Aplicação do algoritmo volumétrico à resolução aproximada e exacta do problema do caixeiro viajante assimétrico

Ana Maria Rocha [†] Edite M.G.P. Fernandes [†] João Soares [*‡]

[†] Departamento de Produção e Sistemas
Universidade do Minho
{arocha;emgpf}@dps.uminho.pt

[‡] Departamento de Matemática
Universidade de Coimbra
jsoares@mat.uc.pt

Abstract

In this paper we present computational results with the volume algorithm, a variant of the subgradient method, when solving the linear relaxation that stems from the extended disaggregated flow formulation of the Asymmetric Travelling Salesman Problems. Computational experiments were performed on a selection of instances from the TSPLib and some randomly generated instances according to the *Dimacs Implementation Challenge*. We have also tried ATSP heuristics within the volume algorithm. Computational experiments show moderated success on medium-scale instances.

Resumo

Neste artigo apresentamos resultados computacionais obtidos com o algoritmo volumétrico, uma variante do método do subgradiente, na resolução da relaxação linear que decorre da formulação estendida de fluxo desagregado para o problema do Caixeiro Viajante Assimétrico. As experiências computacionais foram realizadas numa selecção de instâncias da TSPLib e num conjunto de instâncias geradas aleatoriamente de acordo com o *Dimacs Implementation Challenge*. Também experimentámos a aplicação de heurísticas durante a execução do algoritmo volumétrico. As experiências computacionais mostram sucesso moderado com instâncias de média dimensão.

Keywords: Asymmetric travelling salesman problem, Disaggregated flow formulation, Lagrangian relaxation.

[*]João Soares acknowledges partial financial support from Fundação para a Ciência e Tecnologia (Projecto POCTI/MAT/14243/1998).

Title: Application of the volume algorithm to the approximate and exact solving of the asymmetric travelling salesman problem.

---

1 Introdução

Seja $G = (V, A)$ um grafo orientado simples (*i.e.*, sem laços ou arcos múltiplos), com n vértices e m arcos, e um custo c_{ij} associado a cada arco (i, j). O problema do Caixeiro Viajante Assimétrico, mais adiante referido como PCVA, consiste em determinar em G um circuito Hamiltoniano (*i.e.*, um circuito que passa por todos os vértices uma única vez) cuja soma dos custos associados aos seus arcos seja a menor possível. Este problema pertence à classe dos problemas NP-completos. A referência [27] sumaria a investigação no PCVA em todos os seus aspectos até 1985, enquanto que [17] é uma referência do mesmo género mas mais actual.

Diversas formulações têm sido propostas para este problema. Em geral, mas nem sempre, elas incorporam as restrições de um problema de afectação (em número igual a $2n$ se excluirmos as de não negatividade) e as restrições que eliminam as soluções inteiras que não definem circuitos Hamiltonianos (que podem ser em número polinomial ou exponencial em n e m). Uma das primeiras formulações foi sugerida em [10], a formulação DFJ. Apesar de possuir um número exponencial de restrições, esta formulação permanece como sendo a formulação de base para os códigos *state-of-the-art*, como é o caso de FT-B&C [11]. Outros investigadores propuseram formulações que envolvem um número polinomial de variáveis e restrições (as formulações compactas), à custa da introdução de variáveis auxiliares, *e.g.*, [29, 12, 33, 8, 15, 16]. As diferentes formulações são comparadas em, por exemplo, [30, 14, 26]. Na língua portuguesa, recomendamos a dissertação de José Pires [31] onde todas estas formulações são analisadas.

Ainda que as formulações compactas possuam um número polinomial de variáveis e restrições, esse número pode ser o suficiente para dificultar a mera resolução da relaxação linear. Por exemplo, para um método simplex, a base pode ter dimensão de tal modo elevada que o método tem grandes dificuldades em aproximar-se da base óptima; para um método de pontos interiores, a dificuldade poderá ser o espaço de armazenamento em memória. Esta parece ser uma das razões pela menor utilização deste tipo de formulações em códigos *branch-and-cut*. Neste contexto, surge a questão que motivou o nosso trabalho, a técnica de relaxação Lagrangeana poderá definir uma base inicial mais próxima da base óptima e, deste modo, acelerar a execução do método simplex.

