiras. O raciocínio prossegue até que, para ir da cidade (n-1) para a n-ésima cidade exita apenas um caminho. Desta forma, o número de maneiras de visitar todas as cidade apenas uma vez é dado por (n-1).(n-2).(n-3)....1 = (n-1)!. Desta forma, se o número de

cidades a serem visitadas for 5, então teremos 24 possibilidades (4! possibilidades) de visitar todas as cidades. Se, entretanto, o número de cidades for 1001, o número de possibilidades será dado por 1000!, o que é muito elevado, tornando-se inviável contabilizar todas as possibilidades. Com estas considerações, para se encontrar o menor caminho para visitar todas as cidades, deve ser feito um procedimento tal que não se necessite conferir todos os caminhos possíveis, a menos que o número de cidades seja 3, 4, ou 5, por exemplo.

O *(PCV)* é NP-Completo (Noschang [54], Lenstra et al [45]). Desta forma, é pouco provável que qualquer algoritmo, por mais eficiente que seja, encontre um caminho ótimo quando o número de cidades for grande e, além disso, a execução seja rápida. Embora seja difícil encontrar soluções ótimas para o *(PCV)*, quando o número de cidades é muito grande, com o uso de heurísticas (métodos não exatos), é possível encontrar soluções viáveis que sejam muito próximas do ótimo. Além de se poder encontrar soluções viáveis próximas da solução ótima, é possível encontrar também uma solução não viável que forneça um limite inferior à solução ótima do *(PCV)* em questão. Isto pode ser feito através de uma relaxação. Desta forma, a relaxação seria um limitante inferior e a solução viável dada por uma heurística seria um limitante superior da solução ótima do problema. Com estes limites, podemos então obter a solução ótima usando métodos exatos.

Uma das relaxações usada com sucesso para se encontrar o limite inferior da solução ótima para o *(PCV)* foi a proposta de Held e Karp [33,34]. Em 1970 e 1971, Held e Karp publicaram trabalhos nos quais experimentaram a relaxação lagrangeana para fornecer limites inferiores para a solução ótima de problemas do caixeiro viajante. Estes limites obtidos eram muito próximos do ótimo. Christofides [13], para um conjunto de problemas gerados aleatoriamente, conseguiu valores de 99% da solução ótima, na média, usando a proposta de Held e Karp. Volgenant & Jonker [63], com uma outra série de problemas, conseguiram valores em torno de 99.7% da solução ótima, na média. Segundo Johnson [37], usando a proposta de Held e Karp para problemas conhecidos e disponibilizados na Internet (TSPLIB), a relaxação lagrangeana pode alcançar resultados

variando de 90% (pior caso encontrado) a praticamente 100% (para problemas menores, em geral). Para problemas considerados grandes (acima de 500 cidades), os valores fornecidos pela relaxação lagrangeana eram muito próximos de 99%. Desta forma, embora a proposta de Held e Karp seja de 1971, ainda hoje é empregada com sucesso pois os valores obtidos são muito bons.

Neste capítulo, será enfocada a relaxação lagrangeana, conforme a proposta de Held e Karp, para fornecer o limite inferior da solução ótima para uma série de problemas e também a relaxação *lagsur* para o caixeiro viajante, a qual também usará a proposta de Held e Karp. Os resultados das duas relaxações serão mostrados e comparados. As instâncias usadas para comparar as duas relaxações foram obtidos através de um site na Internet (http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95), o qual disponibiliza os dados relativos ao problema caixeiro viajante simétrico, bem como as soluções ótimas já encontradas ou as melhores soluções viáveis encontradas.

5.2 - Formulação do Problema do Caixeiro Viajante

Seja um grafo não direcionado G=(V,E) com n vértices, onde V={1,2,...,n} e o conjunto de arcos E = {(i,j) tal que i,j $\in$ V e seja c_{ij} o custo não negativo associado ao arco i-j. O Problema do caixeiro viajante pode ser definido como

$$v(PCV) = min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

$$\text{sujeito a} \quad \sum_{j \in V} x_{ij} = 1 \quad i \in V \qquad (5.1)$$

$$(PCV) \qquad \sum_{i \in V} x_{ij} = 1 \quad j \in V \qquad (5.2)$$

$$\sum_{i \in V} \sum_{j \in V} x_{ij} \leq |S| - 1, \quad \forall S \subset V, \; S \neq \emptyset \qquad (5.3)$$

$$x_{ij} = 0 \text{ ou } 1, \quad i,j \in V, \qquad (5.4)$$

onde $x_{ij} = 1$ se o arco i-j pertence a solução e $x_{ij} = 0$, se o arco i-j não pertence a solução.

A função objetivo do problema indica a minimização das distâncias ao se visitar todas as cidades e voltar a cidade de origem. A restrição (5.1) indica que, a partir da cidade i, deve-se ir a apenas uma cidade j, $j \in V\text{-}\{i\}$. Analogamente, a restrição (5.2) indica que, a partir de uma cidade j, deve-se ir apenas a uma cidade i, $i \in V - \{j\}$. A restrição (5.3) serve para eliminar ciclos dentro de todo o trajeto. Desta forma, se existe um conjunto S com k cidades, $k < n$, estas k cidades não serão interligadas de forma a existir um ciclo entre elas. O único ciclo que pode existir é o trajeto começando de uma cidade inicial, percorrendo as outras n-1 cidades e voltando à cidade inicial. Esta restrição também pode ser escrita como

$$\sum_{i \in S} \sum_{j \in V-S} x_{ij} \geq 1, \; \forall S \subset V, \; S \neq \emptyset \qquad (5.5)$$

A restrição (5.4) indica que, para xij = 1, o arco que liga i a j é escolhido para fazer parte da solução. Se o valor for 0, então o arco que liga i a j não é escolhido para o trajeto.

O *(PCV)* é *simétrico* quando $c_{ij} = c_{ji}$, para todo $i,j \in V$. Sendo assim, o *(PCV)* simétrico pode ser definido como um grafo não direcionado completo G= (V,E) com n vértices, V={1,2,...,n}, onde E é o conjunto de todos os arcos conhecidos de i a j ou j a i, e $c_{ij} = c_{ji}$. Desta forma, o *(PCV)* simétrico, doravante chamado de *(PCVS)*, pode ser definido como

$$v(PCVS) = min \quad \sum_{i \in V} \sum_{j>i} c_{ij}x_{ij}$$

(PCVS)

$$\text{sujeito a} \quad \sum_{j<i} x_{ij} + \sum_{j>i} x_{ij} = 2 \qquad i \in V \qquad (5.6)$$

$$\sum_{i \in V} \sum_{\substack{j \in S \\ j>i}} x_{ij} \leq |S| - 1, \ \forall\, S \subset V, S \neq \varnothing \qquad (5.7)$$

$$x_{ij} = 0 \text{ ou } 1, \quad i,j \in V, j>i. \qquad (5.8)$$

A expressão (5.7) pode ser também escrita como

$$\sum_{i \in V} \sum_{\substack{j \in S \\ j>i}} x_{ij} + \sum_{i \in V} \sum_{\substack{j \in S \\ j>i}} x_{ij} \geq 2, \ \forall\ S \subset V, S \neq \varnothing \qquad (5.9)$$

5.3 - A relaxação lagrangeana aplicada ao (PCVS)

Esta relaxação usa a proposta de Held e Karp [33,34]. Para um melhor entendimento da relaxação lagrangeana aplicada ao *(PCVS)*, temos que fazer algumas considerações, as quais estão descritas a seguir.

O problema de obter um subgrafo conectado H de G=(V,E) com n arcos, onde o grau (número de arcos que estão ligados a um determinado vértice) de cada um dos n vértices seja maior ou igual a 1, tal que minimize a função objetivo, é obviamente uma relaxação do *(PCVS)*. O subgrafo H consiste de uma árvore obtida a partir de G em adição com um caminho extra. Pode-se restringir H para uma classe Q de subgrafos na qual alguns vértices arbitrários de G, digamos vertice 1, tenha grau 2 e está contido em um único ciclo de H. Os subgrafos desta classe Q são chamados de 1-tree. Desta forma, a 1-tree de custo mínimo *(M1-tree)* seria formulada matematicamente como

$$v(M1\text{-}tree) = min \quad \sum_{i \in V} \sum_{j>i} c_{ij}x_{ij}$$

$$\text{sujeito a} \quad \sum_{\substack{i \in S \\ j>i}} \sum_{j \in V'-S} x_{ij} + \sum_{i \in V'-S} \sum_{\substack{j \in S \\ j>i}} x_{ij} \geq 1$$

(M1-tree)

$$\forall \quad S \subset V' = V - \{1\}, S \neq \varnothing \qquad (5.10)$$

$$\sum_{i \in V} \sum_{j>i} x_{ij} = n \qquad (5.11)$$

$$\sum_{j \in V} x_{1j} = 2 \qquad (5.12)$$

(5.8)

Para provar que se encontrarmos uma 1-tree que minimiza a função objetivo então esta é uma relaxação do *(PCVS)*, basta perceber que o conjunto de restrições que define a família Q são a função objetivo acima e as expressões (5.8), (5.10), (5.11), e (5.12). A expressão (5.10) é proveniente da expressão (5.9). A expressão (5.11) é proveniente da soma de todas as equações (5.6) dividida por 2. A expressão (5.12) é primeira equação de (5.6).

O problema da 1-tree de custo mínimo pode ser visto como uma decomposição em dois problemas independentes:

(1) encontrar uma árvore geradora de custo mínimo em V - {1}; e

(2) encontrar os dois menores arcos dentre aqueles que vão do vértice 1 a V-{1}.

Os n-2 arcos da árvore geradora encontrados no primeiro problema (1), juntamente com os dois outros arcos encontrados no problema (2), formam a árvore 1-tree de custo mínimo em G.

Para se encontrar a árvore geradora mínima, pode ser usado o algoritmo de Kruskal [36], tal como descrito a seguir.

5.3.1 - Algoritmo de Kruskal para gerar a árvore mínima

Para se obter a árvore gerador de custo mínimo, uma das possibilidades seria o emprego do algoritmo de Kruskal. Para um melhor entendimento do algoritmo, vamos definir algumas variáveis. Seja E o conjunto de todos os arcos, tal como descrito anteriormente. Cada arco liga um vértice i a um vértice j. A caminho de i até j é o mesmo de j a i. Seja D o conjunto dos arcos (i,j) que são escolhidos para comporem a árvore

mínima. Feitas estas duas considerações, temos então o algoritmo de Kruskal, segundo [36].

$D \leftarrow \phi$

Enquanto (D contém menos de n-1 arcos) e (o conjunto E for não vazio) **faça**

escolha um arco (i,j) do E, de menor custo;

suprima (i,j) do conjunto E;

Se (i,j) não cria um ciclo em D

então adicione (i,j) a D

senão suprima (i,j)

fim_enquanto

Se D contém menos de n-1 arcos

então Imprimir (´nenhuma arvore minima foi gerada´)

Vale a pena mencionar que o ponto crítico deste algoritmo é a ordenação necessária para que a instrução “escolha um arco (i,j) do E, de menor custo” seja executada. Antes do algoritmo de Kruskal ser executado, é feita uma ordenação de todos os custos dos arcos. Desta forma, ao começar o algoritmo, já existirá uma lista de arcos ordenados de forma crescente. Para o caso da implementação do algoritmo de Kruskal, o algoritmo de ordenação utilizado foi o Quicksort, amplamente conhecido na literatura (vide Sahni e Horowitz [36]) e também com $O(n\log(n))$ para pior caso.

5.3.2- Algoritmo para gerar a 1-tree de custo mínimo

Para se obter a 1-tree, podemos então usar o seguinte algoritmo

1 - Escolher um vértice pertencente a G. Seja este vértice denominado de vértice 1.

2 - Determinar uma árvore geradora mínima para os n-1 vértices restantes.

3 - Escolher as duas menores distancias que vão do vertice 1 a árvore geradora mínima e adicionar estes dois caminhos à árvore geradora mínima, formando assim a 1-tree.

5.3.3 - Proposta de Held e Karp - a relaxação lagrangeana aplicada ao PCVS

O problema de se obter o mínimo custo da 1-tree, cuja função objetivo é a mesma do *(PCVS)*, é uma relaxação do *(PCVS)*. Em Lenstra et al [45], temos que a solução ótima obtida pela 1-tree, em um conjunto de 140 problemas, é, em média, 63% do valor ótimo do *(PCVS)*. Entretanto, esta relaxação pode ser melhorada consideravelmente por se tomar a equação (5.6) e adicioná-la à função objetivo tal como na relaxação lagrangeana e então maximizar o lagrangeano como uma função de multiplicadores $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$, $\lambda \in \Re^n$. Desta forma, teríamos então a formulação matemática função objetivo da relaxação lagrangeana para o *(PCVS)* definida como

$$v(L_l PCVS) = \min\left\{ \sum_{i \in V} \sum_{j>i} c_{ij}x_{ij} + \sum_{i \in V} \lambda_i \left(\sum_{j<i} x_{ij} + \sum_{j>i} x_{ij} - 2 \right) \right\},$$

Esta função objetivo pode ser reescrita como:

$$v(L_l PCVs) = \min\left\{ \sum_{i \in V} \sum_{j>i} (c_{ij} + \lambda_i + \lambda_j)x_{ij} - 2.\sum_{i \in V} \lambda_i \right\}$$

Feitas estas considerações, temos então a relaxação lagrangeana definida por:

$$v(L_l PCVS) = \min \left\{ \sum_{i \in V} \sum_{j>i} (c_{ij} + \lambda_i + \lambda_j)x_{ij} - 2.\sum_{i \in V} \lambda_i \right\}$$

$(L_l PCVS)$ sujeito a (5.10), (5.11), (5.12) e (5.8)

Para se obter o valor de $v(L_l PCV)$, basta se obter a 1-tree, tal como foi descrito anteriormente, fazendo uma pequena modificação no valor de c_{ij}, isto é, o valor de c_{ij} deve ser sempre atualizado para $(c_{ij} + \lambda_i + \lambda_j)$. Ao se obter a 1-tree, determine o custo desta e subtraia do custo a parcela $2\sum \lambda_i$, onde i varia de 1 a n. Desta forma

$$v(L_l PCVS) = v(1\text{-}tree) - 2\sum \lambda_i,\ i = \{1,2,...,n\}.$$

Os arcos obtidos pelo algoritmo da 1-tree, desta forma, seriam os mesmos da relaxação lagrangeana, isto é, x_{ij} (obtido via 1-tree) = x_{ij} (lagrangeano)

O maior valor que a função objetivo de $v(L_l PCVS)$ pode obter é dado por

$$v(DL_l PCVS) = \max \{ v(L_l PCVS) \},\ \lambda \in \Re^n$$

Para resolver $v(DL_l PCVS)$ usa-se o algoritmo de subgradientes, versão II. Comece por um $\lambda=\lambda_0$ arbitrário. Em cada iteração k, atualize λ^k. Para atualizar λ^k, é necessário resolver $(L_l PCVS)$. Ao se resolver $(L_l PCVS)$, determine o vetor de subgradientes. Cada componente i deste vetor é dado por

$d_i^k - 2$, onde d_i^k é o grau do vértice i, $i \in V$. Desta forma,

$$\lambda_i^{k+1} = \lambda_i^k + p^k(d_i^k - 2)$$

onde p^k é o passo definido por $p^k = \alpha(U - v(L_l PCVS))/ \sum_{i \in V} (d_i^k - 2)^2$.

Para este cálculo de p^k, temos que. $0<\alpha\leq 2$, U é a solução viável obtida na iteração k (ou solução ótima do problema).

A seguir, tem-se o exemplo da aplicação da relaxação lagrangeana, retirada de [45] para se obter o limite inferior do *(PCVS)* do seguinte problema. Sejam 8 cidades pelas quais o caixeiro deverá passar. Seja $c_{ij} = c_{ji}$ o custo de se ir da cidade i para a cidade j. Como o problema é simétrico, então $c_{ij} = c_{ji}$ A matriz de custos está disposta da seguinte maneira:

$$c = \begin{vmatrix} 0 & 2 & 4 & 5 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 0 & 0 & 7 & 5 & 0 \\ 4 & 4 & 0 & 1 & 7 & 4 & 0 & 0 \\ 5 & 0 & 1 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 7 & 10 & 0 & 1 & 0 & 4 \\ 0 & 7 & 4 & 0 & 1 & 0 & 3 & 5 \\ 0 & 5 & 0 & 0 & 0 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 4 & 5 & 2 & 0 \end{vmatrix}$$

Seja U=25, α=2, λ_i^0=0, para i=1,...,8. A solução obtida para a 1-tree de custo mínimo, para os custos iniciais dados acima, é igual a 21 e a 1-tree é dada pelos seguintes arcos i-j:

8-7, 7-6, 6-5, 6-3, 3-2, 3-4, 1-3 e 1-2.

