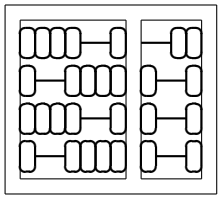

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

MO824 - Programação Inteira e Combinatória

MÉTODO DO SUBGRADIENTE PARA A RESOLUÇÃO DA RELAXAÇÃO LAGRANGEANA DO PROBLEMA DO CAIXEIRO VIAJANTE SIMÉTRICO

Tiago Agostinho de Almeida

RA 025625

Professor:
Dr. Cid. C. Souza

Campinas, 16 de outubro de 2006.

Sumário

1 Introdução

O problema do Caixeiro Viajante é certamente um dos mais estudados na literatura de Pesquisa Operacional. Muitos artigos têm sido publicados sobre o assunto e o problema permanece interessante e desafiador ainda nos dias atuais. A interpretação mais usual ao problema busca por um caminho mais curto para um caixeiro viajante em um determinado número de cidades ou clientes. Os clientes devem ser visitados exatamente uma vez e o caixeiro deve retornar a sua cidade de origem.

A relaxação Lagrangeana é uma técnica bem conhecida e usada freqüentemente na obtenção de limitantes para problemas de Otimização Combinatória [1, 5, 6]. Held e Karp [3] aplicaram com sucesso a relaxação Lagrangeana ao Problema do Caixeiro Viajante (PCV) no início da década de 70. O limite da relaxação aproxima o limite conhecido atualmente como HK (Held e Karp), um limite muito bom (menor que 1% do ótimo) para uma grande classe de problemas simétricos [2]. Jonhson *et al.* [2] comentam que o limite HK exato foi conseguido para instâncias com até 33.810 cidades, usando um programa especialmente adaptado. Para instâncias ainda maiores, é aplicado o método do subgradiente, proposto originalmente nos trabalhos de Held e Karp, mas melhorados por inúmeros recursos. O Método do Subgradiente é uma generalização do conceito do método do gradiente, tratando-se de um método iterativo para obtenção do máximo de uma função. Como para algumas instâncias grandes ainda não são conhecidas as soluções ótimas, a comparação de resultados de heurísticas com o limite HK tornou-se uma prática comum.

Neste trabalho, é estudado o Método do Subgradiente para resolver o problema obtido pela relaxação Lagrangeana do Problema do Caixeiro Viajante Simétrico. Com esse método pretende-se obter limitantes inferiores e por meio de uma heurística encontrar limitantes superiores, de tal forma que, iterativamente, enquanto um vai aumentando, o outro irá reduzindo até se aproximarem da solução ótima do problema. Nas Seções 2, 3 e 4 são apresentados os principais tópicos que compõem a base teórica necessária para a compreensão do método. As referências [5] e [6] foram as principais fontes utilizadas para a elaboração dessas seções. Na Seção 5 é explicado o PCV e as adaptações necessárias para o método do subgradiente. Na Seção 6 é apresentado os métodos utilizados para encontrar limitantes para o PCV. Nas Seções 7 e 8 é explicada uma heurística Lagrangeana e um método para fixação de variáveis, respectivamente. Na Seção 9 são apresentados os resultados experimentais obtidos e na Seção 10 são feitas as conclusões sobre o trabalho.

2 Limitantes e Condições de Otimalidade

Considere o seguinte problema de Programação Inteira (PI):

$$z = max\ \{c^T x : x \in X \subseteq Z^n\}$$

sendo x^* uma solução candidata, como é possível provar que ela é uma solução ótima? Em outras palavras, procuramos condições de otimalidade que provêm uma condição de parada para algoritmos.

Um método bastante simples, consiste em encontrar um limite inferior $\underline{z} \leq z$ e um limite superior $\overline{z} \geq z$, tal que se $\underline{z} = \overline{z}$ então z é ótimo. Tipicamente, algoritmos produzem duas seqüencias de limitantes:

- uma seqüencia de limitantes superiores: $\overline{z_1} > \overline{z_2} > \ldots \overline{z_t} \geq z$
- uma seqüencia de limitantes superiores: $\underline{z_1} < \underline{z_2} < \ldots \underline{z_k} \leq z$

Assim, dado um $\epsilon \geq 0$, pode-se definir um critério de parada como $|\overline{z_t} - \underline{z_k}| \leq \epsilon$. Fazendo dessa maneira, no momento em que o algoritmo termina, tem-se certeza que a solução candidata z é no máximo ϵ unidades inferior à solução ótima. A habilidade de estimar a qualidade de uma solução é de fundamental importância para otimização.

2.1 Limitante Primal

Encontrar uma solução factível pode ser uma tarefa fácil ou árdua, dependendo intrinsicamente do problema e da instância particular. Qualquer solução viável x, $x \in X$, induz um limitante inferior (limite primal para o caso de maximização) já que $c^T x \leq c^T x^* = z^*$. No caso de problemas NP-Completos, como o de encontrar um circuito Hamiltoniano em um grafo, a busca de uma solução factível se resulta a resolver o problema enquanto, por outro lado, encontrar uma rota factível do problema do caixeiro viajante é fácil, estando a dificuldade em encontrar a rota mais curta.

2.2 Limitante Dual

Um método para encontrar limites superiores (para o caso de maximização) faz uso de uma relaxação do problema original, ou seja, o problema a ser resolvido é "transformado" em um problema mais simples cuja solução ótima não é inferior à solução ótima do problema original.