A aplicação de relaxação Lagrangeana ao problema do Caixeiro Viajante foi iniciada por Held e Karp [18, 19] para a versão simétrica. Em geral, aplica-se relaxação Lagrangeana quando um grupo de restrições *destrói* o facto de que um problema combinatório é estruturado e pode ser resolvido por métodos específicos muito eficientes (*e.g.*, caminho mais curto com restrições adicionais, árvore de suporte com restrições de grau, etc). Aplicar esta técnica significa minimizar uma função convexa não diferenciável num conjunto poliedral. Um dos métodos que mais tem sido usado neste tipo de problemas é o método do subgradiente. Cada iteração deste método consiste na resolução de um subproblema estruturado (árvore de peso mínimo, no caso da abordagem de Held e Karp) e na redefinição da respectiva função objectivo - essencialmente caracterizada pelos valores de variáveis duais relativamente às tais restrições

problemáticas. Uma variante do método de subgradiente que tem recentemente obtido razoável sucesso na resolução da relaxação linear de problemas de optimização combinatória é o Algoritmo Volumétrico (AV) [4].

Neste trabalho abordamos, essencialmente, a resolução da relaxação linear da formulação de fluxo desagregado para o PCVA, proposta em [8], através do algoritmo volumétrico. O subproblema estruturado a resolver em cada iteração é um problema de afectação. As soluções primais e duais assim obtidas são, então, usadas para definir uma base inicial para o método dual-simplex. Os custos reduzidos gerados no decurso do AV são também aproveitados para a execução de métodos heurísticos. Testámos a nossa abordagem em instâncias da TSPLib e instâncias geradas aleatoriamente.

Este artigo está organizado da seguinte forma. A secção 2 introduz a formulação de fluxo desagregado do PCVA. A secção 3 apresenta o problema dual Lagrangeano, incluindo as simplificações que ocorrem na resolução do subproblema. A secção 4 sumaria o algoritmo volumétrico, que é usado na resolução do problema dual. A secção 5 apresenta os resultados computacionais obtidos e a secção 6 sumaria as conclusões.

2 Formulação

O conjunto dos vectores característicos de circuitos Hamiltonianos de G é um subconjunto das soluções inteiras que pertencem ao poliedro P^{A} descrito pelo seguinte sistema de igualdades e desigualdades

$$\left.\begin{array}{rll} \displaystyle\sum_{(i,j)\in\delta^+(i)} x_{ij} & = 1 & (i \in V) \\ \displaystyle\sum_{(i,j)\in\delta^-(j)} x_{ij} & = 1 & (j \in V) \\ x_{ij} & \geq 0 & ((i,j) \in A) \end{array}\right\} \tag{1}$$

onde $\delta^-(j)$ denota o conjunto dos arcos que convergem para o vértice j e $\delta^+(i)$ denota o conjunto dos arcos que divergem do vértice i. O conjunto dos vectores característicos de circuitos Hamiltonianos de G é o subconjunto daquelas soluções inteiras x que satisfazem, para alguns vectores $y^2, y^3, \ldots, y^n \in \mathbb{R}^m$, o seguinte sistema de igualdades e desigualdades

$$\left.\begin{array}{ll} \displaystyle\sum_{(1,j)\in\delta^+(1)} y^k_{1j} - \sum_{(j,1)\in\delta^-(1)} y^k_{j1} = -1 & (k \in V \setminus \{1\}) \\ \displaystyle\sum_{(k,j)\in\delta^+(k)} y^k_{kj} - \sum_{(j,k)\in\delta^-(k)} y^k_{jk} = 1 & (k \in V \setminus \{1\}) \\ \displaystyle\sum_{(i,j)\in\delta^+(i)} y^k_{ij} - \sum_{(j,i)\in\delta^-(i)} y^k_{ji} = 0 & (k \in V \setminus \{1\}, i \in V \setminus \{1,k\}) \\ 0 \leq y^k_{ij} \leq x_{ij} & ((i,j) \in A, k \in V \setminus \{1\}). \end{array}\right\} \tag{2}$$

Esta formulação proposta por Claus [8], e inspirada em Wong [33], é conhecida por formulação estendida de fluxo desagregado. A designação "fluxo" deve-se ao facto de se usar variáveis de fluxo y^k_{ij} $((i,j) \in A, k \in V \setminus \{1\})$ que se podem interpretar como indicando a quantidade de fluxo percorrendo o arco (i,j), enviada pelo vértice 1 e com destino ao vértice k. A designação "desagregado" deve-se ao facto de haver distinção por vértice.