Visto que λ_i^0=0, então a solução dada pela 1-tree é igual ao valor da função objetivo do lagrangeano.

Além desta solução dada pelo 1-tree, temos: d_i^0=(2,2,4,1,1,3,2,1), p^0=2(25-21)/8 = 1,

λ_i^0=(0, 0, 2, -1, -1,1,0,-1).

Desta forma, na proxima iteração, os custos serão modificados para $c_{ij}^1 = (c_{ij}^0 + \lambda_i + \lambda_j)$.

A nova matriz de custos é dada por:

$$c = \begin{vmatrix} 0 & 2 & 6 & 4 & 0 & 0 & 0 & 0 \\ 2 & 0 & 6 & 0 & 0 & 8 & 5 & 0 \\ 6 & 2 & 0 & 2 & 8 & 7 & 0 & 0 \\ 4 & 0 & 2 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 8 & 8 & 0 & 1 & 0 & 2 \\ 0 & 8 & 7 & 0 & 1 & 0 & 4 & 5 \\ 0 & 5 & 0 & 0 & 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 5 & 1 & 0 \end{vmatrix}$$

A solução fornecida pelo algoritmo de cálculo da 1-tree de custo mínimo tem os seguintes arcos:

1-2, 2-7, 7-8, 8-5, 5-6, 6-3, 3-4, 4-1. Desta forma, visto que todos os vértices tem grau 2, então a solução é ótima. Entretanto, geralmente a solução obtida pela relaxação não é ótima, mas o valor da função objetivo da relaxação lagrangeana se aproxima consideravelmente do valor da solução ótima.

5.4 - A relaxação lagsur aplicada ao (PCVS)

A relaxação lagsur, para o *(PCVS)*, relaxa a restrição (5.6), tal com na relaxação lagrangeana, descrita anteriormente. Entretanto, Antes de adicionar esta restrição à função objetivo, esta restrição é relaxada conforme a relaxação surrogate. Desta forma, a restrição (5.6), após a relaxação surrogate, ficaria:

$$\sum_{i \in V} \lambda_i \left(\sum_{j<i} x_{ij} + \sum_{j>i} x_{ij} \right) = \sum_{i \in V} \lambda_i * 2 = 2 * \sum_{i \in V} \lambda_i,$$

onde λ_i pertence ao conjunto dos números reais e i pertence ao conjunto {1, 2,...,n}. Esta restrição é relaxada novamente, conforme o modo lagrangeano, e então adicionada à função

objetivo. Seja t, um número real qualquer, o multiplicador lagrangeano. Desta forma, teríamos a seguinte formulação matemática para o *lagsur*, no que se refere ao *(PCVS)*:

$$v(L_tS_lPCVS) = \min \left\{ \sum_{i \in V} \sum_{j>i} (c_{ij} + t.\lambda_i + t.\lambda_j)x_{ij} - 2.\sum_{i \in V} t.\lambda_i \right\}$$

(L_tS_lPCVS) sujeito a (5.10), (5.11), (5.12) e (5.8)

Para resolver o dual *lagsur*, isto é,

$$v(DL_tS_lPCVS) = \max \{v(L_tS_lPCV)\},\ \lambda \in \Re^n \text{ e } t \geq 0,$$

procede-se de forma análoga ao que é feito para a relaxação lagrangeana, isto é, usando o algoritmo de subgradientes, versão II. Entretanto, a relaxação *lagsur* precisa de buscar o melhor t, para um dado λ^* fixo, tal como descrito anteriormente. Na figura (5.1), a seguir, tem-se a representação qualitativa de um gráfico $v(L_tS_{l^*}PCVS)$ versus t, para um valor fixo de λ^*. Supondo que o valor de t que maximize a função objetivo seja t*, então tem-se que obter um valor de t tal que seja próximo de t*. No caso do *(PCVS)* , na busca do melhor t, tem-se o seguinte gráfico $v(L_tS_{l^*}PCV)$ versus t:

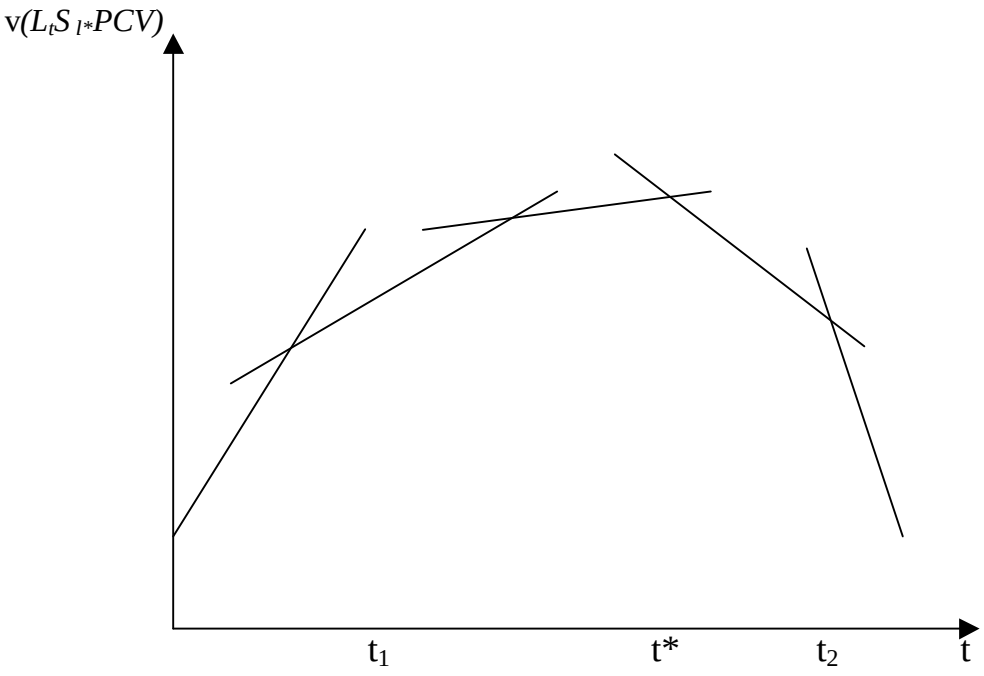

Figura 5.1 - Gráfico $v(L_tS_{l^*}PCV)$ versus *t*

No gráfico acima, t_1 e t_2 e t* são tais como descrito no *capítulo 3*. O algoritmo de busca do melhor t, usado para o *lagsur*, no caso do *(PCVS)*, está descrito a seguir.

Algoritmo de busca do melhor t

Dados *incremento = 1.5;*
k_max = 5 (número máximo de iterações);
t_1 := -¥ (limite inferior para o melhor t);
t := incremento (multiplicador lagrangeano/surrogate atual);
t_2 := ¥ (limite superior para o melhor t);
v := (melhor limite superior atual). Inicialmente, v* := -¥ ;*
k := 0 (número de iterações atual);
delta = 0.50 (diferença entre t_2 e t_1);

Repita
Fazer *k := k + 1;*
Se *k > k_max* ***ou*** *(t_2 - t_1) < delta* ***então pare;***
Senão resolva *($L_t S_l PCVS$)*
Fim_Se
Se *v($L_t S_l PCVS$) > v** ***então***
Fazer *melhor_t = t,*
v := v($L_t S_l PCVS$);*
Fim_Se
Calcule $w^l = \sum_{i \in V} l_i.(d_i - 2)$ *(d_i é o grau do vértice i e w^l é a inclinação da função lagsur);*

Se $w^l < 0$ ***então***
t_2 = t;
t = t - incremento;
Se *t_1 ¹ ¥ (t_1 já foi determinado)* ***então***
incremento = (t_2 - t_1)/2.0;
t = t + incremento;
Fim_Se
senão
t_1 = t;
*incremento = incremento*2;*
t = t + incremento;
Se *t ³ t_2* ***então***
t = t - incremento;
incremento = (t_2 - t_1)/2.0;
t = t + incremento;
Fim_Se
Fim_Se
Até *(condições de parada).*

5.5 - Resultados obtidos com as relaxações lagrangeana e lagsur

Para verificar o comportamento da relaxação *lagsur* e também da relaxação lagrangeana, foram usados problemas testes disponíveis na *Internet* no endereço http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/tsp, doravante chamada de (TSPLIB). Os resultados estão dispostos em uma tabela para uma melhor visualização. Os problemas testes usados são aqueles que tem solução ótima conhecida. Em cada tabela, tem-se os seguintes campos:

Prob - contém o número de cidades a serem consideradas

t - contém o tempo, em segundos, de execução para se resolver o dual lagrangeano/lagsur

n_iter - contém o número de iterações para que o dual lagrangeano/lagsur
venha a convergir

10%, 5%,.....,0.1% - contém o tempo necessário para o dual lagrangeano/lagsur
venha a atingir a porcentagem de 10%, 5%,.....,0.1% de diferença
em relação à solução ótima.

gap1, gap2 - são definidos como:

gap1 = (solução viável - relaxação)/solução ótima

gap2 = (solução ótima - relaxação)/solução ótima

Cada problema foi resolvido usando-se o algoritmo de subgradientes, versão II. A implementação foi em linguagem C. Os problemas forma executados em estação Sun Sparc Ultra, 128 Mbytes de memória, 167 Mhz.

5.5.1 - Resultados obtidos com a relaxação lagsur x lagrangeana

5.5.1.1 - Resultados obtidos com a relaxação lagrangeana

Prob	t	gap1	gap2	10%	5%	4%
16	2.0	0.000230	0.000230	1.00	1.00	1.00
22	9.380	0.000123	0.000123	2.94	4.130	4.37
48	23.0	0.008141	0.002355	1.00	1.00	2.00
52	6.0	0.002132	0.002132	1.00	2.00	2.00
100	52.00	0.044380	0.018157	0.00	6.00	9.00
225	675.0	0.090032	0.039181	2.00	67.00	495.00
442	6469.0	0.067190	0.007115	2.00	527.00	754.00
1002	52420.0	0.030597	0.030597	6853.00	27475.00	36714.00
1291	56066.0	0.023840	0.023842	14.00	420.00	2376.00
1304	42816.0	0.040637	0.040637	4234.90	28094.30	--
1379	38018.0	0.015077	0.015077	9.90	1918.60	2826.10
1655	128326.0	0.022040	0.02204	40.18	6405.25	12701.90
1748	64421.0	0.040159	0.040159	9739.00	48413.00	--
1889	87629.0	0.049982	0.049982	10786.00	87568.00	--
2152	99230.0	0.012201	0.012201	21.40	21.40	2900.10

Tabela 5.1 - Lagrangeano para instâncias do *(PCVS)* com solução ótima conhecida.

prob	n_iter	3%	2%	1%	0.5%	0.4%	0.3%	0.2%	0.1%
16	289	1.00	1.00	1.00	1.00	1.00	1.00	1.00	2.00
22	487	4.64	4.87	6.34	7.71	7.98	8.40	8.71	9.10
48	1431	4.00	7.00	11.00	15.00	17.00	19.00	--	--
52	282	2.00	3.00	4.00	5.00	5.00	5.00	--	--
100	714	15.00	27.00	--	--	--	--	--	--
225	950	--	--	--	--	--	--	--	--
442	3000	1119.0	1810.0	4054.0	--	--	--	--	--
1002	3000	--	--	--	--	--	--	--	--
1291	3000	13431.	--	--	--	--	--	--	--
1304	3000	--	--	--	--	--	--	--	--
1379	2986	4412.8	9465.7	--	--	--	--	--	--
1655	3000	29368.	--	--	--	--	--	--	--
1748	3000	--	--	--	--	--	--	--	--
1889	3000	--	--	--	--	--	--	--	--
2152	3000	11413.	31334.	--	--	--	--	--	--

Tabela 5.2 - Lagrangeano para instâncias do *(PCVS)* com solução ótima conhecida (continuação).

5.5.1.2 - Resultados obtidos com a relaxação lagsur

Prob	t	gap1	gap2	10%	5%	4%
16	1.10	0.000233	0.000233	0.17	0.19	0.20
22	7.00	0.006471	0.000096	1.01	1.16	1.20
48	8.00	0.009697	0.002988	0.15	1.00	1.00
52	6.50	0.002121	0.002121	0.33	1.00	1.00
100	28.00	0.050297	0.021871	0.56	3.00	3.00
225	652.00	0.098934	0.039154	2.28	83.00	392.00
442	997.00	0.084628	0.009726	1.95	82.00	92.00
1002	17856.00	0.011068	0.011068	455.50	869.50	1054.80
1291	15384.00	0.021880	0.021880	14.00	291.00	736.000
1304	15033.00	0.018360	0.018360	626.00	1511.00	2111.0
1379	12249.00	0.014109	0.014109	9.15	315.00	351.000
1655	128325.0	0.019877	0.019877	40.18	1251.000	1600.00
1748	64419.00	0.014932	0.014932	785.00	1802.00	2285.0
1889	87643.00	0.017504	0.017504	675.80	2275.40	3785.90
2152	99222.0	0.009182	0.009182	25.77	25.77	1106.0

Tabela 5.3 - Lagsur para instâncias do *(PCVS)* com solução ótima conhecida.

prob	n_iter	3%	2%	1%	0.5%	0.4%	0.3%	0.2%	0.1%
16	264	0.22	0.89	0.93	1.00	1.00	1.00	1.00	1.03
22	372	1.28	1.41	1.72	2.02	2.36	2.52	3.94	4.67
48	521	1.00	3.00	6.00	7.00	7.00	8.00	--	--
52	309	1.00	4.00	5.00	5.00	5.00	5.00	--	--
100	373	4.00	14.00	--	--	--	--	--	--
225	882	--	--	--	--	--	--	--	--
442	506	110.00	152.00	997.00	--	--	--	--	--
1002	905	1428.4	2514.5	--	--	--	--	--	--
1291	618	3230.0	--	--	--	--	--	--	--
1304	1057	3678.0	9060.0	--	--	--	--	--	--
1379	962	440.0	3147.0	--	--	--	--	--	--
1655	3000	3029.0	42846.	--	--	--	--	--	--
1748	3000	3098.0	6716.0	--	--	--	--	--	--
1889	3000	9080.7	37393.	--	--	--	--	--	--
2152	3000	1805.0	3648.0	21829.	--	--	--	--	--

Tabela 5.4 - Lagsur para instâncias do *(PCVS)* com solução ótima conhecida (continuação).

5.5.1.3 - Detalhes de implementação das tabelas

Para os problemas de até 442 cidades, para se obter a solução viável, foi usada uma heurística, baseada no algoritmo de Kruskal. Esta heurística está descrita a seguir.

Seja E o conjunto de todos os arcos, tal como descrito anteriormente. Cada arco liga o vértice i ao vértice j. A caminho de i até j é o mesmo de j a i. Seja R o conjunto dos arcos (i,j) a que são escolhidos para comporem a solução viável. Feitas estas duas considerações, temos então o seguinte algoritmo:

$R \leftarrow \phi$

Enquanto (R contém menos de n - 1 arcos) e (o conjunto E for não vazio) **faça**

escolha um arco (i,j) do E, de menor custo;

suprima (i,j) do conjunto E;

Se (i,j) não cria um ciclo em R e o grau de i e j forem menores do que 2

então adicione (i,j) a R

senão suprima (i,j)

fim_enquanto

Se R contém menos de n vértices

então Imprimir (´nenhuma solução viável foi gerada´)

senão unir os dois únicos vértices que estão com grau 1.

Esta heurística foi escolhida para se obter soluções viáveis para as relaxações lagrangeana e *lagsur* devido ao fato de ser mais fácil de se resolver e também pelo fato de

que, para se escolher um arco de menor custo, esta heurística aproveita a ordenação feita pela heurística de Kruskal. Outras heurísticas foram testadas (algoritmo do vizinho mais próximo [36] menor distância, Christofides [13,36]). Entretanto, estas heurísticas consumiam muito tempo, inviabilizando os testes. Foi observado também que os resultados obtidos por elas não melhoravam significamente a solução viável.