Definição 1 *Um problema (RP)* $z^R = max\ \{f(x) : x \in T \subseteq \mathbb{R}^n\}$ *é uma relaxação do problema (PI)* $z = max\ \{c(x) : x \in X \subseteq \mathbb{R}^n\}$ *se:*

- $X \subseteq T$
- $f(x) \geq c(x)$ *para todo* $x \in X$.

Proposição 1 *Se (RP) é uma relaxação de (PI) então* $z^R \geq z$.

3 Relaxações

Existem diversos tipos de relaxações para obtenção de limitantes duais, entre elas, as mais comuns são: Relaxação baseada em Programação Linear (PL), Relaxação Combinatória e Relaxação Lagrangeana.

3.1 Relaxação Linear

Definição 2 *Para o problema de programação inteira* $max\ \{c^T x : x \in P \cap Z^n\}$ *com formulação* $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ *a relaxação linear é dada por* $z^{PL} = max\ \{c^T x : x \in P\}$.

Proposição 2 *Sejam* P_1 *e* P_2 *duas formulações para o problema inteiro* $z = max\ \{c^T x : x \in X \cap Z^n\}$ *sendo* P_1 *mais "apertada" do que* P_2 *(*$P_1 \subset P_2$*). Se* $z_i^{PL} = max\ \{c^T x : x \in P_i\}$ *para* $i = 1, 2$*, então* $z_1^{PL} \leq z_2^{PL}$.

Proposição 3

- *Se a relaxação RP é infactível, então o problema original PI é infactível*
- *Seja* x^* *uma solução ótima para RP. Se* $x^* \in X \cap Z^n$ *e* $f(x^*) = c(x^*)$*, então* x^* *é uma solução ótima para PI.*

3.2 Relaxação Combinatória

Sempre que a relaxação é um problema de otimização combinatória,dizemos que ela é uma relaxação combinatória. Por exemplo, para o problema do caixeiro viajante (Seção 5) se eliminarmos as restrições de subciclos, obtém-se um problema de alocação. Note que o problema de alocação pode ser eficientemente resolvido fazendo uso de algoritmos para fluxo em redes de custo mínimo.

3.3 Relaxação Lagrangeana

Primeiramente, considere o problema de programação inteira na forma abaixo:

$$(PI)\ \ z = max\ c^T x$$

$$s.a : Ax \leq b$$

$$x \in X \subseteq Z^n$$

Como visto na seção anterior, uma relaxação pode ser obtida pela eliminação de um conjunto de restrições do problema. Ao invés de eliminar essas restrições, a relaxação Lagrangeana as incorpora na função objetivo como segue:

$$(RL)\ \ z = max\ c^T x + u^T(b - Ax)$$

$$s.a : x \in X$$

$$u \geq 0$$

Proposição 4 *Seja* $z(u) = max\ \{c^T x + u^T(b - Ax) : x \in X\}$*. Então* $z(u) \geq z$ *para todo* $u \geq 0$.

De acordo com a Proposição 4, a solução ótima do problema RL para qualquer $u \geq 0$ induz um a limite superior $z(u) \geq z^*$, onde z^* é o valor da solução ótima de PI. É fácil de verificar que $z(u) \geq c^T x$, para todo $x \in \Omega = \{x : x \in X \text{ e } Ax \leq b\} : c^T x \leq c^T x + u^T(b - Ax)$, uma vez que $b - Ax \geq 0$ para x factível e $u \geq 0$; já que as soluções factíveis de PI também satisfazem parte do espaço de soluções factíveis de RL, verifica-se que $z(u) \geq z^*$. Uma vez que $z(u)$ é um limite superior de z^* naturalmente deseja-se minimizá-lo, dando origem ao problema Dual Lagrangeano:

$$LD: \quad min_{u \geq 0}\ z(u) \cong min_{u \geq 0}\ \{max\ c^T x + u^T(b - Ax)\}$$

$$s.a:\ x \in X$$

Um aspecto relevante do Dual Lagrangeano é o fato do problema ser convexo mas não diferenciável. Um método bastante utilizado para encontrar uma solução aproximada para o problema LD, e até mesmo a solução ótima quando certas condições são satisfeitas, é o Método do Subgradiente.

4 Método do Subgradiente

Este método opera de forma semelhante ao algoritmo de descenso. Dado um vetor de multiplicadores de Lagrange u_k, o método encontra um subgradiente d_k que indica a direção na qual os multiplicadores devem ser aumentados/diminuídos. Assim, iterativamente, o método busca o conjunto ótimo de multiplicadores u^*, tal que $c^T x + u^{*T}(b - Ax)$ seja máximo.

Logo abaixo, é descrito o algoritmo iterativo do subgradiente (Algoritmo 1):

Algoritmo 1 Método do Subgradiente

$u = u^0$
$k = 1$
Enquanto critério de parada não for satisfeito **faça**
  Resolver o problema Lagrangeano (LD) com solução ótima $x(u^k)$
  $u^{k+1} = max\{u^k - \mu_k(b - Ax(u^k)), 0\}$
  $k \leftarrow k + 1$
Fim do enquanto

O vetor $d_k = b - Ax(u^k)$ é a direção do subgradiente de $z(u)$ para u^k. A cada iteração k é dado um passo do ponto corrente u^k em direção oposta ao subgradiente. A dificuldade está na escolha do tamanho do passo μ_k.