A projecção do poliedro definido pelos sistemas de igualdades e desigualdades (1) e (2) no espaço das variáveis $x_{ij}, (i,j) \in A$, define um poliedro denotado P_L^{PCVA} que é definido por (1) e

$$\sum_{(i,j)\in A(S)} x_{ij} \leq |S| - 1 \quad (S \subseteq V \setminus \{1\}) \tag{3}$$

onde $A(S)$ denota o conjunto dos arcos do grafo induzido $G[S]$. O interesse em usar (1) e (2) em vez de (1) e (3) tem naturalmente a ver com o número de restrições que é, respectivamente, polinomial num caso e exponencial no outro.

3 Problema Lagrangeano

O problema de optimização linear $z_L^* \equiv \min\{cx\colon x \in P_L^{\text{PCVA}}\}$ pode descrever-se do seguinte modo usando o formato matricial, que aproveitamos para exibir o dual,

$$z_L^* = \left\{ \begin{array}{lllll} \min & cx & & & \\ \text{s.a} & Dx & = & \mathbb{1} & \\ & By^k & = & b^k & (k \in V_1) \\ & x - y^k & \geq & 0 & (k \in V_1) \\ & y^k & \geq & 0 & (k \in V_1) \\ & x \geq 0 & & & \end{array} \right\} = \left\{ \begin{array}{lllll} \max & w\mathbb{1} + \sum_{k\in V_1} \pi^k b^k & & & \\ \text{s.a} & wD + \sum_{k\in V_1} \rho^k & \leq & c & \\ & \pi^k B - \rho^k & \leq & 0 & (k \in V_1) \\ & \rho^k & \geq & 0 & (k \in V_1) \end{array} \right\} \tag{4}$$

onde $V_1 \equiv V \setminus \{1\}$, D denota a matriz de incidência vértice-aresta do grafo bipartido não orientado $G' = (V \times V, A)$, $\mathbb{1}$ é um vector de uns, B denota a matriz de incidência vértice-arco do grafo G e, para cada $k \in V_1$, b^k é um vector cujas componentes 1 e k são iguais, respectivamente, a -1 e 1 e as restantes são iguais a 0. Procurando explorar a estrutura do problema de minimização em (4), consideremos o problema Lagrangeano

$$z_L^* = \max\left\{ z\left(\pi\right) : \pi = \left[\pi^k\right] \in \mathbb{R}^{(|V|-1)\times|V|} \right\} \tag{5}$$

cuja função objectivo $z\colon \mathbb{R}^{(|V|-1)\times|V|} \to \mathbb{R}$ é definida em cada π por

$$z(\pi) \equiv \left\{ \begin{array}{ll} \min & cx + \sum_{k\in V_1} \pi^k \left(b^k - By^k\right) \\ \text{s.a} & Dx = \mathbb{1} \\ & x - y^k \geq 0 \quad (k \in V_1) \\ & y^k \geq 0 \quad (k \in V_1) \\ & x \geq 0 \end{array} \right\} = \left\{ \begin{array}{ll} \max & w\mathbb{1} + \sum_{k\in V_1} \pi^k b^k \\ \text{s.a} & wD + \sum_{k\in V_1} \rho^k \leq c \\ & -\rho^k \leq -\pi^k B \quad (k \in V_1) \\ & \rho^k \geq 0 \quad (k \in V_1) \end{array} \right\}. \tag{6}$$

Uma solução óptima do problema de minimização em (6) pode ser obtida através da resolução de um adequado problema de afectação. Primeiro, observe-se que para todo x fixo, uma solução óptima $\bar{y}(x) = [\bar{y}^k(x)]$ no espaço das variáveis y é solução óptima de

$$\begin{array}{rl} \max & \displaystyle\sum_{k\in V_1} \sum_{(i,j)\in A} \left(\pi_i^k - \pi_j^k\right) y_{ij}^k \\ \text{s.a} & 0 \leq y_{ij}^k \leq x_{ij} \ ((i,j) \in A, k \in V_1) \end{array} \tag{7}$$

que admite o valor óptimo $\sum_{k\in V_1} \sum_{(i,j)\in A} \max\left(\pi_i^k - \pi_j^k, 0\right) x_{ij}$ para uma solução óptima explícita

$$\bar{y}_{ij}^k(x) = \left\{ \begin{array}{ll} x_{ij} & \text{se } \pi_i^k - \pi_j^k \geq 0, \\ 0 & \text{se } \pi_i^k - \pi_j^k < 0. \end{array} \right\} \quad ((i,j) \in A, k \in V_1)\,. \tag{8}$$