Para problemas com 1002 cidade em diante, não foi usada a heurística para se obter a solução viável. Quando o programa chamava a heurística que fornecia o valor da solução viável, o valor fornecido era o da solução ótima, a qual e´conhecida. Isto foi feito devido ao fato de que o tempo para se executar o método subgradientes para instâncias com valores acima de 1002 cidades aumentou demais ao se usar a heurística para fornecer a solução viável. Desta forma , visto que o enfoque desse trabalho é a relaxação *lagsur*, optou-se por desistir do algoritmo e fornecer a solução ótima conhecida diretamente. Entretanto, a heurística usada forneceu bons valores para o limite superior até 442 cidades. A maioria dos resultados foi muito próximo do ótimo. O pior resultado foi para 1002 cidades, onde o desvio em relação à solução ótima foi de 7%.

Para problemas a partir de 1655 cidades, o critério de parada foi o limite de 3000 iterações para que melhor se observasse o desempenho do *lagsur* e da versão lagrangeana. Com apenas este critério, pode-se ver que o *lagsur* consegue porcentagens da solução ótima em tempos bem menores do que a versão lagrangeana.

A instância com maior número de cidades usada nos testes foi 2152 cidades. Existiam outras instâncias com um número maior de cidades (3000, 4000 e 7000) para serem testadas. Entretanto, conforme foi citado anteriormente, seria preciso uma máquina dedicada para os testes pois, para o *lagsur* e a relaxação lagrangeana, com limite de 3000 iterações, não se atingia sequer 95% da solução ótima e o tempo de execução da relaxação lagrangeana, para 3000 cidades, demorou em torno de 15 dias. Para um número maior de cidades, 4000 e 7000, estima-se uma maior quantidade de dias e mesmo assim não se pode

garantir que, para estes valores, as duas relaxações iriam conseguir atingir a pelo menos 95%.

5.5.2 - Gráficos lagsur versus iteração e lagrangeano versus iteração

Para uma melhor visualização do comportamento do dual da relaxação *lagsur* e o comportamento do dual da relaxação lagrangeana para o *(PCVS)* em cada iteração, tem-se 3 comparações entre os gráficos dos dois duais. São usadas 3 instâncias diferentes. Estas instâncias são conhecidas como att48.tsp (48 cidades), pcb442.tsp (442 cidades), e pr1002.tsp (1002 cidades) e foram retiradas da biblioteca de problemas para o caixeiro viajante (TSPLIB). Os gráficos são dispostos, a seguir, para 48, 442 e 1002 cidades.

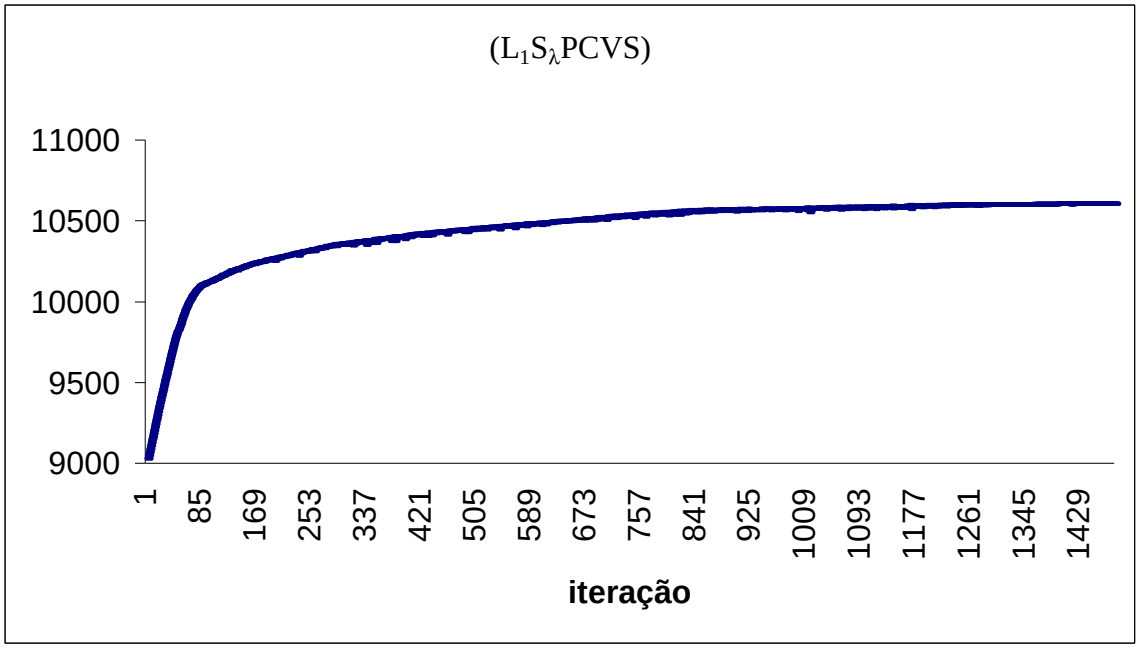

Figura 5.2 - Gráfico *(L_1S_lPCVS)* versus *iteração* para 48 cidades.

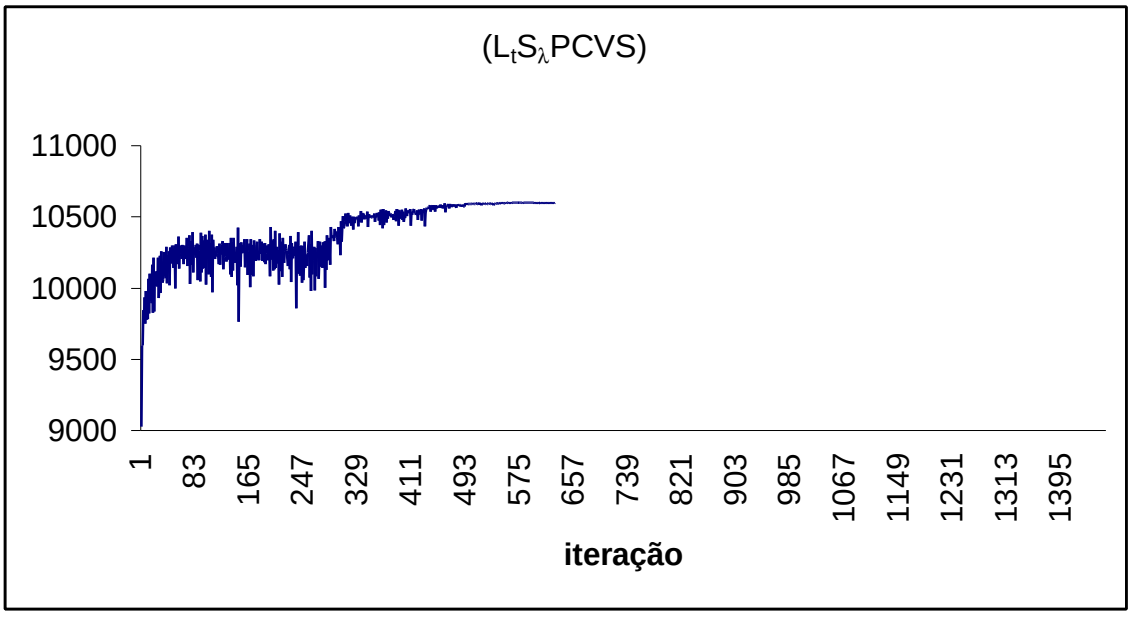

Figura 5.3 - Gráfico *(L_tS_lPCVS)* versus *iteração* para 48 cidades.

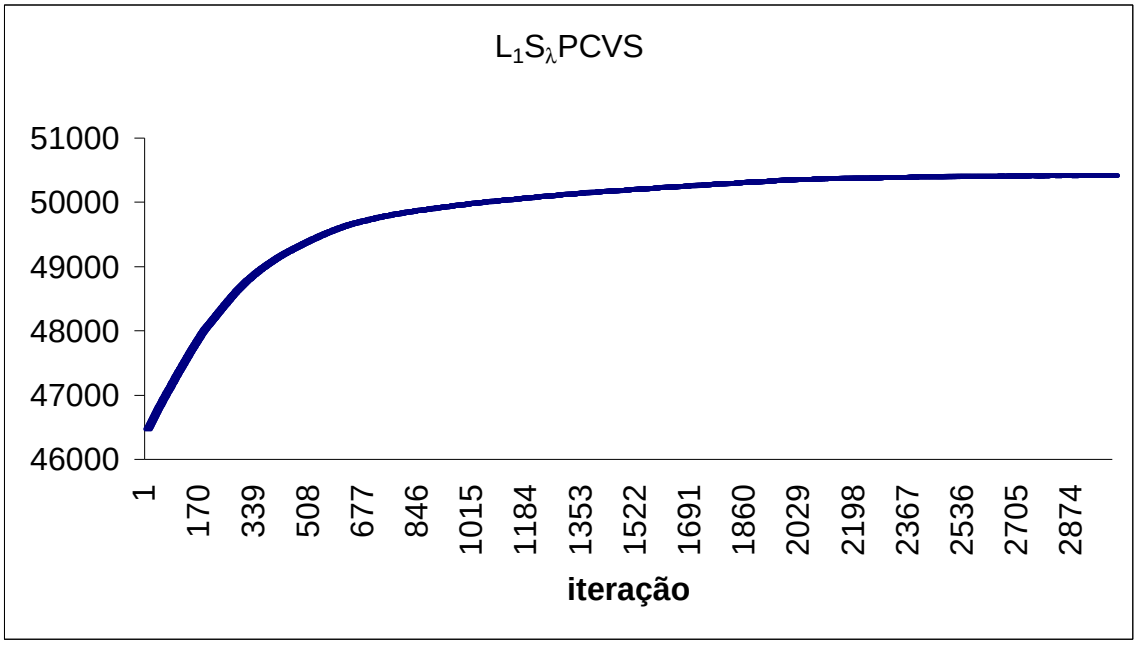

Figura 5.4 - Gráfico *($L_1S_\lambda PCVS$)* versus *iteração* para 442 cidades.

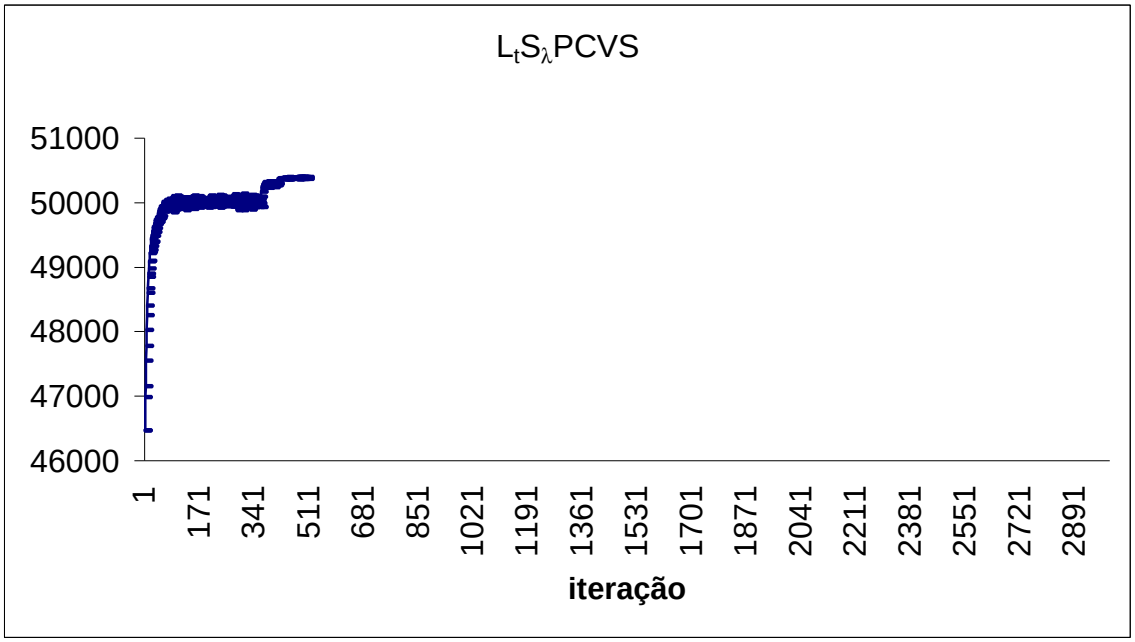

Figura 5.5 - Gráfico *($L_tS_\lambda PCVS$)* versus *iteração* para 442 cidades.

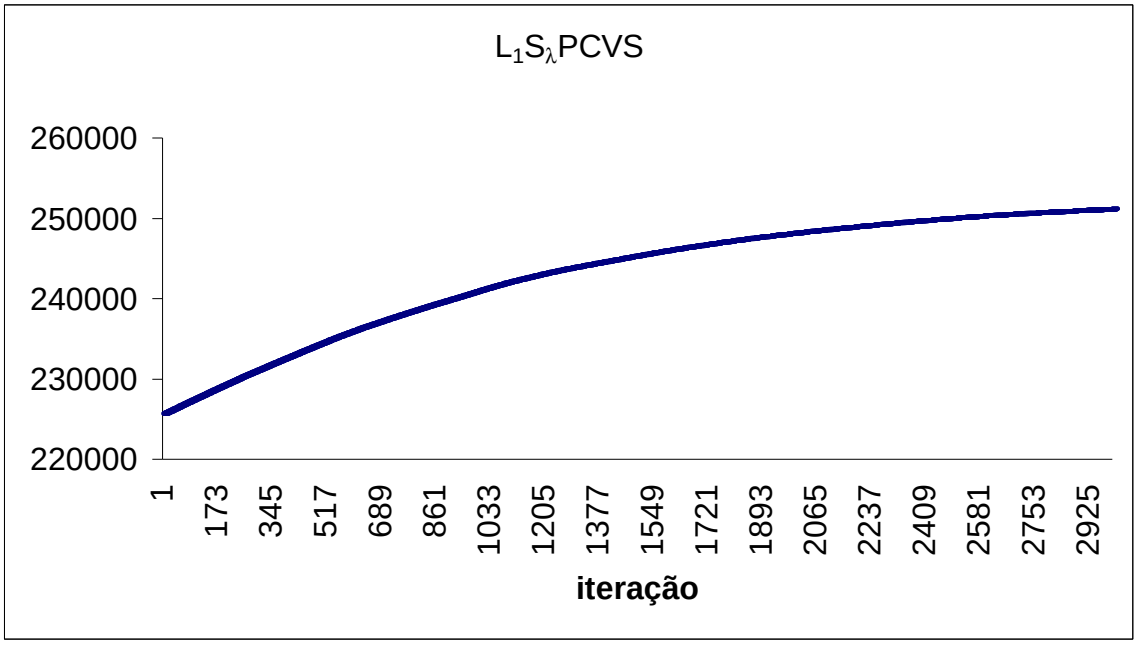

Figura 5.6 - Gráfico *($L_1S_\lambda PCVS$)* versus *iteração* para 1002 cidades.

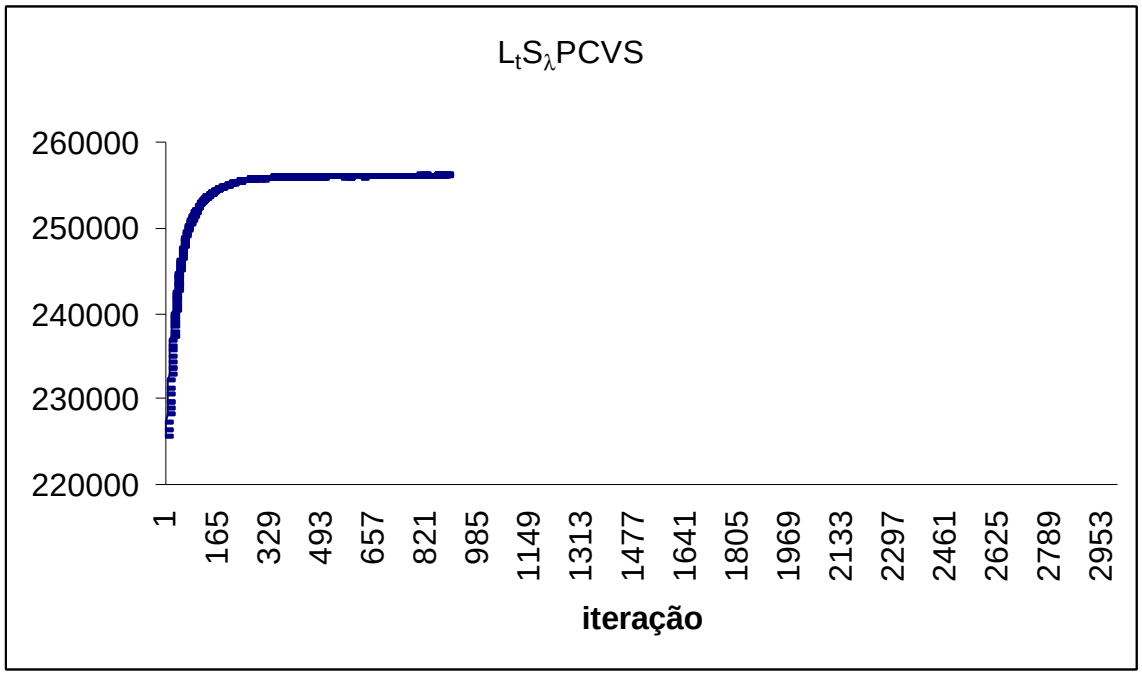

Figura 5.7 - Gráfico *($L_tS_\lambda PCVS$)* versus *iteração* para 1002 cidades.