Teorema 1

1. Se $\sum_k \mu_k \to \infty$, e $\mu_k \to 0$ com $k \to \infty$, então $z(u^k) \to w_{LD}$ o valor ótimo de LD.
2. Se $\mu_k = \mu_0 \rho^k$ para algum $\rho < 1$, então $z(u_k) \to w_{LD}$ se μ_0 e ρ são suficientemente grandes.
3. Se $\overline{w} \geq w_{LD}$ e $\mu_k = \epsilon_k[z(u^k) - \overline{w}]/||b - Ax(u^k)||^2$ com $0 < \epsilon_k < 2$, então $z(u^k) \to \overline{w}$, ou o algoritmo encontra u^k cujo $\overline{w} \geq z(u^k) \geq w_{LD}$ para algum k finito.

5 O Problema do Caixeiro Viajante Simétrico

Considere um Grafo $G : (V, E)$, onde V é conjunto de vértices e E o conjunto de arestas, no qual $V = 1, \ldots, n$. Seja, a variável binária x_e igual a 1, se a aresta $e \in E$ é utilizada no caminho do caxeiro e 0 caso contrário. Define-se ainda a matriz de custos (ou distâncias) $C = [c_e]$. Uma possível formulação para este problema é dada por:

$$(PI) \quad z = min \sum_{e \in E} c_e x_e$$

$$\sum_{e \in \delta(i)} x_(e) = 2, \forall i \in V \tag{1}$$

$$\sum_{e \in E(S)} x_(e) \leq |S| - 1, \forall 2 \leq |S| \leq |V| - 1 \tag{2}$$

$$x \in \mathbb{B}^{|E|} \tag{3}$$

As restrições (1) especificam que cada vértice deve ter grau de incidência igual a 2. As restrições (2) eliminam a possibilidade da existência de subciclos, e as restrições (3) representam as condições binárias.

5.1 Relaxação Lagrangeana para o PCV Simétrico

Considere os dois seguintes fatos:

1. Toda rota é um caminho saindo do vértice 1;
2. Todo caminho é uma árvore.

Baseados nestes fatos, na década de 70, Held e Karp [3] propuseram uma famosa relaxação para o PCV Simétrico. Essa relaxação consiste em eliminar as restrições de eliminação de subciclos para o vértice $1 \in S$, e para obter a relaxação Lagrangeana, são dualizadas todas as restrições referentes aos graus de incidência nos vértices, exceto para o nó 1. Assim, também é criada uma nova restrição para que o número total de arestas seja igual a n, como pode ser visto pela formulação do problema relaxado, abaixo:

$$(RL) \quad z(u) = min \sum_{e \in E} (c_e - u_i - u_j) x_e + 2 \sum_{i \in V} u_i$$

$$\sum_{e \in \delta(1)} x_(e) = 2 \tag{4}$$

$$\sum_{e \in E(S)} x_(e) \leq |S| - 1, \forall 2 \leq |S| \leq |V| - 1, 1 \notin S \tag{5}$$

$$\sum_{e \in E} x_(e) = n \tag{6}$$

$$x \in \mathbb{B}^{|E|} \tag{7}$$

As soluções factíveis do problema RL são subgrafos chamados 1-árvores [3] (Figura 1). Por definição, uma 1-árvore é um subgrafo constituído pelo vértice 1 conectado por 2 arestas à árvore geradora mínima (AGM) obtida pelo grafo G original sem a presença do vértice 1. Assim, resolver o problema RL equivale a encontrar a 1-árvore mínima para os custos Lagrangeanos. Encontrar a 1-árvore mínima é um procedimento relativamente simples, pois basta conectar o nó 1 à AGM utilizando as duas arestas de menores custos. Portanto, é evidente que esta é uma forma interessante de relaxar o PCV. Note também, que enquanto o PCV é um problema de minimização, o seu Dual Lagrangeano é um problema de maximização e, como as restrições dualizadas são restrições de igualdade, os multiplicadores de Lagrange (variáveis duais) são irrestritos em sinal. Então, $w_{LD} = max_u z(u)$.

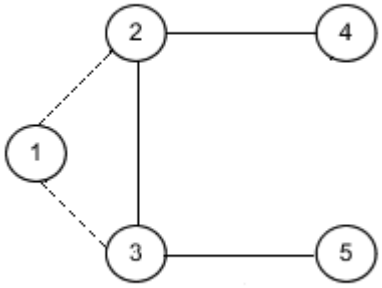

Figura 1: Exemplo de uma 1-árvore

5.2 Adaptação do Método do Subgradiente

Alguns ajustes devem ser realizados no método do subgradiente para resolver o PCV Simétrico [5, 6]:

A direção u (penalizadores de Lagrange) é calculada pela expressão

$$u_i^{k+1} = u_i^k + \mu_k(2 - \sum_{e \in \delta(i)} x_e(u^k)) \tag{8}$$

assim, a i-ésima coordenada do subgradiente $2 - \sum_{e \in \delta(i)} x_e(u^k)$ é dois menos o número de arestas incidentes no vértice i na 1-árvore ótima. O tamanho do passo é calculado por

$$\mu_k = \epsilon_k(\underline{w} - z(u^k))/\sum_{i \in V}(2 - \sum_{e \in \delta(i)} x_e(u^k))^2 \tag{9}$$

onde um limitante inferior $\underline{w}$ de w_{LD} é apropriado porque o PCV Simétrico é um problema de minimização e LD um problema de maximização.