Por isso, o problema primal em (6) é equivalente ao problema de afectação:

$$\begin{array}{ll} \min & \bar{c}x \\ \text{s.a} & Dx = \mathbb{1}, x \geq 0 \end{array} \tag{9}$$

para um vector $\bar{c}$ definido componente a componente por $\bar{c}_{ij} = c_{ij} - \sum_{k \in V_1} \max\left((\pi_i^k - \pi_j^k), 0\right)$, na medida em que, se $\bar{x}$ é uma solução óptima de (9) então $(\bar{x}, \bar{y})$, com $\bar{y} = \bar{y}(\bar{x})$ definido em (8), é solução óptima para (6). A resolução de (9) pode efectuar-se através de, por exemplo, o método Húngaro em $\mathcal{O}(n^3)$ operações. Mais, se $\bar{x}$ é uma solução óptima de (9) então

$$z(\pi) = \bar{c}\bar{x} + \sum_{k \in V_1} \left(\pi_1^k - \pi_k^k\right),$$

e uma solução óptima $(\bar{w}, \bar{\rho})$ do dual em (6) é caracterizada pela solução dual $\bar{w}$ óptima para o problema de afectação (9) e $\bar{\rho}_{ij}^k = \max(\pi_i^k - \pi_j^k, 0)$, para todo $k \in V_1$ e $(i,j) \in A$. Note-se que o vector $(\pi, \bar{w}, \bar{\rho})$ é dual admissível em (4) pois, para cada $(i,j) \in A$,

$$\left[c - \left(\bar{w}D + \sum_{k \in V_1} \bar{\rho}^k\right)\right]_{ij} = c_{ij} - \left(\bar{u}_i + \bar{v}_j + \sum_{k \in V_1} \max(\pi_i^k - \pi_j^k, 0)\right) = \bar{c}_{ij} - (\bar{u}_i + \bar{v}_j) \geq 0,$$

e ainda porque, para cada $(i,j) \in A$ e $k \in V_1$,

$$\left[0 - \left(\pi^k B - \bar{\rho}^k\right)\right]_{ij} = -\left(\pi_i^k - \pi_j^k - \max(\pi_i^k - \pi_j^k, 0)\right) = \max(\pi_j^k - \pi_i^k, 0) \geq 0.$$

A próxima secção mostra como o algoritmo volumétrico é usado para resolver (5). Tal como foi mencionado anteriormente, foi dada especial atenção à forma como foi usada a relaxação Lagrangeana e às técnicas do subgradiente de tal forma que os subproblemas Lagrangeanos a resolver são reoptimizados de forma muito rápida.

4 Algoritmo Volumétrico (AV)

O algoritmo volumétrico, proposto em [4], é uma variante do algoritmo do subgradiente tradicional que, com comparável esforço computacional, produz não só, boas aproximações para as variáveis duais como também para as variáveis primais. A convergência global de uma implementação específica desse algoritmo é analisada em [3]. Experiências computacionais com a aplicação do algoritmo volumétrico na resolução da relaxação linear de problemas de optimização combinatória incluem o problema de partição [4, 1, 5], o problema da árvore de "Steiner" [2] e o problema de localização [6]. A resolução de problemas de optimização combinatória por *branch-and-cut*, onde os modelos lineares são resolvidos pelo algoritmo volumétrico, incluem os problemas da árvore de "Steiner" e de corte máximo [7].

Uma adaptação do algoritmo volumétrico para o PCVA é descrita na Figura 1. O método possui algumas semelhanças com os métodos de feixe [3]. Em particular, usa a ideia de, em cada iteração, usar uma direcção de deslocamento que combina uma nova direcção com a anterior direcção de deslocamento. Deste modo, procura-se evitar o comportamento de ziguezague, típico do algoritmo do subgradiente, de modo a acelerar a convergência.