Para as instâncias acima, não foi usado o algoritmo que fornece solução viável em cada iteração do método subgradientes. A solução viável fornecida em cada iteração era o valor da solução ótima, a qual é conhecida para cada instância.

Os gráficos dos exemplos acima resumem o que ocorreu com os testes. Nas instâncias usadas nos testes com a relaxação *lagsur* e a relaxação lagrangeana, observou-se que a sequência de pontos fornecida pelo dual lagrangeano era crescente e praticamente monótona. Entretanto, a convergência é muito lenta. A relaxação *lagsur* converge mais rapidamente. Entretanto, no início da sequência de pontos fornecida pelo dual *lagsur*, o comportamento é instável e vai se estabilizando à medida em que o número de iterações vai crescendo. Observe que os primeiros valores da sequência fornecida pelo dual *lagsur* são maiores do que os valores fornecidos pelo dual lagrangeano.

Nos testes feitos, observou-se que, para a maioria dos casos, o valor do melhor t fixado foi em 46.5 após este valor se repetir em 5 iterações consecutivas do método subgradientes. Para alguns casos, o melhor valor de t foi 22.5. Desta forma, o valor de λ, após t ter sido fixado, era sempre multiplicado por um escalar rasoavelmente grande (22.5 ou 46.5). Isto poderia explicar o motivo do comportamento oscilante em uma boa parte da sequência de pontos fornecidos pelo dual *lagsur*. Observe ainda que a fixação, após 5 primeiras iterações consecutivas em que o melhor valor de t se repita, foi equivalente a, nesse caso, a fixar desde a primeira iteração os valores de 46.5, para a maioria das instâncias testadas, ou 22.5.

5.6 - Conclusão

A relaxação *lagsur*, para problemas de pequeno porte, obteve um comportamento mais instável do que a relaxação lagrangeana nas primeiras iterações, ou ainda, em uma boa parte do número de iterações. Entretanto, convergiu mais rapidamente. Para um número de cidades maiores do que 442, para as instâncias testadas, a relaxação *lagsur* foi ficando mais estável e também continuou a convergir mais rapidamente.

À medida em que o número de cidades vai aumentando, é necessário um número maior de iterações para a relaxação *lagsur* ou lagrangeana convergir. Desta forma, para um limite de 3000 iterações, para algumas instâncias (a partir de 1655 cidades), a relaxação *lagsur* não conseguiu convergir pelos outros critérios de parada a não ser o limite máximo de 3000 iterações. Porém, o dual da relaxação lagrangeana, a partir de 442 cidades, na maioria das instâncias, não conseguiu convergir antes de 3000 iterações. À medida em que o número de cidades vai crescendo, é necessário estabelecer um limite maior para o número de iterações para melhor ver o comportamento do dual das duas relaxações. Entretanto, para isto ser feito, seria necessário máquinas dedicadas exclusivamente aos testes, com uma maior quantidade de memória e também uma maior velocidade pois o tempo de execução pode demorar semanas ou meses.

Observando o gap2 nas tabelas, pode-se observar que a relaxação *lagsur* tem aproximadamente os mesmos valores para quando o dual da relaxação lagrangeana converge antes de 3000 iterações. Quando a o dual da relaxação lagrangeana não consegue convergir antes de 3000 iterações, a relaxação *lagsur* é superior em termos de resultados. Isto pode ser melhor visto através do item "porcentagens" (10% a 0.10). Na maioria dos casos, a relaxação *lagsur* atinge a uma dada porcentagem em um número de iterações/tempo menor que a relaxação lagrangeana. Desta forma, ao se atingir a 3000 iterações, a relaxação lagrangeana não terá atingido ainda a uma dada porcentagem que a

relaxação lagsur atinge. Em suma, a relaxação *lagsur* atinge a valores mais próximos da solução ótima em menos iterações/tempo do que a relaxação lagrangeana.

CAPÍTULO 6

6 - Conclusão

6.1 - Conclusão geral sobre o lagsur

A relaxação lagsur conseguiu nos testes computacionais, para um mesmo limite máximo de iterações, obter resultados mais próximos da solução ótima dos problemas que a relaxação lagrangeana.

No *(PGA)*, observamos que na grande maioria dos casos, a relaxação *lagsur* obteve resultados melhores do que a lagrangeana, no que se refere a tempo e também em limites. A relaxação proposta por Jornsten e Nasberg, que converge mais lentamente, mostrou de uma de forma mais nítida a superioridade do *lagsur* sobre a relaxação lagrangeana. Nas outras duas relaxações propostas (relaxando a restrição das capacidades e relaxando a restrição da integralidade), também podemos notar que a relaxação *lagsur* foi superior a lagrangeana, conforme os resultados mostraram. Desta forma, com menos esforço computacional, podemos obter bons resultados (tão bons quanto aos da relaxação lagrangeana) usando a relaxação *lagsur*.

No problema do caixeiro viajante, os resultados mostraram que a relaxação *lagsur* obtém resultados muito mais rapidamente do que a relaxação lagrangeana. Vale a pena mencionar que quanto mais lenta for a convergência, devido a característica de cada problema, mais nitidamente se nota o fato de que a relaxação *lagsur* fornece melhores resultados em menos tempo. Desta forma, para relaxações de convergência lenta no algoritmo de subgradientes, versão II, parece ser adequada a utilização da relaxação lagsur pois fornece bons resultados em menor tempo que a lagrangeana.

Conforme foi mencionado anteriormente, outros algoritmos de subgradientes poderiam ser testados com a abordagem lagsur. O lagsur, assim como a relaxação lagrangeana, é independente do métodos subgradientes usados.

O problema dual local (DL_tS_{l*}) (veja seção 3.2) parece ser a indicação do sucesso no emprego da relaxação lagsur. Entretanto, suponha que, para λ^* tenhamos uma situação como a da figura (6.1) abaixo

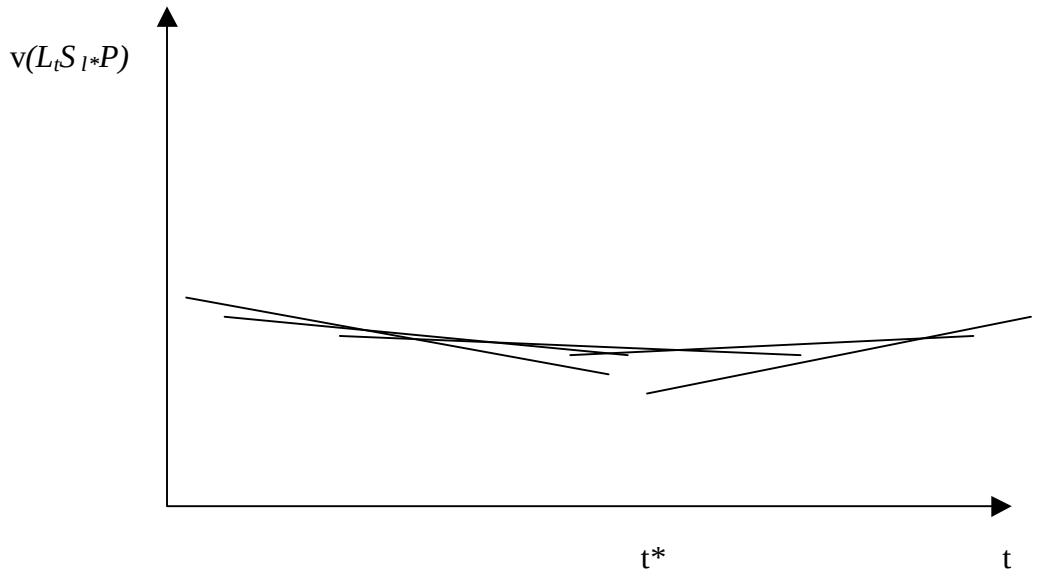

Figura 6.1 - Gráfico v$(L_tS_{l*}P)$ versus t

Neste caso, t=1 ou qualquer outro valor de t não produzirá um valor v$(L_tS_{l*}P)$ significativamente diferente de v$(L_{t*}S_{l*}P)$. Seria, portanto, indiferente o uso de qualquer das relaxações (lagrangeana ou lagsur). Esta situação poderá ocorrer quando os dados não possuem uma dispersão adequada, principalmente nos custos da função objetivo, ou ainda quando se está próximo da convergência do algoritmo de subgradientes (o multiplicador inicial é bom). Pode-se, portanto, esperar algum insucesso na aplicação do lagsur para esses casos.

Como sugestão do emprego do lagsur, pode-se tomar como base o resultado da busca de t na primeira iteração do algoritmo de subgradientes, versão II, e conforme a diferença | v$(L_tS_{lo}P)$ - v$(L_1S_{lo}P)$| , tomar a decisão de usar ou não o lagsur.

Os algoritmos de busca unidimensional, usados para o *(PGA)* e o *(PCVS*), se mostraram adequados nos testes computacionais. Outros algoritmos poderiam ser testados.

O lagsur pode ainda ser testado incorporando-o a um algoritmo exato do tipo "branch and bound". Uma busca local poderá ser suficiente para melhorar o limite lagrangeano, com a correspondente economia em tempo, fator crucial nos algoritmos exatos. Entretanto, esta sugestão não foi testada.

Uma última conclusão é que o uso da relaxação *surrogate contínua* pode ser amplamente substituído pela relaxação *lagsur* com busca do melhor t em algumas iterações. Além do fato da relaxação *surrogate contínua* ter que ser de fácil resolução, o que não é frequente, sua resolução exata toma maior tempo computacional que a relaxação *lagsur* com as buscas unidimensionais propostas.

Espera-se que a relaxação *lagsur* venha no futuro a ter grande uso tal como o atual emprego da relaxação lagrangeana.

7 - REFERÊNCIAS BIBLIOGRÁFICAS

[1] Almiñana, Marcos e Pastor, T. J. "Na adaptation of SH heuristic to the location set covering problem". European Journal of Operational Research, 100, 586-593, 1997.

[2] Allen, E., Helgason, R., Kennington, et al. "A generalization of Poliak's convergence results for subgradient optimization". Mathematical Programming, 37, 309-317, 1987.

[3] Balachandran, V. "An integer generalized transportation model for optimal job assignment in computers networks". Operations Research, 24:(4), 742-749, 1976.

[4] Balas, E., Padberg, M. W. "Set partitioning - a survey" in Christofides, (ed.), Combinatorial Optimization, Wiley, New York, 1979.

[5] Banerjee,K. "Generalized Lagrange multipliers in Dynamic Programming", Research Report No. ORC 71-12, Operations Research Center, University of California, Berkeley, California, 1971.

[6] Bazaraa, M. S., Sherali, H. D. "On the choice of step size in subgradient optimization". European Journal of Operational Research, 7, 380-388, 1981.

[7] Beasley, J. E. A lagrangian heuristic for set covering problem. Naval Research Logistics 37, 151-164, 1990.

[8] Beasley, J. E. "OR-Library: Distributing test problems by eletronic mail". Journal of Operational Research Society, 41:(11), 1069-1072, 1990.

[9] Benders, J. F., "PartitioningProcedures for Solving Mixed-Variables Programming Problems", Numerische Mathematik, 6, pp 63-72, 1962.

[10] Brännlund, U., " A Generalized subgradient method with relaxation step", *Mathematical Programming* 71 (1995) 207-219.

[11] Camerini, P. ; Fratta, L. and Maffioli F., On improving relaxation methods by modified gradient techniques" *Mathematical Programming Study*, 3, (1975) 26-34.

[12] Catrysse, D. Van Wassenhove, L. N. "A survey of algorithms for the Generalized Assignment Problem". European Journal of Operational Research, 60, 260-272, 1992.

[13] Christofides, N. “The Traveling Salesman Problem”. Combinatorial Optimization, Wiley, Chichester, 1979.

[14] Correa, R. and Lemaréchal, C, Convergence of some algorithms for convex minimization”, *Mathematical Programming* 62 (1993) 261-275.

[15] Day, R. H. "On optimal extracting from a multiple file data storage system: an application of integer programming". Operations Research, 13:(3), 482-494, 1965.

[16] De Maio, A., Roveda, C. An all zero-one algorithm for a certain class of transportation problems". Operations Research, 19, 1406-1418, 1971.

[17] Dudzinsky, k., Waluckiewicz, S. "A fast algorithm for the linear multiple-choice knapsack problem". Operations Research Letters, 3(4), 205-209, 1984.

[18] Dyer, M. E. "Calculating surrogate constraints". Mathematical Programming, 19, 255-278, 1980.

[19] Ermol'ev, Y. M. "Methods for solving nonlinear extremal problems". Cybernetics, 16/1, 1-14, 1966.

[20] Espejo, L. G. A. "Relaxações e heurísticas para o problema de localização de máxima cobertura". (Dissertação de Mestrado em Engenharia de Produção) - COPPE/UFRJ, Rio de Janeiro, 1997.

[21] Fisher, M. L. "The lagrangian relaxation method of solving integer programming problems". Management Science, 27, 1-18, 1981.

[22] Fisher, M. L., Jaikumar, R. A generalized assignment heuristic for vehicle routing". Networks, 11, 109-124, 1981.

[23] Fisher, M. L., Jaikumar, R., Wassenhove, L. N. V. "A multiplier adjustment method for the generalized assignmet problem". Management Science, 32, 1095-1103, 1986.

[24] Foster, B. A., Ryan, D. M. "An integer programming approach to the vehicle scheduling problem". Operational Research Quarterly, 27, 367-384, 1976.

[25] Garfinkel, R. S., Neeb, A. W., Rao, M. R. "The m-center problem: bottleneck facility location". Working paper number 7414, Graduate School of Management, University of Rochester, Rochester, N. Y., 1974.

[26] Geoffrion, A. " Lagrangean relaxation and its uses in integer programming". Mathematical Programming Study, 2, 82-114, 1974.

[27] Glover, F. "A multiphase dual algorithm for the zero-one integer programming problem". Operations Research, 13, 879-919, 1965.

[28] Glover, F. "Surrogate constraints". Operations Research, 16(4):741-749, 1968.

[29] Glover, F. "Surrogate Constraints Duality in Mathematical Programming". Operations Research, 23, pp 434-451, 1975.

[30] Goffin, J. L. "On convergence rates of subgradiente optimization methods", Mathematical Programming, 13, 329-347, 1977.

[31] Greenberg, H. J., Pierskalla, W. P. "Surrogate Mathematical Programming". Operations Research, 18, 924-939, 1970.

[32] Handler, G., Zang, I. "a dual algorithm for the constrained shortest path problem". Networks, 10, 193-310, 1980.

[33] Held. M., Karp, R. M. " The Traveling salesman problem and minimum spanning trees". Operations Research, 18,1138-1162, (1970).

[34] Held. M., Karp, R. M. " The Traveling salesman problem and minimum spanning trees: Part II". Mathematical Programming 1, 6-25, (1971).

[35] Held. M., Wolfe, P., Crowder, H. P. "Validation of subgradient optimization", Mathematical Programming, 6, 62-88, 1974.

[36] Horowitz, E., Sahni, Sartaj. Fundamentos de Estrutura de Dados.Editora Campus, Rio de Janeiro, 1987.

[37] Jonhson, D.S., McGeoch, L.A., Rothberg, E. E. "Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound. Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, 341-350, 1996.

[38] Jornsten, K, Nasberg, M. "A new Lagrangian relaxation approach to the generalized assignment problem". European Journal of Operational Research, 27, 313-323, 1986.

[39] Jornsten, K.; Varbrand, P. "Relaxation techniques and valid inequalities applied to the generalized assignment problem". Asia Pacific Journal of Operational Research, 7, 172-189, 1990.

[40] Karwan, M. H., Rardin, R. L. "Some realtionships between lagrangian and surrogate duality in integer programming". Mathematical Programming, 17, 320-334, 1979.

[41] Kim, S. and Um, B. S., “Polyak’s subgradient method with simplified projection for nondifferentiable optimization with linear constraints”, *Optimization* 20 (1989) 451-456.