Em [1] e [4] várias modificações são propostas para adaptar o método do subgradiente para resolver o PCV Simétrico. Neste trabalho, o algoritmo implementado segue as adaptações propostas por [1]:

1. O algoritmo termina quando $\overline{w} - \underline{w} \leq 1$, ou quando $\sum_{e \in \delta(i)} x_e(u^k) = 2$ para todo o vértice i pertencente à 1-árvore, ou quando $\epsilon_k \leq 0,0005$, ou ainda por número máximo de iterações;

2. Inicialmente, o tamanho do passo $\epsilon_0 = 2$. Se o valor de $\underline{w}$ não aumentar nas 30 primeiras iterações do subgradiente, então ϵ permanece igual a 2, caso contrário $\epsilon \leftarrow \epsilon/2$. A cada 30 iterações o valor de ϵ é dividido por 2;

3. Segundo [1], a forma como os multiplicadores de Lagrange são inicializados não influencia na velocidade de convergência do algoritmo nem na qualidade do limitante inferior $\underline{w}$. Neste caso, optou-se por inicializar os multiplicadores com zeros;

4. Ao invés de calcular o tamanho do passo utilizando a expressão 9, fazer

$$\mu_k = \epsilon_k(1,05\overline{w} - \underline{w}) / \sum_{i \in V}(2 - \sum_{e \in \delta(i)} x_e(u^k))^2 \tag{10}$$

6 Limitantes Primais e Duais para o PCV Simétrico

Nesta seção serão apresentados os algoritmos utilizados para encontrar os limitantes superiores (limitantes primais) e inferiores (limitantes duais) para o PCV Simétrico.

6.1 Limitante Dual

Para encontrar um limitante dual basta resolver o problema relaxado do PCV (ver Seção 5.1). Neste caso, o objetivo é encontrar a 1-árvore ótima, pois esta solução satisfaz a todas as restrições do problema *RL*.

Em questão à implementação, foi optado pelo uso do algoritmo de Kruskal para encontrar a árvore geradora mínima. O Algoritmo 2 descreve como encontrar a 1-árvore ótima.

Algoritmo 2 Algoritmo para encontrar a 1-árvore ótima

Entrada: Grafo $G : (V, E)$
Saída: 1-árvore ótima T

Passo 1: $G' \leftarrow G : (V \backslash \{1\}, E)$ // G' recebe o grafo G sem o vértice 1
Passo 2: $T \leftarrow Kruskal(G')$ // T recebe a árvore geradora mínima obtida pelo algoritmo de Kruskal quando aplicado ao grafo G'
Passo 3: Sejam e_i e e_j as duas arestas de menores custos ligadas ao vértice 1
Passo 4: $T \leftarrow T \cup \{e_i, e_j\}$
Passo 5: Retornar T

Basicamente a complexidade do Algoritmo 2 é limitada à complexidade do algoritmo de Kruskal. A complexidade do algoritmo de Kruskal é $\theta(|E|log|E|)$. Como as instâncias do PCV utilizadas neste trabalho são grafos completos, então, $|E| = O(|V|^2)$. Portanto, a complexidade do Algoritmo 2 é $O(|V|^2 log|V|)$.

Após ser encontrada a 1-árvore mínima T com custo $C(T)$, para calcular o valor do limitante dual (limite inferior) basta calcular

$$\underline{w} = C(T) + 2\sum u_i$$

Neste trabalho, foi utilizada uma idéia proposta em [4], no qual para gerar a 1-árvore deve-se obter a árvore geradora mínima (AGM) do grafo completo e ligar a folha com menor custo do grafo, ao invés de fixar o nó 1 e ligá-lo a AGM com as duas arestas mais baratas (ver Algoritmo 3).

Algoritmo 3 Algoritmo utilizado para encontrar a 1-árvore ótima

Entrada: Grafo $G : (V, E)$
Saída: 1-árvore ótima T

Passo 1: $T \leftarrow Kruskal(G)$ // T recebe a árvore geradora mínima obtida pelo algoritmo de Kruskal quando aplicado ao grafo G
Passo 2: Seja i um vértice folha //i tem grau de incidência menor que 2
Passo 3: Seja e_i a aresta de menor custo que conecta o vértice i à AGM
Passo 4: $T \leftarrow T \cup \{e_i\}$
Passo 5: Retornar T

6.2 Limitante Primal

Um limitante primal é dado por qualquer solução factível do PCV. Para isso, foi implementada uma heurística e um método de busca local para encontrar uma rota factível de boa qualidade. A heurística implementada é um algoritmo guloso (ver Algoritmo 4) que ordena as arestas em ordem crescente de custo e aloca-as de vértice em vértice, seqüencialmente.

O Algoritmo 4 é, na verdade, o algoritmo de Kruskal adaptado para encontrar rotas. Essa adaptação foi obtida pelas restrições de que não pode haver subciclos e que o grau de incidência nos vértices deve ser exatamente igual a 2. Caso essas restrições fossem extraídas, o algoritmo resultante seria exatamente o método de Kruskal para encontrar a árvore geradora mínima.

A complexidade desse algoritmo é $\theta(|E|(log|V| + log|E|))$, mas como as instâncias utilizadas são grafos completos, então, $|E| = O(|V|^2)$, portanto, a complexidade do algoritmo é $O(|V|^2 log|V|)$.

Para melhorar a qualidade das rotas encontradas pela heurística, foi implementada uma busca local do tipo 2-opt [6, 4].