Entrada: $\pi^0 \in \mathbb{R}^{(|V|-1)\times|V|}$.

Inicialização: Resolver (6) com $\pi = \pi^0$ para obter a solução (x^0, y^0).
Definir $v^0 = b - By^0 \in \partial z(\pi^0)$.
Iniciar $\bar{\pi}^1 = \pi^0$, $(\bar{x}, \bar{y}) = (x^0, y^0)$, $\bar{w} = v^0$, $j = 1$ e $l = 1$.

Iteração Genérica j:

Passo 1: Para algum comprimento do passo $s_j > 0$, definir $\pi^j = \bar{\pi}^l + s_j \bar{w}$.
Passo 2: Obter (x^j, y^j) uma solução óptima de (6).
Definir $v^j = [v^{k,j}] \in \partial z(\pi^j)$ com $v^{k,j} = b^k - By^{k,j}$.
Passo 3: Para algum $\alpha_j \in [0, 1]$, definir

$$(\bar{x}, \bar{y}) = \alpha_j \left(x^j, y^j\right) + (1 - \alpha_j)\,(\bar{x}, \bar{y}) \quad (10)$$

$$\bar{w} = \alpha_j v^j + (1 - \alpha_j)\,\bar{w}. \quad (11)$$

Passo 4: Se $z(\pi^j) > z(\bar{\pi}^l)$, então definir $\bar{\pi}^{l+1} = \pi^j$ e fazer $l \leftarrow l + 1$.
Passo 5: Testar critério de paragem. Fazer $j \leftarrow j + 1$ e voltar para *Passo 1.*

Figura 1: Algoritmo Volumétrico adaptado para o PCVA.

A descrição do algoritmo volumétrico deixa em aberto, em cada iteração j, a definição do tamanho do passo s_j e o escalar α_j envolvido na redefinição da solução primal $\bar{x}$. As nossas escolhas para definição destes dois escalares são essencialmente empíricas e semelhantes às usadas em [4]. O escalar α_j é escolhido por forma a forçar a primal admissibilidade de $(\bar{x}, \bar{y})$. Conforme [4], primeiro define-se α_{opt} como

$$\alpha_{\text{opt}} \equiv \arg\min \left\|\alpha v^j + (1 - \alpha)\,\bar{w}\right\|^2 = \arg\min \sum_{k \in K} \left\|b^k - B\left(\alpha y^{k,j} + (1 - \alpha)\,\bar{y}^k\right)\right\|^2. \quad (12)$$

Depois, é atribuído um valor a $\alpha_{\max}$ (configurado pelo parâmetro `alphainit=0.01`) e define-se

$$\alpha_j = \begin{cases} \alpha_{\max}/10 & \text{se } \alpha_{\text{opt}} \leq 0, \\ \min(\alpha_{\text{opt}}, \alpha_{\max}) & \text{se } \alpha_{\text{opt}} > 0, \end{cases}$$

onde $\alpha_{\max}$ é um limite superior para o valor de α.

Após um certo número de iterações (configurado pelo parâmetro `alphaint=80`), é testado o progresso da função dual $z(\pi)$. Caso tenha aumentado menos de 1% e se $\alpha_{\max}$ for superior a um certo valor (configurado pelo parâmetro `alphamin=0.0001`) então $\alpha_{\max}$ é multiplicado por uma dada quantidade (configurado pelo parâmetro `alphafactor=0.5`).

Tal como no método do subgradiente [20], o comprimento do passo s_j é feito através da seguinte fórmula:

$$s_j = \lambda_j \frac{T - z(\bar{\pi}^l)}{\|\bar{w}\|^2} \quad (13)$$

com $\lambda_j \in (0, 2]$ e T é um valor estimado para o valor óptimo de (5). Inicialmente é usado um valor pequeno para T (por exemplo, $T = z(\pi^0)$). Cada vez que $z(\bar{\pi}^l)$ está a 5% de T, o valor de T é aumentado em 5%.