[42] Kim, S. and Um, B. S., “An improved subgradient method for constrained nondifferentiable optimization”, *Operations Research Letters* 14 (1993) 61-64.

[43] Klastorin, T. D. "An effective subgradiente algorithm for the Generlized Assignment Problem". Computers and Operations Research, 6, 155-164, 1979.

[44] Larsson, T. ; Patriksson, M. and Strömberg, A-B., “Conditional subgradient optimization - theory and applications”, *European Journal of Operational Research* 88 (1996) 382-403.

[45] Lenstra, J. K., Lawler, E. L., Rinnooy Kan, A. H. G., Shmoys, D. D. "The Traveling Salesman Problem". John Wiley & Sons. New York, USA, 1985.

[46] Lopes, F. B. "Nova heurística para o problema de cobertura de conjuntos". (Tese de Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais. São José dos Campos, 1992. (INPE - 5471 - TDI/502).

[47] Lorena, L. A. N., Lopes, F. B. " A surrogate heuristic for set covering problems". European Journal of Operational Research. 79(1), 138-150, 1994.

[48] Lorena, L. A. N., Narciso, M. G. "Relaxation heuristics for a generalized assignment problem". European Journal of Operational Research, 91(1), 600-610, 1996.

[49] Lorena, L. A. N., Plateau, G and Freville, A. "New subgradiente algorithms for the multiknapsack lagrangean and surrogate duals". Pre-publication # 90. LIPN-Universite Paris Nord, 1990.

[50] Martello, S Toth, P. "Knapsack Problems - Algorithms and Computer Implementations". John Wiley & Sons. New York, USA, 1990.

[51] Minoux, M.. "Plus courts chemins avec constraints: Algorithmes et applications", Annals of Telecommunications, 30, 383-394, 1975.

[52] Narciso, M. G. "Novas heurísticas para o problema generalizado de atribuição". (Dissertação de Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais. São José dos Campos, 1994. (INPE - 5607 - TDI/730).

[53] Narciso, M. G., Lorena, L. A.N. "Lagrangean/surrogate relaxation for generalized assignment problem". European Journal of Operational Research. A aparecer. (1998)

[54] Noschang, M. H. The Traveling Salesman Problem - a review of theory and current research.

[55] Parker, R. G., Rardin, L. R. "Discrete Optimization". Academic Press, INC, London, 1988.

[56] Poljak, B. T. "Minimization of unsmooth functionals". USSR Computational Mathematics and Mathematical Physics 9 (1969) 14-29.

[57] Quine, W. V. "A way to simplify truth functions". Am. Math. Mon., 62, 627-631, 1955

[58] Ross, G. T., Soland, M. S. "A branch and bound algorithm for the generalized assignment problem". Mathematical Programming, 8, 91-103, 1975.

[59] Salverson, M. E. "The assembly line balancing problem". Journal of Industrial Engineering, 6, 18-25, 1955.

[60] Sarin, S., Karwan, M. H., Rardin, R. L. "A new surrogate dual multiplier search procedure". Naval Research Logistics, 34, 431-450, 1987.

[61] Senne, E.L.F. and Lorena, L.A.N **"**A lagrangean/surrogate approach to facility location problems", EURO-TIMS Congress - Barcelona (1997).

[62] Shapiro, J. F. "Generalized lagrange multipliers in integer programming". Operations Research, 19, 68-76, 1971.

[63] Volgenant, T., Jonker, R. " A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation". European Journal of Operational Research, 9, 83-89, 1982.

APÊNDICE A

Resultados obtidos para o (PGA) e o (PCVS)

A . 1 -Resultados obtidos com o (PGA)

Antes de descrever os resultados obtidos, vale a pena descrever os tipos de instâncias (problemas) que serão usadas para os testes com o *(PGA)*. As instâncias usadas para os testes com o *(PGA)* são conhecidas como instâncias de classe A, B, C, D. Estas instâncias são definidas a seguir.

Seja Xmn um tipo de problema (ou instância), onde X denota um problema da classe A, B, C ou D. O índice m significa número de agentes e n significa número de tarefas. Cada problema Xmn vem em um arquivo diferente. Cada arquivo contém valores para p_{ij}, w_{ij}, e b_i. Desta forma, tem-se as seguintes leis para cada uma das classes:

a) Classe A - $w_{ij} \in [5,25]$, $p_{ij} \in [1,40]$, e

$$b_i = 9.(n/m) + 0.4\max \Big(\sum_{j \in N_i} w_{ij} \Big)$$

$i \in M$, $M = \{1,...,m\}$, $j \in N$, $N = \{1,...,n\}$

$N_i = \{j \in N$ tal que, na coluna j, p_{ij} seja máximo$\}$

b) Classe B - $w_{ij} \in [5,25]$, $p_{ij} \in [1,40]$, e

$$b_i = 0{,}7.[9.(n/m) + 0.4\max (\sum_{j \in N_i} w_{ij})]$$

$i \in M$, $M = \{1,...,m\}$, $j \in N$, $N = \{1,...,n\}$

$N_i = \{j \in N$ tal que, na coluna j, p_{ij} seja máximo$\}$

c) Classe C - $w_{ij} \in [5,25]$, $p_{ij} \in [1,40]$, e

$$b_i = 0{,}8.(\sum_{j \in N} w_{ij} /m)]$$

$i \in M$, $M = \{1,...,m\}$, $j \in N$, $N = \{1,...,n\}$

d) Classe D - $w_{ij} \in [1,100]$, $p_{ij} \in [w_{ij}, w_{ij}+ 20]$, e

$$b_i = 0{,}8.(\sum_{j \in N} w_{ij} /m)]$$

$i \in M$, $M = \{1,...,m\}$, $j \in N$, $N = \{1,...,n\}$

Feitas estas considerações, pode-se então descrever os resultados obtidos. Estes resultados estão dispostos em tabelas. Em cada tabela, tem-se os seguintes campos:

prob - instância da classe C (m por n) com solução ótima conhecida ou problemas do tipo A, B, C e D (m por n) sem solução viável conhecida previamente.

t- tempo gasto para resolver o problema

gap1 - (limite superior - limite inferior)/(solução ótima).

gap2 - ((solução ótima) - limite inferior)/(solução ótima).

gap3 - (limite superior - (solução ótima))/(solução ótima).

n_iter - número de iterações total para resolver o problema (número de vezes que a relaxação é resolvida).

5%,4%, .., 0.5% - tempo médio para a solução ótima ter 5%, 4%, ...,0.5% de diferença em relação ao limite superior.

Os problemas da classe C (m por n) são obtidos na OR-Library [8] e tem solução ótima de cada problema conhecida. O maior problema tem dimensões 10 por 60. O menor problema tem dimensões 5 por 15. Na OR-Library existem outros problemas, com dimensões maiores mas com solução ótima não conhecida. Alguns dos problemas tiveram a solução ótima obtida durante os testes, observando-se a diferença entre o melhor limite inferior de um determinado problema e o melhor limite superior deste mesmo problema. Se a diferença entre estes limites for menor do que 1, então o melhor limite inferior obtido para um dado problema é a solução ótima, visto que o valor da solução ótima é inteiro e que o limite inferior é uma solução viável obtida. Com estas considerações feitas, temos os seguintes valores obtidos, isto é, as soluções ótimas obtidas (SO) e as melhores soluções viáveis (MSV) obtidas:

A5x100 = 4456 (SO), A5x200 = 8788 (SO), A10x100 = 4700 (SO),

A10x200 = 9413 (SO), A20x100 = 4857 (SO), A20x200 = 9666 (SO)

B5x100 = 4008 (MSV), B5x200=8502 (MSV), B10x100= 4633 (SO),

B10x200=9255 (MSV), B10x200 = 4817 (SO), B20x200 = 9670 (MSV)

C5x100 = 4411 (SO), C5x200 = 8347 (MSV), C10x100 = 4528 (MSV),

C10x200 = 9247 (MSV), C20x200 = 4784 (MSV), C20x200 = 9611 (MSV)

D5x100 = 9147 (SO), D5x200 = 18750 (SO), D10x100 = 10349 (SO),

D10x200 = 20562 (SO), D20x100 = 10839 (SO), D20x200 = 21733 (SO).

Todos estes problemas com solução ótima não conhecida foram resolvidos com programas escritos em linguagem C, e executados em uma estação SUN Sparc 5, 70 Mhz, 128 Mb de RAM. Os problemas com solução ótima conhecida foram resolvidos em um micro 486/DX2, 60 Mhz, 16M de memória RAM. Os programas fontes usados para resolver estes problemas no micro foram os mesmos que foram executados na estação Sparc 5.

Após estes comentários, tem-se a seguir os resultados obtidos para quando a relaxação for a restrição das capacidades, da atribuição e a proposta de Jornsten e Nasberg, a qual foi descrita anteriormente no capítulo 4.

A.1.1 - Relaxação da restrição das capacidades

A.1.1.1 - Relaxação lagrangeana

prob	t_med	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	0.703	38.33	8.991	30.54	275	0.033	0.099	0.257	--	--	--
C5x20	0.912	21.38	3.323	18.51	305	0.011	0.022	0.055	--	--	--
C5x25	1.022	9.577	1.044	8.621	303	0.033	0.033	0.033	0.033	0.211	--
C5x30	1.143	13.71	2.142	11.74	288	0.044	0.044	0.055	0.077	--	--
C8x24	1.440	17.70	2.145	15.88	295	0.055	0.055	0.066	0.090	--	--
C8x32	1.692	13.82	2.381	10.95	340	0.066	0.077	0.088	0.121	1.117	--
C8x40	2.022	10.31	4.218	6.156	279	0.088	0.099	0.121	0.132	0.209	--
C8x48	3.418	11.24	5.300	6.003	325	0.132	0.132	0.132	0.143	0.429	--
C10x30	2.121	16.02	6.133	10.07	299	0.066	0.099	0.110	0.198	--	--
C10x40	2.506	11.64	4.189	7.547	273	0.154	0.165	0.165	0.187	0.780	--
C10x50	3.231	8.201	3.247	5.000	317	0.220	0.242	0.264	0.275	0.351	--
C10x60	3.670	6.258	3.321	2.957	307	0.297	0.308	0.330	0.352	0.396	0.890

Tabela A1 - Lagrangeano para problemas da classe C com solução ótima conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a das capacidades.

prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	0.250	0.213	0.000	0.213	34	0.160	0.160	0.170	0.180	0.190	0.200
A 5x200	0.720	0.180	0.000	0.180	26	0.550	0.580	0.620	0.660	0.680	0.710
A 10x100	0.630	1.316	0.638	0.679	56	0.380	0.400	0.420	0.440	0.450	0.470
A 10x200	1.980	0.104	0.000	0.104	70	1.400	1.470	1.550	1.640	1.710	1.730
A 20x100	1.520	0.369	0.000	0.369	72	1.030	1.070	1.100	1.130	1.170	1.190
A 20x200	4.990	0.211	0.000	0.211	108	3.670	3.790	3.950	4.070	4.200	4.270
B 5x100	0.220	124.2	108.5	17.94	36	0.140	0.140	0.150	0.170	--	--
B 5x200	3.170	2.694	1.176	1.522	320	0.530	0.530	0.530	0.530	0.530	0.530
B 10x100	2.500	4.699	3.238	1.469	329	0.260	0.260	0.260	0.280	0.320	0.360
B 10x200	6.280	3.531	2.701	0.833	350	1.060	1.060	1.060	1.060	1.110	1.160
B 20x100	4.880	8.988	5.813	3.204	305	0.610	0.660	0.700	0.750	0.820	0.910
B 20x200	15.06	2.899	0.724	2.181	463	2.120	2.270	2.430	2.560	2.720	2.950
C 5x100	1.580	2.837	1.134	1.707	393	0.130	0.130	0.130	0.130	0.130	0.140
C 5x200	3.470	4.851	3.714	1.143	345	0.530	0.530	0.530	0.530	0.530	0.530
C 10x100	3.040	7.270	1.106	6.209	367	0260	0260	0260	0260	0.310	--
C 10x200	5.560	2.358	0.108	2.255	324	1.060	1.060	1.060	1.060	1.110	1.160
C 20x100	7.210	9.525	4.390	5.185	389	0.530	0.530	0.560	0.610	0.780	--
C 20x200	14.29	5.530	2.393	3.154	422	2.110	2.110	2.210	2.360	2.520	2.700
D 5x100	0.170	0.990	0.000	---	23	---	---	---	---	---	---
D 5x200	0.590	98.12	0.000	---	23	---	---	---	---	---	---
D 10x100	1.560	0.120	0.000	0.120	197	1.000	1.030	1.070	1.110	1.230	1.320
D 10x200	5.360	0.042	0.000	0.042	325	4.050	4.210	4.370	4.540	4.770	4.910
D 20x100	3.29	0.088	0.000	0.088	219	2.360	2.440	2.520	2.620	2.740	2.860
D 20x200	11.60	0.254	0.000	0.254	385	9.030	9.340	9.640	9.990	10.40	10.69

Tabela A2 - Lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a das capacidades.

A.1.1.2 - Relaxação Lagsur

prob	t_med	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	0.473	38.72	9.758	30.18	235	0.022	0.022	0.145	--	--	--
C5x20	0.560	21.37	3.323	18.49	233	0.022	0.022	0.033	--	--	--
C5x25	0.516	14.82	6.290	8.656	167	0.033	0.033	0.033	0.033	0.083	--
C5x30	0.813	17.23	5.523	11.92	261	0.055	0.055	0.055	0.055	--	--
C8x24	0.824	19.43	3.932	15.86	184	0.033	0.033	0.033	0.033	--	--
C8x32	0.901	16.33	5.556	10.95	192	0.055	0.055	0.055	0.066	0.210	--
C8x40	1.066	10.30	4.205	6.156	155	0.077	0.077	0.077	0.077	0.099	--
C8x48	2.286	12.56	6.540	6.092	294	0.143	0.143	0.143	0.143	0.154	--
C10x30	1.088	18.53	5.629	13.16	162	0.099	0.099	0.099	0.110	--	--
C10x40	1.286	10.05	4.599	5.518	147	0.110	0.110	0.110	0.110	0.138	--
C10x50	1.637	9.903	4.959	4.998	163	0.187	0.187	0.187	0.187	0.198	--
C10x60	1.934	5.967	3.040	2.946	161	0.231	0.231	0.231	0.231	0.231	0.264

Tabela A3 - Lagsur para problemas das classes A, B, C e D com solução ótima conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a das capacidades.

prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	0.310	0.291	0.000	0.291	53	0.160	0.160	0.160	0.160	0.160	0.160
A 5x200	0.810	0.229	0.000	0.220	47	0.580	0.580	0.580	0.580	0.580	0.580
A 10x100	0.570	2.311	1.489	0.824	59	0.270	0.270	0.270	0.270	0.270	0.270
A 10x200	1.530	0.480	0.160	0.374	47	1.080	1.080	1.080	1.080	1.080	1.080
A 20x100	1.340	1.316	0.000	1.318	77	0.560	0.560	0.560	0.560	0.560	0.560
A 20x200	3.730	0.998	0.103	0.895	77	2.150	2.150	2.150	2.150	2.150	2.150
B 5x100	1.100	123.7	108.5	17.30	222	0.140	0.140	0.140	0.150	--	--
B 5x200	2.390	3.641	2.117	1.529	176	0.580	0.580	0.580	0.580	0.580	0.580
B 10x100	1.700	7.264	5.828	1.447	229	0.280	0.280	0.280	0.280	0.280	0.330
B 10x200	5.650	3.523	2.701	0.824	306	1.080	1.080	1.080	1.080	1.080	1.080
B 20x100	1.210	12.82	9.134	3.725	61	0.540	0.540	0.540	0.540	0.580	0.870
B 20x200	7.730	4.762	2.585	2.187	225	2.160	2.160	2.160	2.160	2.160	2.160
C 5x100	0.930	2.851	1.134	1.722	183	0.140	0.140	0.140	0.140	0.140	0.150
C 5x200	2.400	8.638	7.068	1.584	173	0.570	0.570	0.570	0.570	0.570	0.570
C 10x100	2.910	7.485	1.327	6.204	326	0.280	0.280	0.280	0.280	0.290	---
C 10x200	4.300	3.863	1.622	2.250	221	1.070	1.070	1.070	1.070	1.070	1.070
C 20x100	1.720	17.05	11.29	5.861	65	0.540	0.540	0.540	0.540	0.580	---
C 20x200	9.730	5.323	2.185	3.154	250	2.150	2.150	2.150	2.150	2.150	2.210
D 5x100	0.350	1.083	0.000	1.084	77	0.140	0.140	0.140	0.140	0.140	0.140
D 5x200	1.370	1.211	0.000	1.213	139	0.550	0.550	0.550	0.550	0.550	0.550
D 10x100	0.790	1.548	0.000	1.550	108	0.270	0.270	0.270	0.270	0.270	0.270
D 10x200	2.620	1.631	0.000	1.634	139	1.080	1.080	1.080	1.080	1.080	1.080
D 20x100	1.850	1.862	0.000	1.866	139	0.550	0.550	0.550	0.550	0.550	0.550
D 20x200	5.340	1.855	0.000	1.858	170	2.170	2.170	2.170	2.170	2.170	2.170

Tabela A4 - Lagsur para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a das capacidades.