Após ser encontrada a Rota R com custo $C(R)$, o valor do limitante primal (limite superior) é dado por

$$\overline{w} = C(R)$$

Algoritmo 4 Algoritmo Guloso para encontrar uma rota para o PCV

Entrada: Grafo $G : (V, E)$
Saída: Rota R

Passo 1: Ordenar as arestas em ordem crescente de custos: $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$, onde $m = n(n-1)/2$

Passo 2: $R \leftarrow \emptyset$
$k \leftarrow 1$
$g[i] = 0$ para todo $i \in \{1, \ldots, n\}$

Passo 3:
Enquanto $|R| < n$ **faça**
$\epsilon_k = (i, j)$ e $R' \leftarrow R \cup \{\epsilon_k\}$
Se $(g[i] < 2)$ e $(g[j] < 2)$ e (R' não contiver subciclo) **então**
$R \leftarrow R'$ e $k \leftarrow k + 1$
Fim do condicional
Fim do enquanto

Passo 4: Retornar R

7 Heurística Lagrangeana

Uma heurística lagrangeana HL para o PCV é um método que transforma uma 1-árvore (limitante dual) em uma rota factível (limitante primal). Muitas vezes, uma 1-árvore é uma forte candidata a solução do problema, se convertida numa rota [1]. A partir dessa idéia, é interessante utilizar os limitantes duais para melhorar os limitantes primais. No caso do problema do PCV, esse algoritmo é relativamente fácil de ser implementado, pois basta remover as arestas mais custosas dos vértices com grau de incidência maior que 2 da 1-árvore, e conectar arestas nos vértices com grau de incidência menor do que 2, com a preocupação de não formar subciclos.

Por sua vez, como explicado na Seção 6.2, o algoritmo guloso implementado gera uma rota factível baseando-se no algoritmo de Kruskal. Portanto, uma heurística Lagrangeana para o problema do PCV pode ser obtida pela utilização da mesma heurística gulosa, apenas com a substituição do parâmetro de entrada, que ao invés de ser o grafo com os custos originais, passa a ser o grafo com os custos atualizados pelos multiplicadores de Lagrange.

Como foi visto na Seção 6.2, caso as restrições de rota fossem retiradas do algoritmo guloso, as soluções encontradas seriam AGMs. Portanto, conclui-se que uma solução encontrada pela heurística Lagrangeana HL é igual a encontrada pelo algoritmo guloso quando aplicado ao grafo com custos atualizados.

8 Fixação de Variáveis

Segundo [1], reduzir o tamanho do problema consiste em fixar os valores de certas variáveis em 1 ou em 0, ou seja, como as variáveis x_e são binárias, fixá-las em 1 significa dizer que essas arestas fazem parte da solução para o PCV, e fixá-las em 0 é o mesmo que eliminar definitivamente a possibilidade delas pertencerem a solução. O problema de realizar fixação de variáveis para o PCV é que o custo computacional é altíssimo, portanto, recomenda-se que o método de fixação seja utilizado poucas vezes (aproximadamente 5) durante a execução do método do subgradiente, para que o tempo de execução não seja demasiadamente grande.

8.1 Fixando $x_e = 0$

Dada uma 1-árvore T com custo $C(T)$, para cada a aresta e_{ij} que não pertence à T, é encontrada uma 1-árvore T' com custo $C(T')$ na qual $e_{ij} \in T'$. Se $C(T') - C(T) > \overline{w} - \underline{w}$, então significa que a aresta e_{ij} tem um custo muito alto, e que a sua inclusão na 1-árvore aumenta demais o limitante inferior. Portanto, fazemos $x_e = 0$ (ver Algoritmo 5).

Algoritmo 5 Algoritmo para fixar variáveis em 0

Entrada: Grafo $G : (V, E)$, 1-árvore T com custo $C(T)$, $\overline{w}$ e $\underline{w}$

Passo 1: Armazene em A as arestas e_{ij} que não pertencem à 1-árvore T

Passo 2:
Para cada aresta $e_{ij} \in A$ **faça**
    $e_{ij} \leftarrow -\infty$
    Encontre uma nova 1-árvore em T' com custo $C(T')$
    Se $C(T') - C(T) > \overline{w} - \underline{w}$ **então**
        $x_{e_{ij}} \leftarrow 0$
    Fim do condicional
Fim do laço

8.2 Fixando $x_e = 1$

Dada uma 1-árvore T com custo $C(T)$, para cada a aresta e_{ij} que pertence à T, é encontrada uma 1-árvore T' com custo $C(T')$ na qual $e_{ij} \notin T'$. Se $C(T') - C(T) > \overline{w} - \underline{w}$, então significa que a aresta e_{ij} tem um custo baixo, e que substituí-la por uma outra aresta na 1-árvore aumenta demais o limitante inferior. Portanto, fazemos $x_e = 1$ (ver Algoritmo 6).

Algoritmo 6 Algoritmo para fixar variáveis em 1

Entrada: Grafo $G : (V, E)$, 1-árvore T com custo $C(T)$, $\overline{w}$ e $\underline{w}$

Para cada aresta $e_{ij} \in T$ **faça**
    $e_{ij} \leftarrow +\infty$
    Encontre uma nova 1-árvore em T' com custo $C(T')$
    Se $C(T') - C(T) > \overline{w} - \underline{w}$ **então**
        $x_{e_{ij}} \leftarrow 1$
    Fim do condicional
Fim do laço

9 Resultados Experimentais

Nesta Seção são apresentados diversos experimentos realizados com o método do subgradiente para resolver o PCV. Todos eles foram realizados na seguinte plataforma: Notebook Dell Inspiron - Intel Centrino Core Duo 1.83 Ghz - 512 Mb. O método foi implementado em linguagem C - sistema operacional Linux Ubuntu.