A determinação do valor de λ, em cada iteração, obedece aos seguintes passos. O valor inicial de λ é 0.1 (`lambdainit=0.1`), depois o valor de λ difere consoante a iteração é *red*, *yellow* ou *green.*

- Quando $z(\pi^j) \leq z(\bar{\pi}^l)$, não se verifica um melhoramento no valor dual e a iteração diz-se *red.* Uma sequência de **40** (`redtestinvl=40`) iterações *red* consecutivas sugere a necessidade de um comprimento do passo menor. Assim, o valor de λ é multiplicado por 0.67, se $\lambda \geq 0.0005$, e é mantido constante, caso contrário.
- Quando $z(\pi^j) > z(\bar{\pi}^l)$, obteve-se um melhoramento no valor dual e efectua-se o seguinte cálculo
$$d = \bar{w}\,(b - B\bar{y})\,.$$
  - Se $d < 0$, a iteração chama-se de *yellow*, e um passo mais longo na direcção $\bar{w}$ origina um valor menor para $z(\pi^j)$. Assim, após **2** (`yellowtestinvl=2`) iterações *yellow* consecutivas λ é multiplicado por 1.1.
  - Se $d \geq 0$, a iteração diz-se *green*, e é sugerido um passo maior. Nestas condições, após **2** (`greentestinvl=2`) iterações *green* consecutivas λ é multiplicado por 2. Se o resultado for superior a 2 passa a ter o valor de 2.

é necessário que uma das seguintes duas condições seja verificada para fazer a paragem do algoritmo:

1. Uma delas baseia-se na verificação do limite máximo de iterações
$$\text{n}^o \text{ máximo de iterações} \leq 500/1000/5000/10000 \text{ (}\texttt{maxsgriters}\text{)}$$
2. A outra é composta pela conjunção de duas condições
   (a) o valor máximo absoluto da violação das restrições
   $$\|\bar{w}\| \leq 0.01 \text{ (}\texttt{primal\_abs\_precision}\text{)}$$
   e
   (b) a diferença relativa ou absoluta entre o limite inferior $z(\bar{\pi}^l)$ e o valor da função objectivo primal $c\bar{x}$
   $$\left(\frac{\left|c\bar{x} - z(\bar{\pi}^l)\right|}{|z(\bar{\pi}^l)|}\right) \leq 0.01 \text{ (}\texttt{gap\_rel\_precision}\text{)} \quad \text{ou} \quad \left|c\bar{x} - z(\bar{\pi}^l)\right| \leq 0.05 \text{ (}\texttt{gap\_abs\_precision}\text{).}$$

5 Resultados computacionais

Nesta secção descrevemos as experiências computacionais que efectuámos. Todos os códigos foram implementados em C++ e todas as experiências foram efectuadas num computador com processador Pentium 4 a 2.66Ghz com 512 Mb de memória RAM.

A Tabela 1 contém as principais características das instâncias seleccionadas da TSPLIB[32]. As primeiras colunas da tabela identificam o problema e as últimas colunas sumariam o comportamento do método dual-simplex (CPLEX DUALOPT), do método dos pontos interiores (CPLEX BAROPT) e do método *branch-and-bound* (CPLEX MIP) que decorre da execução do CPLEX 7.0 com base na formulação de fluxo desagregado. A execução de CPLEX MIP foi interrompida ao fim de 8 horas, sendo nesse caso apresentado '*' na respectiva entrada da Tabela 1. Nas colunas associadas à descrição do problema são listados o nome da instância, o número de vértices e o número de arcos, seguido do valor óptimo inteiro e do valor óptimo fraccionário (*i.e.*, o valor óptimo da relaxação linear). Nas restantes colunas listamos o número total de iterações do CPLEX e o tempo gasto em segundos.

Tabela 1: Instâncias do PCVA e soluções obtidas com o CPLEX.