A.1.1.3 - Relaxação lagsur versus relaxação lagrangeana

Divisão, campo a campo, dos valores obtidos com o lagsur com a relaxação lagrangeana.

prob	t_med	gap1	gap2	gap3	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	0.673	1.010	1.085	0.988	0.855	0.667	0.222	0.564	--	--	--
C5x20	0.614	1.000	1.000	0.999	0.764	2.000	1.000	0.600	--	--	--
C5x25	0.505	1.548	6.025	1.004	0.551	1.000	1.000	1.000	1.000	0.393	--
C5x30	0.711	1.257	2.578	1.015	0.906	1.250	1.250	1.000	0.071	--	--
C8x24	0.572	1.098	1.833	0.999	0.624	0.600	0.600	0.500	0.367	--	--
C8x32	0.532	1.182	2.333	1.000	0.565	0.833	0.714	0.625	0.546	0.188	--
C8x40	0.527	0.999	0.997	1.000	0.556	0.875	0.778	0.636	0.583	0.474	--
C8x48	0.669	1.117	1.234	1.015	0.905	1.083	1.083	1.083	1.000	0.354	--
C10x30	0.513	1.157	0.918	1.307	0.542	1.500	1.000	0.900	0.556	1.000	--
C10x40	0.513	0.863	1.098	0.731	0.539	0.714	0.667	0.667	0.588	0.177	--
C10x50	0.507	1.208	1.527	0.999	0.514	0.850	0.773	0.708	0.680	0.564	--
C10x60	0.527	0.954	0.915	0.996	0.524	0.778	0.750	0.700	0.656	0.583	0.297

Tabela A5. Lagsur/lagrangeano para problemas da classe C com solução ótima conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a das capacidades.

prob	t	gap1	gap2	gap3	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	1.240	1.366	1.000	1.366	1.559	1.000	1.000	0.942	0.889	0.842	0.800
A 5x200	1.125	1.272	1.000	1.222	1.808	1.055	1.000	0.935	0.879	0.853	0.817
A 10x100	0.905	1.756	2.333	1.214	1.053	0.711	0.675	0.643	0.614	0.600	0.575
A 10x200	0.773	4.615	∝	3.596	0.671	0.771	0.735	0.697	0.659	0.632	0.624
A 20x100	0.882	3.686	1.000	3.572	1.069	0.544	0.523	0.509	0.496	0.479	0.471
A 20x200	0.748	4.730	∝	4.242	0.713	0.586	0.567	0.544	0.528	0.512	0.504
B 5x100	5.00	0.996	1.000	0.964	6.167	1.000	1.000	0.933	0.882	--	--
B 5x200	0.754	1.352	1.800	1.005	0.550	1.094	1.094	1.094	1.094	1.094	1.094
B 10x100	0.680	1.546	0.934	0.985	0.696	1.077	1.077	1.077	1.000	0.875	0.917
B 10x200	0.900	0.998	1.000	0.989	0.874	1.019	1.019	1.019	1.019	0.973	0.931
B 20x100	0.248	1.426	1.571	1.163	0.200	0.885	0.818	0.771	0.720	0.707	0.956
B 20x200	0.513	1.643	3.570	1.003	0.489	1.019	0.952	0.889	0.844	0.794	0.732
C 5x100	0.589	1.005	1.000	1.009	0.466	1.077	1.077	1.077	1.077	1.077	1.071
C 5x200	0.692	1.780	1.903	1.386	0.502	1.076	1.076	1.076	1.076	1.076	1.076
C 10x100	0.957	1.030	1.200	0.999	0.888	1.077	1.077	1.077	1.077	0.935	--
C 10x200	0.773	1.638	15.02	0.998	0.682	1.009	1.009	1.009	1.009	0.964	0.922
C 20x100	0.239	1.790	2.572	1.130	0.167	1.019	1.019	0.964	0.885	0.743	--
C 20x200	0.681	0.963	0.913	1.000	0.592	1.019	1.019	0.973	0.911	0.853	0.819
D 5x100	2.059	1.094	1.000	--	3.348	--	--	--	--	--	--
D 5x200	2.322	0.012	1.000	--	6.044						
D 10x100	0.506	12.90	1.000	12.91	0.548	0.270	0.262	0.252	0.243	0.220	0.205
D 10x200	0.489	38.83	1.000	38.90	0.427	0.267	0.257	0.247	0.238	0.226	0.220
D 20x100	0.562	21.16	1.000	21.20	0.635	0.233	0.225	0.218	0.210	0.201	0.192
D 20x200	0.460	7.303	1.000	7.315	0.442	0.240	0.232	0.225	0.217	0.209	0.203

Tabela A6 - Lagsur/lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a das capacidades.

A.1.2 - Relaxação da restrição de atribuição

A.1.2.1 - Relaxação lagrangeana

prob	t_med	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	1.011	4.294	0.595	3.717	107	0.0879	0.121	0.198	0.341	0.560	0.750
C5x20	1.560	2.611	0.000	2.619	115	0.198	0.253	0.363	0.538	0.802	1.165
C5x25	1.912	1.519	0.000	1.521	110	0.462	0.539	0.604	0.714	0.912	1.231
C5x30	4.648	2.902	0.620	2.291	187	0.429	0.517	0.681	0.923	1.736	2.758
C8x24	4.132	2.301	0.358	1.948	157	0.725	0.802	0.879	1.000	1.362	2.198
C8x32	8.121	3.102	1.321	1.787	207	1.506	1.626	1.791	2.011	2.659	4.429
C8x40	11.04	0.967	0.000	0.968	185	2.725	2.890	3.121	3.429	4.308	6.176
C8x48	24.26	3.504	2.300	1.209	301	3.538	4.110	4.802	5.868	7.549	11.98
C10x30	11.20	3.031	1.120	1.916	241	1.780	1.912	2.110	2.352	3.286	5.736
C10x40	15.60	3.227	1.884	1.348	208	3.593	3.901	4.198	4.626	5.483	8.176
C10x50	26.03	2.007	1.025	0.984	234	5.626	6.055	6.571	7.286	8.615	11.87
C10x60	38.37	1.622	0.830	0.793	252	10.45	10.99	11.73	12.65	14.32	16.59

Tabela A7 - Lagrangeano para problemas das classes A, B, C e D com solução ótima conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a da atribuição.

prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	7.970	0.218	0.000	0.218	226	6.290	6.520	6.790	7.110	7.420	7.610
A 5x200	47.64	0.333	0.000	0.333	470	31.43	33.52	35.92	38.51	41.36	43.08
A 10x100	19.24	0.173	0.000	0.173	285	16.04	16.49	16.93	17.36	17.90	18.19
A 10x200	109.2	0.089	0.000	0.089	544	87.25	89.84	92.65	95.72	99.44	101.7
A 20x100	49.28	0.171	0.000	0.171	394	42.87	43.51	44.14	44.90	45.63	46.12
A 20x200	244.7	88.79	0.414	96.99	600	--	--	--	--	--	--
B 5x100	15.17	23.52	0.000	24.09	398	1.630	1.980	2.800	--	--	--
B 5x200	60.07	1.218	0.000	1.218	600	12.51	13.91	16.03	18.70	23.16	26.58
B 10x100	35.88	0. 570	0. 432	0. 138	575	9.670	10.49	11.62	13.11	15.16	17.29
B 10x200	114.2	3.603	2.917	0. 688	600	50.02	53.61	57.78	62.94	70.99	78.36
B 20x100	70.82	0. 612	0. 208	0. 405	565	30.43	31.54	32.81	34.74	36.99	39.01
B 20x200	244.7	88.79	0.414	96.99	600	--	--	--	--	--	--
C 5x100	15.78	0.622	0.453	0.169	466	2.870	3.320	3.860	4.550	5.650	6.680
C 5x200	60.72	0.697	0.000	0.698	600	8.63	9.65	11.13	13.68	18.21	22.41
C 10x100	40.04	4.154	1.546	2.619	600	9.710	10.50	11.37	12.62	14.89	18.28
C 10x200	112.3	3.017	0.865	2.159	600	48.25	52.14	56.94	62.43	70.84	81.18
C 20x100	78.74	4.663	2.926	1.745	600	28.70	30.15	31.82	34.82	37.70	42.49
C 20x200	246.5	3.319	0.208	3.121	600	158.1	166.0	175.0	187.1	204.7	222.1
D 5x100	67.16	0.079	0.000	0.079	491	65.58	65.89	66.20	66.51	66.80	66.95
D 5x200	1366.	205.1	0.000	258.0	600	--	--	--	--	--	--
D 10x100	75.78	288.9	0.000	406.3	600	--	--	--	--	--	--
D 10x200	340.9	503.0	0.097	1000.	600	--	--	--	--	--	--
D 20x100	103.8	600.8	0.000	1000	600	--	--	--	--	--	--
D 20x200	901.7	661.1	0.000	1950	600	--	--	--	--	--	--

Tabela A8 - Lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a da atribuição.

A.1.2.2 - Relaxação lagsur

Problem	t_med	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	1.681	5.653	1.840	3.840	184	0.330	0.495	0.747	0.934	1.198	1.500
C5x20	2.637	2.911	0.000	2.921	196	0.396	0.495	0.780	1.275	1.692	2.132
C5x25	3.362	1.900	0.349	1.554	185	0.659	0.725	0.835	1.198	2.253	2.879
C5x30	6.121	2.549	0.311	2.246	251	0.758	0.846	1.011	1.516	3.209	4.517
C8x24	6.066	3.062	0.716	2.355	213	0.890	0.978	1.055	2.582	3.418	4.198
C8x32	10.14	2.528	0.793	1.740	258	1.450	1.528	1.659	2.077	5.791	6.934
C8x40	15.77	0.939	0.000	0.940	258	2.384	2.495	2.648	3.088	7.868	10.99
C8x48	27.17	3.687	2.477	1.215	333	3.582	3.802	4.330	5.242	12.42	15.13
C10x30	13.33	2.900	0.839	2.067	280	1.813	1.934	2.131	3.912	7.176	8.868
C10x40	21.09	2.996	1.888	1.112	273	3.231	3.341	3.626	4.066	11.56	12.66
C10x50	29.22	2.110	1.022	1.091	257	4.308	4.593	4.901	5.451	15.40	19.47
C10x60	45.59	1.328	0.554	0.776	293	7.055	7.352	7.780	8.352	18.23	26.47

Tabela A9 - Lagsur para problemas das classes A, B, C e D com solução ótima conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a da atribuição.

Prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A5x100	6.820	0.128	0.000	0.128	222	2.650	2.750	2.850	2.970	3.100	3.250
A5x200	25.54	0.085	0.000	0.085	256	12.39	13.27	14.17	15.37	16.54	17.13
A10x100	16.87	0.413	0.000	0.413	277	6.290	6.470	6.600	6.780	7.070	7.190
A10x200	64.56	0.089	0.000	0.089	326	31.89	32.92	33.92	35.12	36.53	3752
A20x100	31.29	0.784	0.000	0.785	260	16.33	16.57	16.82	17.06	17.42	17.77
A20x200	181.4	0.098	0.000	0.098	452	89.01	90.20	91.87	93.91	96.79	98.42
B5x100	16.75	8.628	0.000	8.703	464	1.040	1.100	1.210	1.390	10.24	--
B5x200	49.35	1.821	0.588	1.236	496	6.020	6.470	7.240	8.350	10.45	12.24
B10x100	23.74	0.199	0.000	0.199	388	4.440	4.810	5.230	5.780	6.840	8.320
B10x200	118.0	0.396	0.108	0.288	600	19.70	21.16	22.63	24.87	28.28	32.08
B20x100	43.67	0.438	0.000	0.438	361	12.13	12.48	12.97	13.69	14.79	15.77
B20x200	222.4	1.797	0.414	1.386	557	56.75	59.42	62.54	66.68	74.02	83.30
C5x100	11.62	0.591	0.453	0.138	357	1.700	1.900	2.110	2.430	2.990	3.510
C5x200	56.49	2.225	1.557	0.669	569	4.560	4.790	5.150	5.640	7.100	16.64
C10x100	34.38	5.435	2.871	2.578	523	4.530	4.820	5.170	5.720	7.480	21.46
C10x200	100.6	1.677	0.108	1.571	518	19.48	20.92	22.88	25.08	29.21	35.06
C20x100	53.56	2.735	1.045	1.694	429	11.63	12.11	12.71	13.68	15.72	19.97
C20x200	229.0	1.926	0.000	1.929	565	55.82	58.52	61.96	66.23	73.32	82.15
D5x100	32.37	0.095	0.000	0.095	249	29.97	30.07	30.17	30.26	30.35	30.41
D5x200	609.9	0.048	0.000	0.048	394	600.6	601.1	601.7	602.3	602.9	603.2
D10x100	40.87	0.094	0.000	0.094	367	36.05	36.17	36.28	36.40	36.52	36.63
D10x200	210.0	0.348	0.000	0.340	600	197.2	198.0	198.7	199.5	200.5	201.3
D20x100	83.17	0.073	0.000	0.073	579	71.77	72.00	72.34	72.57	72.81	73.04
D20x200	517.5	447.9	0.000	811.3	600	--	--	--	--	--	--

Tabela A10 - Lagsur para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a da atribuição.

A.1.2.3 - Relaxação lagsur versus relaxação lagrangeana

Divisão, campo a campo, dos valores obtidos com a relaxação lagsur com a relaxação lagrangeana.

prob	t_med	gap1	gap2	gap3	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	1.663	1.317	3.092	1.033	1.720	3.754	4.091	3.773	2.739	2.139	2.000
C5x20	1.690	1.115	1.000	1.115	1.704	2.000	1.960	2.149	2.470	2.110	1.288
C5x25	1.758	1.251	∝	1.015	1.682	1.426	1.345	1.382	1.678	2.470	2.339
C5x30	1.317	0.878	0.502	0.980	1.342	1.767	1.636	1.485	1.643	1.849	1.638
C8x24	1.468	1.330	2.000	1.209	1.357	1.228	1.219	1.200	2.582	2.510	1.910
C8x32	1.249	0.815	0.600	0.974	1.246	0.962	0.940	0.926	1.032	2.178	1.566
C8x40	1.428	0.971	1.000	0.971	1.395	0.875	0.863	0.849	0.900	1.826	1.780
C8x48	1.118	1.052	1.077	1.005	1.106	1.012	0.925	0.902	0.893	1.645	1.263
C10x30	1.190	0.957	0.749	1.079	1.162	1.019	1.012	1.010	1.663	2.184	1.546
C10x40	1.352	0.928	1.002	0.825	1.313	0.899	0.856	0.864	0.879	2.108	1.548
C10x50	1.123	1.051	0.997	1.109	1.098	0.766	0.758	0.746	0.748	1.787	1.640
C10x60	1.188	0.818	0.667	0.979	1.163	0.675	0.667	0.663	0.660	1.273	1.596

Tabela A11 . Lagsur/lagrangeano para problemas da classe C com solução ótima conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a da atribuição.

prob	t	gap1	gap2	gap3	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	0.856	1.00	1.00	1.00	0.982	0.421	0.422	0.420	0.418	0.418	0.427
A 5x200	0.538	0.255	1.00	0.255	0.545	0.394	0.400	0.395	0.399	0.400	0.398
A 10x100	0.877	2.374	1.00	2.374	0.972	0.392	0.392	0.390	0.391	0.395	0.395
A 10x200	0.591	1.00	1.00	1.00	0.599	0.366	0.366	0.366	0.367	0.367	0.369
A 20x100	0.635	4.585	1.00	4.586	0.660	0.381	0.381	0.381	0.380	0.382	0.385
A 20x200	0.741	0.001	0.000	0.001	0.753	--	--	--	--	--	--
B 5x100	1.104	0.367	1.000	0.361	1.166	0.638	0.556	0.432	--	--	--
B 5x200	0.821	1.495	∝	1.015	0.827	0.481	0.465	0.452	0.447	0.451	0.461
B 10x100	0.662	0.349	0.000	1.442	0.675	0.459	0.459	0.450	0.441	0.451	0.481
B 10x200	1.033	0.110	0.037	0.417	1.000	0.394	0.395	0.392	0.395	0.398	0.409
B 20x100	0.620	0.716	0.000	1.082	0.639	0.399	0.396	0.395	0.394	0.400	0.404
B 20x200	0.909	0.020	1.000	0.014	0.928	--	--	--	--	--	--
C 5x100	0.736	0.950	1.00	0.817	0.766	0.592	0.572	0.547	0.534	0.529	0.525
C 5x200	0.930	3.192	∝	0.959	0.948	0.528	0.496	0.463	0.412	0.390	0.743
C 10x100	0.859	1.308	1.857	0.984	0.872	0.467	0.459	0.455	0.453	0.502	1.174
C 10x200	0.896	0.556	0.125	0.728	0.863	0.404	0.401	0.402	0.402	0.412	0.432
C 20x100	0.680	0.586	1.208	0.971	0.715	0.405	0.401	0.223	0.219	0.417	0.470
C 20x200	0.929	0.580	0.000	0.618	0.942	0.353	0.353	0.354	0.354	0.358	0.370
D 5x100	0.482	1.203	1.000	1.203	0.507	0.457	0.437	0.456	0.455	0.454	0.454
D 5x200	0.447	0.000	1.000	0.000	0.657	--	--	--	--	--	--
D 10x100	0.539	0.000	1.000	0.000	0.612	--	--	--	--	--	--
D 10x200	0.615	0.000	0000	0.000	1.000	--	--	--	--	--	--
D 20x100	0.801	0.000	1.000	0.000	0.965	--	--	--	--	--	--
D 20x200	0.574	0.678	1.000	0.416	1.000	--	--	--	--	--	--

Tabela A12 - Lagsur/lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. A restrição relaxada é a da atribuição.