Todas instâncias testadas foram retiradas da base de dados TSPLib disponível em: http://www.iwr.uni-heidelberg.de/groups/comopt/soft/TSPLIB95/TSPLIB.html. As 29 instâncias do Caixeiro Viajante Simétrico possui número de cidades entre 200 e 750.

Os seguintes parâmetros mantiveram-se constantes em todos os experimentos:

- **Multiplicador do Passo inicial** $\epsilon_0 = 2$
- **Multiplicador do Passo final (parada)** $\epsilon_k \leq 0,0005$
- **Multiplicadores de Lagrange inicial** $u^0 = \mathbf{0}$
- **Método para obtenção do Limitante Primal:** Algoritmo Guloso (Algoritmo 4)
- **Método para obtenção do Limitante Dual:** Algoritmo da 1-árvore (Algoritmo 3)

9.1 Experimento 1

Neste primeiro experimento (Tabela 1), foi testado o método do subgradiente na sua forma mais básica. Neste caso, o objetivo é comprovar a eficiência da busca local 2-opt para encontrar limites primais melhores. A Expressão 9 foi utilizada para calcular o tamanho do passo (μ). Para estes testes não foi realizada fixação de variáveis.

A utilização da busca local 2-opt melhorou a qualidade do limitante primal para a maioria das instâncias testadas. Por exemplo, em ali535, o limite superior reduziu de 215.702 para 211.088. Em gr229, reduziu de 141.659 para 138.722. Além dessas, outras 25 instâncias também melhoraram o limitante primal pela utilização do método 2-opt, sendo que, apenas para as instâncias a280 e pr264 foram obtidos limitantes primais piores com a utilização da busca local. Entretanto, essa piora não está relacionada à utilização da busca local, e sim, a primeira solução encontrada pelo algoritmo

Tabela 1: Resultados obtidos para o Experimento 1

	Sem busca local 2-opt			Com busca local 2-opt		
Grafo	Lim.Dual	Lim.Primal	Tempo (s)	Lim.Dual	Lim.Primal	Tempo (s)
a280	2.565,94	2.769	19,51	2.565,94	2.784	23,02
ali535	201.134,48	215.702	129,10	201.122,98	211.088	134,54
att532	27.409,41	29.799	96,02	27.412,67	29.195	112,71
d493	34.827,67	36.981	77,73	34.827,86	36.464	85,35
d657	48.449,39	52.441	166,05	48.449,31	51.685	186,51
fl417	11.543,78	12.679	297,41	11.554,08	12.531	265,71
gil262	2.354,47	2.474	57,48	2.354,47	2.440	17,34
gr202	40.054,81	40.469	28,10	40.054,92	40.266	7,24
gr229	133.313,74	141.659	20,22	133.314,90	138.722	16,62
gr431	170.121,43	186.232	63,33	170.200,85	183.508	78,63
gr666	292.482,63	313.057	188,52	292.485,00	308.958	239,98
kroA200	29.069,21	30.966	9,37	29.064,36	29.969	9,49
kroB200	29.162,35	32.121	9,90	29.161,86	31.260	9,96
lin318	41.875,43	42.823	35,21	41.882,43	42.743	32,12
linhp318	45.096,06	46.179	27,83	45.096,23	45.838	34,42
p654	33.640,13	38.021	362,02	33.654,42	36.231	641,35
pa561	2.739,20	2.964	181,99	2.739,28	2.911	148,05
pcb442	50.496,41	53.080	131,41	50.496,86	53.502	80,76
pr226	79.837,24	83.166	19,64	79.759,04	81.982	18,58
pr264	49.027,90	49.684	44,27	49.020,30	49.736	28,55
pr299	47.378,89	51.213	33,79	47.379,38	50.255	21,64
pr439	105.845,17	112.426	105,04	105.861,15	110.706	98,38
rat575	6.723,81	7.252	118,87	6.723,82	7.156	129,97
rd400	15.156,79	16.020	48,95	15.156,78	15.864	58,30
si535	48.385,84	48.656	135,01	48.388,14	48.575	176,78
ts225	115.598,71	131.247	11,74	115.599,25	129.152	13,99
tsp225	3.877,82	4.269	14,61	3.877,95	4.193	16,72
u574	36.711,33	39.214	133,46	36.711,63	38.643	132,84
u724	41.651,12	45.333	244,47	41.651,74	44.127	255,69

guloso. Pois, caso as soluções encontradas pela busca não são melhores que a solução de partida, então esta é mantida, evitando perda de qualidade do limitante primal.

A realização da busca 2-opt ocasionou um pequeno acréscimo no tempo consumido para a maioria das instâncias, porém não comprometeu diretamente o desempenho do algoritmo (aumentando demais o custo computacional). Portanto, conclui-se que a sua utilização é benéfica para encontrar melhores limitantes primais.

9.2 Experimento 2

Neste segundo experimento (Tabela 2), foi testado o método do subgradiente com o limitante primal calculado de duas maneiras: uma somente pelo algoritmo guloso e outra com o algoritmo

guloso seguido de busca local 2-opt. Neste teste, o objetivo é avaliar a melhoria dos resultados pela alteração do cálculo do tamanho do passo (μ) utilizando a Expressão 10. Para analisar o comportamento dos Limitantes Primais e Duais durante as iterações, foi plotado um gráfico (Gráfico 2) para a primeira instância (a280), utilizando o método do subgradiente com os limitantes primais sendo calculados pelo algoritmo guloso seguido de busca local. Vale ressaltar que, para estes testes não foi realizada fixação de variáveis.