PCVA					**CPLEX (dualopt)**		**CPLEX (baropt)**	**CPLEX MIP**	
ID	$\|V\|$	$\|A\|$	Int. óptimo	Frac. óptimo	Simplex It.	Tempo (seg.)	Tempo (seg.)	Simplex It.	Tempo (seg.)
br17	17	272	39	39.0	3055	2.2	0.4	27349	162.2
ftv33	34	1122	1286	1286.0	24231	42.7	18.2	24226	56.8
ftv35	36	1260	1473	1457.3	31627	76.2	23.5	288329	2094.9
ftv38	39	1482	1530	1514.3	41240	121.9	32.2	2403409	25536.7
ftv44	45	1980	1613	1584.9	68004	278.9	59.1	*	*
ftv47	48	2256	1776	1748.6	100650	627.1	106.5	*	*
ftv55	56	3080	1608	1584.0	137580	1035.9	237.7	*	*
ftv64	65	4160	1839	1807.5	245467	2762.7	511.2	*	*
ftv70	71	4970	1950	1909.0	316196	3719.2	791.0	*	*
p43	43	1806	5620	5611.0	131426	2117.6	60.9	*	*
ry48p	48	2256	14422	14289.3	98816	600.2	107.0	*	*
ft53	53	2756	6905	6905.0	127702	827.1	175.2	*	*
ft70	70	4830	38673	38652.5	291086	3657.6	753.4	*	*

Nas instâncias maiores, como é o caso da instância ftv70, o método dual-simplex é muito mais lento do que o método de pontos interiores. O comportamento superior do método de pontos interiores nestas instâncias era esperado porque o número de variáveis e restrições é muito elevado. Destacamos o comportamento do método de pontos interiores com a instância p43, requerendo apenas 2.9% do tempo despendido pelo método dual-simplex.

Motivados pela dificuldade do método dual-simplex em identificar a base óptima para estas instâncias ocorreu-nos a ideia de usar a solução proveniente do algoritmo volumétrico para definir uma base inicial mais próxima da base óptima. Também, averiguaremos a adequação do AV como mecanismo de identificação de arcos que não participam na solução óptima inteira e aproveitaremos os custos reduzidos gerados no decurso do AV para obter soluções inteiras pela via heurística.

5.1 Resolução do modelo relaxado

Começamos por analisar a estratégia de usar o par de soluções primal-dual encontrada pelo algoritmo volumétrico para definir a solução inicial para o método dual-simplex. Usámos a

implementação do algoritmo volumétrico disponível em `http://www.coin-or.org`, a página-web do projecto COIN-OR. Notámos que, à semelhança do algoritmo de subgradiente clássico, também o algoritmo volumétrico tem um comportamento muito diferenciado para diferentes valores dos seus parâmetros. Os valores para os parâmetros que utilizámos foram já descritos na Secção 4. Em cada iteração do algoritmo volumétrico, o problema de afectação (9) é resolvido pelo método dual-simplex. Experiências computacionais preliminares permitiram concluir que essa estratégia é superior, em tempo de execução, à invocação do método primal-simplex para redes.

A Tabela 5.1 sumaria os nossos resultados computacionais. Cada uma das linhas dessa tabela corresponde à resolução da relaxação linear de uma instância. Cada instância é resolvida duas vezes pelo método dual-simplex, consoante o número máximo de iterações imposto ao algoritmo AV. Nas instâncias de menor dimensão optou-se por um número máximo de 500 e 1000 iterações, nas instâncias intermédias esse número foi de 1000 e 5000 e nas instâncias de maior dimensão o mesmo número foi de 5000 e 10000 iterações. A informação inicial fornecida ao CPLEX é efectuada através de uma chamada da função `CPXcopystart(env,lp,NULL,NULL,x, NULL,NULL,y)`, onde `env` e `lp` são apontadores para o ambiente CPLEX e para o problema, respectivamente, `x` designa o vector primal e `y` designa o vector dual tal como foram obtidos no final do algoritmo AV.

Na primeira coluna da Tabela 5.1 é identificada a instância do PCVA. Nas cinco colunas seguintes são apresentados os resultados obtidos com o AV: o número máximo de iterações; o limite inferior obtido, com base em informação dual; o valor na função objectivo da última solução primal encontrada; o valor da violação máxima das restrições nessa solução; e, o tempo gasto em segundos. Nas duas colunas seguintes apresentamos: o número de iterações executadas pelo método dual-simplex; e, o tempo gasto em segundos. As duas últimas colunas incluem uma coluna com a soma dos tempos de execução dos dois algoritmos e uma coluna que mostra, em percentagem, o ganho de tempo entre esta estratégia combinada (AV+CPLEX) e o tempo do CPLEX (dualopt) apresentado na Tabela 1.