A.1.3 - Proposta de Jornsten e Nasberg (J e N)

A.1.3.1 - Relaxação lagrangeana conforme a proposta de J e N

prob	t_med	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	1.648	4.959	0.595	4.388	130	0.0769	0.0989	0.165	0.440	0.934	--
C5x20	4.440	4.120	0.000	4.140	253	0.1319	0.2198	0.3077	0.7143	1.681	3.335
C5x25	7.868	2.597	0.000	2.604	306	0.2637	0.3517	0.604	1.297	2.659	4.571
C5x30	9.253	3.407	0.311	3.111	265	0.220	0.297	0.418	0.758	2.506	4.383
C8x24	16.13	3.785	0.358	3.443	413	1.055	1.517	2.462	4.550	8.637	11.28
C8x32	34.26	4.354	0.531	3.842	520	1.593	2.539	4.407	8.769	18.17	25.45
C8x40	50.01	4.533	0.631	3.921	541	3.451	5.143	8.132	14.26	28.63	39.55
C8x48	80.54	5.473	1.943	3.549	526	1.385	2.066	3.483	7.527	30.44	62.14
C10x30	47.68	5.830	0.558	5.304	582	3.220	4.868	8.066	15.19	26.59	43.66
C10x40	74.09	8.403	2.517	5.938	600	7.857	11.95	19.08	32.34	54.98	--
C10x50	97.31	6.495	1.020	5.511	593	8.121	11.73	17.94	30.40	59.32	--
C10x60	134.21	14.289	3.043	11.410	600	22.12	33.56	51.58	84.94	--	--

Tabela A13 - Lagrangeano para problemas das classes A, B, C e D com solução ótima conhecida. Número máximo de iterações é igual a 600. Proposta de Jornsten e Nasberg.

prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	22.37	0.533	0.000	0.534	600	8.330	9.870	11.62	14.17	17.18	19.03
A 5x200	63.65	22.94	0.000	23.46	600	27.97	38.42	51.83	--	--	--
A 10x100	42.93	36.42	0.638	37.13	600	35.49	40.50	--	--	--	--
A 10x200	123.3	75.62	0.000	81.80	600	--	--	--	--	--	--
A 20x100	87.60	120.2	0.000	136.7	600	--	--	--	--	--	--
A 20x200	253.7	155.5	0.414	183.6	600	--	--	--	--	--	--
B 5x100	30.43	57.94	27.70	32.11	600	1.430	2.210	--	--	--	--
B 5x200	64.62	3.927	1.529	2.407	600	6.940	8.610	10.52	13.08	17.49	26.13
B 10x100	69.04	15.95	5.180	10.94	600	9.280	11.76	16.87	28.83	--	--
B 10x200	122.9	18.45	6.483	12.19	600	34.17	42.45	55.02	78.05	--	--
B 20x100	185.0	60.74	7.058	57.15	600	--	--	--	--	--	--
B 20x200	242.6	69.88	5.274	69.46	600	--	--	--	--	--	--
C 5x100	30.17	0.991	0.453	0.539	600	1.390	1.790	2.240	3.310	6.650	11.33
C 5x200	78.48	8.607	5.271	3.364	600	4.260	5.420	7.100	9.430	13.39	22.80
C 10x100	70.08	13.06	3.757	9.425	600	8.450	10.14	13.12	19.91	63.25	--
C 10x200	123.7	16.95	4.650	12.51	600	33.45	40.49	52.88	78.05	--	--
C 20x100	176.5	40.65	8.361	33.66	600	77.99	122.8	--	--	--	--
C 20x200	250.4	60.10	8.220	55.20	600	--	--	--	--	--	--
D 5x100	223.9	30.41	0.000	31.36	600	218.3	220.9	--	--	--	--
D 5x200	617.4	107.4	0.000	120.3	600	--	--	--	--	--	--
D 10x100	643.4	172.0	0.000	207.8	600	--	--	--	--	--	--
D 10x200	1589.	292.5	0.000	413.4	600	--	--	--	--	--	--
D 20x100	312.7	367.9	0.000	582.1	600	--	--	--	--	--	--
D 20x200	538.9	456.0	0.000	838.2	600	--	--	--	--	--	--

Tabela A14 - Lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. Proposta de Jornsten e Nasberg.

prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	23.37	0.219	0.000	0.219	627	8.220	9.810	11.65	14.18	17.30	19.14
A 5x200	123.0	0.101	0.000	0.101	1172	27.95	38.39	51.78	68.78	90.23	101.9
A 10x100	77.83	0.183	0.000	0.183	1080	35.49	40.50	46.64	52.74	60.94	66.08
A 10x200	247.1	30.62	0.000	31.59	1200	182.3	213.3	--	--	--	--
A 20x100	177.4	50.92	0.000	53.65	1200	--	--	--	--	--	--
A 20x200	496.2	96.74	0.310	106.8	1200	--	--	--	--	--	--
B 5x100	35.45	57.60	27.70	32.06	660	1.470	2.31	--	--	--	--
B 5x200	88.58	3.449	1.529	1.926	763	6.930	8.610	10.52	13.10	17.51	26.12
B 10x100	167.3	6.880	2.374	4.537	1200	9.29	11.76	16.87	28.81	74.75	154.9
B 10x200	373.5	8.097	4.214	3.914	1200	34.17	42.45	55.02	78.05	144.6	312.5
B 20x100	553.2	38.38	5.813	33.87	1051	225.5	378.0	--	--	--	--
B 20x200	556.0	39.18	3.00	37.66	1200	385.5	495.5	--	--	--	--
C 5x100	24.61	782.0	453.0	328.0	664	0.973	1.253	1.561	2.310	4.648	7.931
C 5x200	79.86	8.607	5.271	3.364	607	4.26	5.420	7.10	9.440	13.39	22.77
C 10x100	185.4	5.30	0.000	5.325	1200	8.450	10.14	13.12	19.91	63.25	--
C 10x200	441.4	7.235	1.730	5.545	1200	33.45	40.49	52.88	78.05	167.5	--
C 20x100	473.6	23.03	6.689	16.72	1200	77.99	122.8	351.7	--	--	--
C 20x200	519.5	34.14	5.098	30.07	1200	263.8	339.1	493.3	--	--	--
D 5x100	130.0	0.108	0.000	0.108	902	119.2	120.7	122.4	124.3	126.4	127.4
D 5x200	762.8	27.69	0.000	28.48	1200	727.9	742.5	759.0	--	--	--
D 10x100	342.0	51.87	0.000	54.71	1200	--	--	--	--	--	--
D 10x200	911.9	160.6	0.000	0.191	1200	--	--	--	--	--	--
D 20x100	250.0	192.0	0.000	237.7	1200	--	--	--	--	--	--
D 20x200	1137.	345.7	0.000	528.5	1200	--	--	--	--	--	--

Tabela A15 - Lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 1200. Proposta de Jornsten e Nasberg.

A.1.3.2 - Relaxação lagsur para a proposta de J e N

prob	t_med	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	2.923	4.774	0.000	4.804	254	0.4835	0.6703	0.8462	1.088	1.3956	2.800
C5x20	4.560	4.038	0.000	4.059	299	0.330	0.857	1.516	2.3956	2.8132	2.900
C5x25	4.912	1.674	0.348	1.328	237	0.4176	0.6373	1.0879	2.3626	3.0769	3.4835
C5x30	5.901	3.722	0.311	3.427	213	0.3626	0.5275	0.8901	2.7033	3.5165	3.9300
C8x24	9.143	2.914	0.358	2.565	300	1.033	1.407	2.989	4.681	5.956	7.110
C8x32	21.65	2.069	0.268	1.805	444	1.286	1.857	3.5495	8.571	8.967	10.74
C8x40	28.55	1.554	0.000	1.557	405	2.132	2.857	4.297	10.50	14.37	16.13
C8x48	48.77	2.893	1.237	1.661	412	1.802	2.186	2.956	8.473	16.84	23.73
C10x30	23.02	3.716	0.840	2.887	403	2.077	3.473	7.461	10.10	12.21	15.51
C10x40	40.62	2.348	0.837	1.515	468	3.593	5.363	9.242	19.55	22.30	28.41
C10x50	60.00	1.431	0.000	1.433	447	4.066	5.253	8.088	15.43	25.00	34.85
C10x60	94.09	1.867	0.277	1.594	558	7.769	10.73	17.20	29.46	56.92	61.21

Tabela A16 - Lagsur para problemas das classes A, B, C e D com solução ótima conhecida. Número máximo de iterações é igual a 600. Proposta de Jornsten e Nasberg.

prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	11.32	0.214	0.000	0.214	325	3.010	3.650	4.360	5.390	6.560	7.740
A 5x200	53.71	0.072	0.000	0.072	505	9.490	13.03	17.59	24.32	33.09	37.97
A 10x100	29.39	0.175	0.000	0.175	441	11.35	13.27	15.36	17.66	20.98	24.08
A 10x200	124.5	6.123	0.000	6.161	600	55.19	65.75	78.65	95.00	114.5	--
A 20x100	86.45	8.760	0.000	8.837	600	53.92	59.99	66.80	74.89	84.99	--
A 20x200	243.1	49.11	0.310	51.32	600	239.1	--	--	--	--	-
B 5x100	19.95	55.31	24.50	32.14	504	0.910	1.170	--	--	--	--
B 5x200	42.75	3.259	1.294	1.971	390	3.410	3.820	4.350	5.170	6.420	11.29
B 10x100	44.73	4.688	1.727	2.975	600	2.980	3.760	5.180	8.290	19.60	40.98
B 10x200	122.5	6.406	3.241	3.185	600	10.60	13.06	16.44	23.89	47.29	84.01
B 20x100	149.2	19.17	4.360	15.10	600	35.62	51.10	72.89	116.9	--	--
B 20x200	244.8	24.63	2.172	23.02	600	94.53	125.2	177.6	--	--	--
C 5x100	12.91	0.311	0.000	0.311	346	0.860	0.960	1.140	1.480	3.040	5.240
C 5x200	55.23	8.134	4.912	3.248	487	2.740	2.940	3.270	3.910	5.180	14.17
C 10x100	56.66	4.058	00000	4.074	600	3.090	3.620	4.400	6.870	20.75	45.97
C 10x200	121.3	5.027	0.757	4.292	600	11.00	12.88	15.92	24.20	51.21	102.9
C 20x100	127.2	14.52	3.344	11.34	600	17.30	24.73	36.36	60.48	--	--
C 20x200	258.9	22.22	3.850	18.79	600	72.90	97.33	141.2	235.6	--	--
D 5x100	59.67	0.068	0.000	0.068	409	50.66	51.52	52.58	53.64	54.94	55.68
D 5x200	620.0	0.048	0.000	0.048	600	608.8	610.3	611.8	613.7	615.8	616.7
D 10x100	192.8	1.684	0.000	1.687	600	175.8	178.5	181.2	184.3	187.4	189.3
D 10x200	725.7	77.18	0.000	83.63	600	--	--	--	--	--	--
D 20x100	155.3	94.62	0.000	104.5	600	--	--	--	--	--	--
D 20x200	1703	228.5	0.000	296.2	600	--	--	--	--	--	--

Tabela A17 - Lagsur para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. Proposta de Jornsten e Nasberg.

prob	t	gap1 (10^{-3})	gap2 (10^{-3})	gap3 (10^{-3})	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	11.06	0.214	0.000	0.214	325	2.960	3.580	4.260	5.270	6.420	7.570
A 5x200	52.78	0.072	0.000	0.072	505	9.30	12.80	17.27	23.87	32.50	37.30
A 10x100	29.28	0.175	0.000	0.175	441	11.32	13.22	15.31	17.60	20.92	24.00
A 10x200	171.4	0.139	0.000	0.139	854	55.16	65.69	78.57	94.99	114.5	128.6
A 20x100	115.7	0.215	0.000	0.215	851	54.04	60.11	69.94	75.03	85.16	96.01
A 20x200	475.7	3.408	0.000	3.420	1200	239.4	274.2	313.3	358.8	417.6	457.3
B 5x100	24.65	57.6	24.9	34.7	518	0.50	0.64	--	--	--	--
B 5x200	74.05	3.104	1.294	1.816	589	3.37	3.77	4.29	5.09	6.33	11.11
B 10x100	70.04	1.774	0.432	1.345	779	2.97	3.76	5.17	8.26	19.52	40. 86
B 10x200	258.5	1.229	0.648	0.582	985	10.57	13.01	16.38	23.83	47.26	83.95
B 20x100	430.7	1.590	0.415	1.177	1200	35.7	51.17	73.05	117.1	197.2	356.0
B 20x200	556.3	10.34	0.827	9.615	1200	94.7	125.4	177.9	288.8	535.4	--
C 5x100	12.9	0.311	0.000	0.311	346	0.85	0.95	1.13	1.47	3.03	5.23
C 5x200	54.31	8.134	4.912	3.248	487	2.69	2.89	3.21	3.84	5.090	13.96
C 10x100	63.15	3.084	0.000	3.093	652	3.02	3.500	4.39	6.32	18.08	38.43
C 10x200	316.0	3.240	0.433	2.817	857	14.89	17.45	21.56	32.83	69.48	140.0
C 20x100	434.1	4.435	0.418	4.035	1100	25.35	36.23	53.24	88.46	225.5	301.3
C 20x200	530.0	9.571	1.769	7.878	1200	61.83	81.54	118.3	197.2	411.0	--
D 5x100	58.69	0.068	0.000	0.068	409	49.89	50.74	51.77	52.81	54.08	54.80
D 5x200	391.0	0.048	0.000	0.048	610	383.9	384.7	385.6	386.8	388.1	388.7
D 10x100	196.9	0.092	0.000	0.092	691	175.6	178.3	180.9	184.0	187.1	189.0
D 10x200	856.4	0.047	0.000	0.047	1147	756.3	769.5	784.0	801.4	819.5	828.3
D 20x100	245.6	0.070	0.000	0.070	1193	183.7	191.9	200.8	210.9	221.1	227.0
D 20x200	1188.	84.81	0.000	92.67	1200	--	--	--	--	--	--

Tabela A18 - Lagsur para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 1200. Proposta de Jornsten e Nasberg.