Tabela 2: Resultados obtidos para o Experimento 2

	Sem busca local 2-opt			Com busca local 2-opt		
Grafo	Lim.Dual	Lim.Primal	Tempo (s)	Lim.Dual	Lim.Primal	Tempo (s)
a280	2.565,93	2.810	19,85	2.565,91	2.754	22,09
ali535	201.128,30	214.638	123,13	201.133,94	209.147	119,52
att532	27.414,49	29.986	86,04	27.414,99	29.642	124,67
d493	34.826,99	36.766	73,33	34.827,28	36.395	103,56
d657	48.447,84	52.414	146,83	48.449,35	51.489	168,25
fl417	11.597,13	12.763	244,30	11.616,18	12.531	302,97
gil262	2.354,44	2.474	18,83	2.354,44	2.442	27,67
gr202	40.054,08	40.467	12,16	40.054,11	40.353	8,71
gr229	133.312,55	141.761	20,86	133.312,71	139.116	15,04
gr431	170.150,03	185.064	90,22	170.107,38	183.749	62,99
gr666	292.474,25	314.642	197,41	292.478,91	308.774	150,08
kroA200	29.063,27	30.799	9,94	29.068,34	30.155	10,15
kroB200	29.163,16	31.485	10,75	29.164,09	30.884	11,66
lin318	41.874,18	43.093	29,35	41.874,69	42.941	26,37
linhp318	45.095,32	46.380	23,64	45.096,08	45.851	32,76
p654	33.653,99	37.851	333,40	33.685,08	36.871	358,46
pa561	2.739,12	2.974	119,51	2.739,15	2.920	116,40
pcb442	50.496,57	54.664	121,72	50.492,74	53.505	105,43
pr226	79.841,60	83.129	19,28	79.767,52	82.558	95,68
pr264	49.018,99	49.836	37,08	49.019,56	49.603	166,48
pr299	47.377,88	51.359	23,08	47.378,84	50.319	118,38
pr439	105.849,26	111.833	101,99	105.852,20	111.062	235,04
rat575	6.723,60	7.279	140,64	6.723,73	7.169	152,59
rd400	15.156,60	15.962	66,73	15.156,65	15.845	65,03
si535	48.370,46	48.697	116,12	48.367,78	48.614	176,68
ts225	115.596,01	130.722	13,26	115.596,30	130.360	14,01
tsp225	3.877,70	4.250	15,35	3.877,85	4.105	20,43
u574	36.710,49	39.105	128,98	36.709,99	38.774	164,32
u724	41.650,56	45.219	297,87	41.650,99	44.080	11,57

Para a maioria das instâncias, a substituição do cálculo do tamanho do passo da Expressão 9 para a Expressão 10 melhorou os resultados obtidos pelo método do subgradiente. Em geral, a melhoria apresentada consiste em um aumento no valor do limitante dual e numa redução do limitante primal. Por exemplo, em ali535, o limitante dual aumentou de 201.122,99 para 201.133,94

e o limitante primal reduziu de 211.088 para 209.147. Com isso, foi possível fazer com que os limitantes se aproximassem mais da solução ótima. Portanto, conclui-se que, calcular o tamanho do passo utilizando a Expressão 10 acelera a convergência do algoritmo.

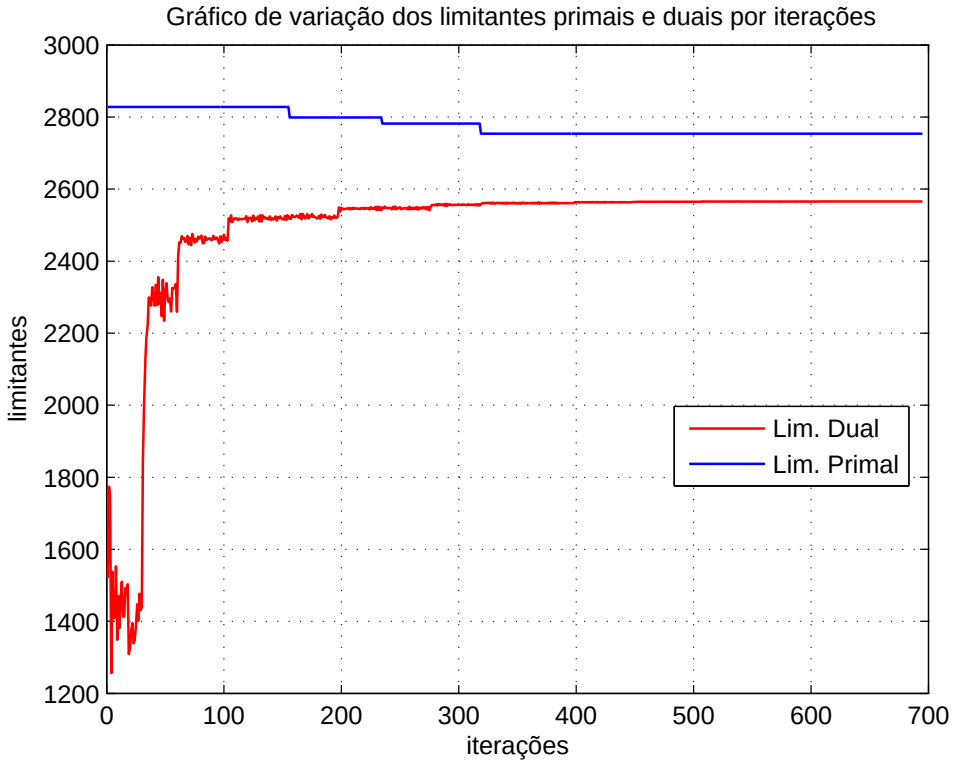

Figura 2: Gráfico de variação dos Limitantes Primais e Duais para a instância a280

Como podemos ver no Gráfico 2, nas primeiras 200 iterações, o valor do limitante dual aumenta rapidamente, e partir de então, passa a aumentar de forma mais lenta. Isso se deve ao fato do multiplicador ϵ_k do tamanho do passo, inicialmente, ser alto e, a cada 30 iterações ser dividido por 2, fazendo com que o aumento seja cada vez menor conforme o passar das iterações. Já para o limitante primal, a sua redução ocorre praticamente de forma gradativa, sendo que, um pouco depois da iteração 300, seu valor fica estagnado até o fim da execução do algoritmo.