A.1.3.3 - Relaxação lagsur versus relaxação lagrangeana para a proposta de Jornsten e Nasberg

Divisão, campo a campo, dos valores obtidos com o lagsur com a relaxação lagrangeana.

prob	t_med	gap1	gap2	gap3	n_iter	5%	4%	3%	2%	1%	0.5%
C5x15	1.774	0.962	0.000	1.094	1.954	6.287	6.778	5.129	2.473	1.494	--
C5x20	1.027	0.980	1.000	0.980	1.182	2.502	3.899	4.927	3.354	1.674	0.870
C5x25	0.624	0.645	∝	0.510	0.775	1.584	1.812	1.801	1.822	1.157	0.762
C5x30	0.638	1.093	1.000	1.102	0.804	1.648	1.776	2.129	3.566	1.403	0.900
C8x24	0.567	0.770	1.000	0.745	0.726	0.980	0.928	1.214	1.029	0.699	0.630
C8x32	0.632	0.475	0.505	0.470	0.854	0.807	0.731	0.805	0.977	0.494	0.422
C8x40	0.571	0.343	0.000	0.397	0.749	0.618	0.556	0.528	0.736	0.502	0.408
C8x48	0.606	0.529	0.637	0.468	0.783	1.301	1.058	0.849	1.126	0.553	0.382
C10x30	0.483	0.637	1.505	0.544	0.692	0.645	0.713	0.925	0.665	0.459	0.355
C10x40	0.548	0.279	0.333	0.255	0.780	0.457	0.449	0.484	0.605	0.406	--
C10x50	0.617	0.220	0.000	0.260	0.754	0.501	0.446	0.451	0.508	0.421	34.85
C10x60	0.701	0.131	0.091	0.140	0.930	0.351	0.320	0.334	0.347	56.92	61.21

Tabela A19 - Lagsur/lagrangeano para problemas da classe C com solução ótima conhecida. Número máximo de iterações é igual a 600. Proposta de Jornsten e Nasberg.

prob	t	gap1	gap2	gap3	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	0.506	0.402	1.000	0.401	0.542	0.361	0.370	0.375	0.380	0.381	0.407
A 5x200	0.844	0.003	1.000	0.003	0.842	0.339	0.339	0.339	--	--	--
A 10x100	0.685	0.005	0.000	0.005	0.735	0.320	0.328	--	--	--	--
A 10x200	1.010	0.081	1.000	0.075	1.000	--	--	--	--	--	--
A 20x100	0.987	0.073	1.000	0.075	1.000	--	--	--	--	--	--
A 20x200	0.958	0.316	0.749	0.280	1.000	--	--	--	--	--	--
B 5x100	0.656	0.955	0.885	1.001	0.840	0.636	0.840	--	--	--	--
B 5x200	0.662	0.830	0.850	0.819	0.650	0.368	0.325	0.258	0.179	--	--
B 10x100	0.648	0.294	0.333	0.272	1.000	0.321	0.320	0.307	0.288	--	--
B 10x200	0.997	0.347	0.499	0.261	1.000	0.310	0.308	0.299	0.306	--	--
B 20x100	0.807	0.316	0.618	0.264	1.000	--	--	--	--	--	--
B 20x200	1.009	0.353	0.412	0.331	1.000	--	--	--	--	--	--
C 5x100	0.428	0.314	0.000	0.577	0.577	0.619	0.536	0.509	0.447	0.457	0.463
C 5x200	0.704	0.945	0.932	0.966	0.812	0.643	0.542	0.461	0.415	0.387	0.622
C 10x100	0.809	0.310	0.000	0.432	1.000	0.366	0.357	0.335	0.345	0.328	--
C 10x200	0.981	0.297	0.163	0.343	1.000	0.329	0.318	0.301	0.310	--	--
C 20x100	0.721	0.357	0.387	0.337	1.000	0.222	0.201	--	--	--	--
C 20x200	1.034	0.370	0.468	0.340	1.000	--	--	--	--	--	--
D 5x100	0.267	0.002	1.000	0.002	0.682	0.232	0.233	--	--	--	--
D 5x200	3.321	0.001	1.000	0.001	1.000	--	--	--	--	--	--
D 10x100	0.300	0.010	1.000	0.008	1.000	--	--	--	--	--	--
D 10x200	0.457	0.264	1.000	0.202	1.000	--	--	--	--	--	--
D 20x100	0.497	0.257	1.000	0.180	1.000	--	--	--	--	--	--
D 20x200	3.160	0.501	1.000	0.353	1.000	--	--	--	--	--	--

Tabela A20 - Lagsur/lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 600. Proposta de Jornsten e Nasberg.

A tabela acima foi construída levando-se em consideração o limite máximo de 600 iterações. Entretanto, observa-se que, dos testes feitos, cujos resultados estão no apêndice A, que ainda é possível alcançar mais resultados, bastando para isso aumentar o número máximo de iterações para 1200. Os resultados obtidos estão a seguir.

prob	t	gap1	gap2	gap3	n_iter	5%	4%	3%	2%	1%	0.5%
A 5x100	0.473	0.977	1.00	0.077	0.002	0.360	0.365	0.366	0.372	0.371	0.396
A 5x200	0.429	0.713	1.00	0.713	0.431	0.333	0.333	0.334	0.347	0.360	0.366
A 10x100	0.376	0.956	1.00	0.956	0.408	0.319	0.326	0.328	0.334	0.343	0.363
A 10x200	0.694	0.005	1.00	0.004	0.712	0.303	0.308	--	--	--	--
A 20x100	0.652	0.004	1.00	0.004	0.709	--	--	--	--	--	--
A 20x200	0.959	0.035	0.00	0.032	1.00	--	--	--	--	--	--
B 5x100	0.695	1.00	0.899	1.082	0.785	0.340	0.277	--	--	--	--
B 5x200	0.841	0.899	0.846	0.943	0.772	0.486	0.437	0.408	0.389	0.362	0.425
B 10x100	0.419	0.258	0.182	0.297	0.649	0.320	0.320	0.307	0.287	0.261	0.264
B 10x200	0.692	0.152	0.154	0.149	0.821	0.309	0.306	0.300	0.305	0.327	0.269
B 20x100	0.779	0.041	0.071	0.035	1.00	0.158	0.135	--	--	--	--
B 20x200	1.001	0.264	0.276	0.255	1.418	0.246	0.253	--	--	--	--
C 5x100	0.524	0.001	0.000	0.001	0.521	0.874	0.758	0.724	0.636	0.625	0.659
C 5x200	0.680	0.945	0.932	0.966	0.802	0.632	0.533	0.452	0.407	0.380	0.613
C 10x100	0.341	0.581	1.00	0.581	0.543	0.357	0.345	0.335	0.317	0.286	- -
C 10x200	0.716	0.448	0.250	0.508	0.714	0.445	0.431	0.408	0.421	0.415	--
C 20x100	0.917	0.193	0.063	0.241	0.917	0.325	0.295	0.151	--	--	--
C 20x200	1.020	0.280	0.347	0.262	1.00	0.234	0.240	0.240	--	--	--
D 5x100	0.452	0.630	1.00	0.630	0.453	0.419	0.420	0.423	0.425	0.428	0.430
D 5x200	0.192	0.002	1.00	0.002	0.508	0.527	0.518	0.508	--	--	--
D 10x100	0.576	0.002	1.00	0.002	0.576	--	--	--	--	--	--
D 10x200	0.939	0.001	1.00	0.246	0.956	--	--	--	--	--	--
D 20x100	0.982	0.001	1.00	0.001	0.994	--	--	--	--	--	--
D 20x200	1.045	0.245	1.00	0.175	1.00	--	--	--	--	--	--

Tabela A21 - Lagsur/lagrangeano para problemas das classes A, B, C e D com solução ótima não conhecida. Número máximo de iterações é igual a 1200. Proposta de Jornsten e Nasberg.

A.2 -Resultados obtidos com o (PCVS)

Para verificar o comportamento da relaxação lagsur e também da relaxação lagrangeana, foram usados problemas testes disponíveis na *Internet* no endereço http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/tsp (TSPLIB). Estes problemas testes têm solução ótima conhecida, a qual vem junto com os dados dos problemas. Os resultados estão dispostos em uma tabela para uma melhor visualização. Em cada tabela, tem-se os seguintes campos:

Prob - contém o número de cidades que o problema teste possui.

t - contém o tempo, em segundos, de execução para se resolver o dual lagrangeano/lagsur

n_iter - contém o número de iterações para que o dual lagrangeano/lagsur venha a convergir

10%, 5%,.....,0.1% - contém o tempo necessário para o dual lagrangeano/lagsur venha a atingir a porcentagem de 10%, 5%,.....,0.1% de diferença em relação à solução ótima.

gap1, gap2, gap3 - são definidos como:

gap1 = (solução viável - relaxação)/solução ótima

gap2 = (solução ótima - relaxação)/solução ótima

gap3 = (solução viável - solução ótima)/solução ótima

Cada problema foi resolvido usando-se o método subgradientes. O número máximo de iterações foi 3000. A implementação foi em linguagem C. Os problemas forma executados em estação Sun Sparc Ultra, 128 Mbytes de memória, 167 Mhz.

A.2.1 - Resultados obtidos com a relaxação lagrangeana

Prob	t	gap1	gap2	gap3	10%	5%	4%
16	2.0	0.00023	0.00023	0.00000	1.0	1.0	1.0
22	9.380	0.000123	0.000123	0.00000	2.940	4.130	4.370
48	23.0	0.008141	0.002355	0.005834	1.00	1.00	2.00
52	6.0	0.002132	0.002132	0.00000	1.00	2.00	2.00
100	52.00	0.044380	0.018157	0.027441	0.00	6.00	9.00
225	675.0	0.090032	0.039181	0.055882	2.00	67.00	495.00
442	6469.0	0.06719	0.007115	0.064378	2.0	527.0	754.0
1002	52420.0	0.030597	0.030597	0.00000	6853.0	27475.0	36714.0
1291	56066.0	0.02384	0.023842	0.00000	14.00	420.0	2376.0
1304	42816.0	0.040637	0.040637	0.00000	4234.90	28094.30	--
1379	38018.0	0.015077	0.015077	0.00000	9.90	1918.60	2826.10
1655	128326.0	0.02204	0.02204	0.00000	40.180	6405.25	12701.90
1748	64421.0	0.040159	0.040159	0.00000	9739.0	48413.0	--
1889	87629.0	0.049982	0.049982	0.00000	10786.0	87568.0	--
2152	99230.0	0.012201	0.012201	0.00000	21.40000	21.40000	2900.100

Tabela A22 - Lagrangeano para instâncias do *(PCVS)* com solução ótima conhecida.

prob	n_iter	3%	2%	1%	0.5%	0.4%	0.3%	0.2%	0.1%
16	289	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
22	487	4.640	4.870	6.340	7.710	7.980	8.40	8.710	9.10
48	1431	4.0	7.0	11.0	15.0	17.0	19.0	--	--
52	282	2.00	3.00	4.00	5.00	5.00	5.00	--	--
100	714	15.00	27.00	--	--	--	--	--	--
225	950	--	--	--	--	--	--	--	--
442	3000	1119.0	1810.0	4054.0	--	--	--	--	--
1002	3000	--	--	--	--	--	--	--	--
1291	3000	13431.	--	--	--	--	--	--	--
1304	3000	--	--	--	--	--	--	--	--
1379	2986	4412.8	9465.7	--	--	--	--	--	--
1655	3000	29368.	--	--	--	--	--	--	--
1748	3000	--	--	--	--	--	--	--	--
1889	3000	--	--	--	--	--	--	--	--
2152	3000	11413.	31334.	--	--	--	--	--	--

Tabela A23 - Lagrangeano para instâncias do *(PCVS)* com solução ótima conhecida.

A.2.2 - Resultados obtidos com a relaxação lagsur

Prob	t	gap1	gap2	gap3	10%	5%	4%
16	1.10	0.000233	0.000233	0.0000	0.17	0.19	0.20
22	7.0	0.006471	0.000096	0.006417	1.010	1.160	1.20
48	8.00	0.009697	0.002988	0.006775	0.15	1.00	1.00
52	6.50	0.002121	0.002121	0.000000	0.330	1.00	1.00
100	28.00	0.050297	0.021871	0.029931	0.56	3.00	3.00
225	652.0	0.098934	0.039154	0.066343	2.28	83.0	392.0
442	997.0	0.084628	0.009726	0.081827	1.950	82.00	92.00
1002	17856.0	0.011068	0.011068	0.00000	455.50	869.50	1054.80
1291	15384.0	0.02188	0.02188	0.00000	14.00	291.0	736.0
1304	15033.0	0.018360	0.018360	0.00000	626.0	1511.0	2111.0
1379	12249.0	0.014109	0.014109	0.00000	9.15	315.0	351.0
1655	128325.0	0.019877	0.019877	0.00000	40.180	1251.0	1600.0
1748	64419.0	0.014932	0.014932	0.00000	785.0	1802.0	2285.0
1889	87643.0	0.017504	0.017504	0.00000	675.80	2275.40	3785.90
2152	99222.0	0.009182	0.009182	0.00000	25.7700	25.7700	1106.0

Tabela A24 - Lagsur para instâncias do *(PCVS)* com solução ótima conhecida.

prob	n_iter	3%	2%	1%	0.5%	0.4%	0.3%	0.2%	0.1%
16	264	0.22	0.89	0.93	1.0	1.0	1.0	1.0	1.03
22	372	1.28	1.41	1.72	2.02	2.36	2.52	3.94	4.67
48	521	1.00	3.00	6.00	7.00	7.00	8.00	--	--
52	309	1.00	4.00	5.00	5.00	5.00	5.00	--	--
100	373	4.00	14.00	--	--	--	--	--	--
225	882	--	--	--	--	--	--	--	--
442	506	110.00	152.00	997.00	--	--	--	--	--
1002	905	1428.4	2514.5	--	--	--	--	--	--
1291	618	3230.0	--	--	--	--	--	--	--
1304	1057	3678.0	9060.0	--	--	--	--	--	--
1379	962	440.0	3147.0	--	--	--	--	--	--
1655	3000	3029.0	42846.	--	--	--	--	--	--
1748	3000	3098.0	6716.0	--	--	--	--	--	--
1889	3000	9080.7	37393.	--	--	--	--	--	--
2152	3000	1805.0	3648.0	21829.	--	--	--	--	--

Tabela A25 - Lagsur para instâncias do *(PCVS)* com solução ótima conhecida.

A.2.3 - Resultados obtidos com a relaxação lagsur x lagrangeana

Prob	t	gap1	gap2	gap3	10%	5%	4%
16	0.550	1.00	1.00	1.00	0.170	0.190	0.200
22	0.748	52.600	0.78	indeterm	0.343	0.280	0.275
48	0.347	1.191	1.268	1.161	0.15	1.00	0.50
52	1.083	0.999	0.999	1.00	0.33	0.500	0.50
100	0.583	1.133	1.204	1.090	Indeterm	0.500	0.333
225	0.925	1.098	0.999	1.187	1.140	1.238	0.792
442	0.154	1.259	1.366	1.271	0.975	0.155	0.122
1002	0.340	0.361	0.361	0.00000	0.06646	0.0316	0.0287
1291	0.274	0.917	0.917	0.00000	1.00	0.692	0.3098
1304	0.351	0.451	0.451	0.00000	0.1478	0.0537	--
1379	0.322	0.935	0.935	0.00000	0.924	0.164	0.124
1655	1.00	0.901	0.901	0.00000	1.00	0.195	0.126
1748	1.00	0.371	0.371	0.00000	0.0806	0.0372	--
1889	1.00	0.350	0.350	0.00000	0.06265	0.0260	--
2152	1.00	0.7525	0.7525	0.00000	1.204	1.204	0.381

Tabela A26 - Lagsur/lagrangeano para instâncias do *(PCVS)* com solução ótima conhecida.

prob	n_iter	3%	2%	1%	0.5%	0.4%	0.3%	0.2%	0.1%
16	0.913	0.22	0.890	0.930	1.00	1.00	1.00	1.00	0.515
22	0.763	0.276	0.295	0.271	0.262	0.296	0.300	0.452	0.513
48	0.364	0.250	0.429	0.546	0.467	0.412	0.421	--	--
52	1.095	0.500	1.333	1.250	1.00	1.00	1.00	--	--
100	0.522	0.267	0.519	--	--	--	--	--	--
225	0.928	--	--	--	--	--	--	--	--
442	0.168	0.0983	0.084	0.246	--	--	--	--	--
1002	0.301	--	--	--	--	--	--	--	--
1291	0.206	0.2405	--	--	--	--	--	--	--
1304	0.352	--	--	--	--	--	--	--	--
1379	0.320	0.0997	0.333	--	--	--	--	--	--
1655	1.00	0.103	--	--	--	--	--	--	--
1748	1.00	--	--	--	--	--	--	--	--
1889	1.00	--	--	--	--	--	--	--	--
2152	1.00	0.158	0.116	--	--	--	--	--	--

Tabela A27 - Lagsur/lagrangeano para instâncias do *(PCVS)* com solução ótima conhecida.