9.3 Experimento 3

Neste terceiro experimento foi testado o método do subgradiente com o limitante primal calculado pelo algoritmo guloso adicionado de uma busca local 2-opt, e cálculo do tamanho do passo utilizando a Expressão 10. Neste teste, o objetivo é avaliar os resultados obtidos pela utilização do método de fixação de variáveis. Devido ao fato da fixação ser um processo muito "custoso" computacionalmente, optamos por fixar variáveis 5 vezes durante todas as iterações.

Analisando a Tabela 3, pode-se notar que a utilização do método de fixação de variáveis não melhorou os resultados obtidos quando comparados com os da Tabela 2 com busca 2-opt. Entre-

Tabela 3: Resultados obtidos para o Experimento 3

Grafo	Lim.Dual	Lim.Primal	Tempo (s)
a280	2.565,91	2.754	21,46
ali535	201.133,94	209.147	117,91
att532	27.414,99	29.642	122,27
d493	34.827,28	36.395	101,95
d657	48.449,35	51.489	172,46
fl417	11.616,18	12.531	3.143,40
gil262	2.354,44	2.442	15,05
gr202	40.054,11	40.353	12,73
gr229	133.312,71	139.116	67,52
gr431	170.107,38	183.749	138,13
gr666	292.478,91	308.774	147,62
kroA200	29.068,34	30.155	9,96
kroB200	29.164,09	30.884	11,51
lin318	41.874,69	42.941	25,94
linhp318	45.096,08	45851	32,29
p654	33.685,08	36.871	352,66
pa561	2.739,15	2.920	114,42
pcb442	50.492,74	53.505	62,07
pr226	79.767,52	82.558	14,50
pr264	49.019,56	49.603	26,70
pr299	47.378,84	50.319	19,29
pr439	105.852,20	111.062	77,36
rat575	6.723,73	7.169	112,82
rd400	15.156,65	15.845	48,26
si535	48.367,78	48.614	130,72
ts225	115.596,30	130.360	10,46
tsp225	3.877,85	4.105	15,09
u574	36.709,99	38.774	134,40
u724	41.650,99	44.080	203,78

tanto, percebe-se que para algumas instâncias houve uma aceleração na convergência do algoritmo, resultando em menor consumo de tempo computacional, como no seguinte caso: em pr439, o tempo reduziu de 235,04 para 77,36 segundos. Entretanto, para algumas outras instâncias houve, um aumento do tempo computacional consumido, como por exemplo: em fl417, o tempo aumentou de 302,97 para 3.143,40 segundos. Portanto, conclui-se que, o desempenho trazido ao método do subgradiente pela utilização do método de fixação de variáveis depende diretamente da instância a ser utilizada. Em alguns casos a sua utilização é benéfica e em outros é maléfica.

10 Conclusões

O Problema do Caixeiro Viajante Simétrico, apesar de ser facilmente formulado, é um problema muito difícil de ser resolvido. Devido a sua complexidade, utilizar métodos exatos para obter a solução ótima é um processo demasiadamente custoso e inviável. Gerar limitantes superiores e inferiores por meio da relaxação do problema é uma técnica eficaz e promissora para a implementação de algoritmos.

A relaxação Lagrangeana proposta por Held e Karp [3], mostrou-se uma boa estratégia para geração dos limitantes inferiores (1-árvores), enquanto que um algoritmo guloso pode ser facilmente utilizado para a obtenção dos limitantes primais. A utilização de uma busca local do tipo 2-opt melhora significativamente as rotas obtidas pelo algoritmo guloso sem influenciar diretamente no desempenho do algoritmo.

Dependendo da instância a ser resolvida, um método de fixação de variáveis pode ser utilizado para reduzir o tamanho do problema e, conseqüentemente, reduzir o tempo computacional.

Enfim, o Método do Subgradiente é uma técnica eficiente para a resolução do Problema do Caixeiro Vaijante, pois apesar da complexidade do problema, o método foi capaz de encontrar bons limitantes em pouco tempo computacional. Além de que, se parâmetros como cálculo do tamanho do passo (μ), e multiplicadores do passo (ϵ) forem corretamente calibrados, a convergência do algoritmo pode ser acelerada.

Referências

[1] J.E. Beasley. *Modern Heuristic Techniques for Combinatorial Problems*, chapter Chapter 6: Lagrangean relaxation. Blackwell Scientific Publications, 1993.

[2] E.E. Rothberg D.S. Jonhson, L.A. McGeoch. Asymptotic experimental analysis for the held-karp traveling salesman bound. In *Proceedings of the7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 341–350, 1996.

[3] M. Held and R.M. Karp. The travelling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, December 1970.

[4] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[5] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, 1988.

[6] L.A. Wolsey. *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization, 1998.