Observamos que, à excepção da instância br17, a um maior número de iterações do algoritmo volumétrico corresponde sempre um menor número de iterações do método dual-simplex. A redução é significativa nalguns casos. Por exemplo, 5000 iterações do AV não têm influência na resolução da instância ftv55. No entanto, depois de duplicar o número de iterações para 10000, o CPLEX necessitou apenas de 20 iterações executadas em 16.8 segundos para chegar à solução óptima. Outro exemplo significativo é o da resolução da instância ftv70 que após 5000 iterações do AV o tempo que dominava na determinação da solução era o tempo despendido pelo CPLEX. Quando duplicamos o número de iterações do AV, passámos a obter um ganho de tempo de 87.8%.

Se compararmos os tempos de resolução das Tabelas 1 e 5.1 observamos, quase sempre, uma vantagem clara em usar o AV antes de invocar o método dual-simplex do CPLEX. Nalguns casos, como ftv70, a abordagem combinada é superior ao método de pontos interiores. No entanto, as instâncias br17 e p43 contrariam esta observação. A Figura 2 permite uma outra visualização da superioridade da estratégia combinada naquele conjunto de instâncias.

Desde o início, pensámos que, atendendo ao papel das restrições dualizadas nesta abordagem Lagrangeana, o número de subcircuitos na solução óptima de (6) diminuiria à medida que o algoritmo volumétrico decorre - tornando-as cada vez mais parecidas com circuitos. Esta hipótese não foi completamente validada embora, de facto, se observe uma tendência para a

Tabela 2: Resultados da aplicação do AV seguido do CPLEX.

PCVA ID	VOLUMÉTRICO					CPLEX		Tempo Total	Tempo Ganho
	AV It.	L (dual)	P (primal)	Max. violação	Tempo (seg.)	Simplex It.	Tempo (seg.)		
br17	500	33.5	40.4	0.00416	0.7	1208	2.3	3.0	-34.1%
br17	1000	38.3	40.0	0.00237	1.4	1208	2.4	3.8	-70.5%
ftv33	500	1246.2	1304.1	0.04462	2.8	4421	19.1	21.9	48.8%
ftv33	1000	1281.9	1311.1	0.03612	5.5	2700	13.8	19.3	54.7%
ftv35	500	1435.6	1464.2	0.13534	3.2	12449	59.4	62.6	17.8%
ftv35	1000	1454.6	1466.0	0.11960	6.4	2190	13.4	19.8	74.1%
ftv38	500	1496.1	1513.5	0.11292	3.6	8999	59.6	63.2	48.2%
ftv38	1000	1512.8	1514.6	0.08486	7.1	2926	20.8	27.9	77.1%
ftv44	1000	1577.1	1586.0	0.10168	10.8	8217	82.5	93.3	66.5%
ftv44	5000	1584.7	1585.8	0.05515	52.4	2408	27.8	80.3	71.2%
ftv47	1000	1740.3	1750.5	0.09320	14.1	27997	347.9	362.1	42.3%
ftv47	5000	1746.9	1749.8	0.05170	70.6	12957	174.6	245.2	60.9%
ftv55	5000	1583.9	1585.6	0.04191	101.7	38867	940.5	1042.1	-0.6%
ftv55	10000	1583.9	1584.7	0.01919	205.7	20	16.8	222.5	78.5%
ftv64	5000	1807.3	1813.9	0.03699	161.8	18252	584.4	746.2	73.0%
ftv64	10000	1807.3	1810.6	0.01694	309.7	9876	349.9	659.5	76.1%
ftv70	5000	1908.4	1910.3	0.02956	198.0	36051	1215.8	1413.8	62.0%
ftv70	10000	1908.4	1909.6	0.01351	409.9	35	42.2	452.2	87.8%
p43	1582	5578.3	5633.8	0.00717	28.8	175194	6424.6	6453.4	-204.7%
p43	5000	5580.1	5625.1	0.00420	76.7	86710	3041.5	3118.3	-47.3%
ry48p	3328	14280.1	14422.9	0.00455	47.0	13181	187.7	234.8	60.9%
ry48p	5000	14280.1	14393.0	0.00351	66.3	6625	87.9	154.3	74.3%
ft53	5000	6904.0	6948.9	0.01414	82.5	1572	38.9	121.4	85.3%
ft53	7243	6904.0	6945.0	0.01000	127.2	74	14.7	141.9	82.8%
ft70	5000	38640.1	8831.2	0.03572	208.9	32930	983.4	1192.3	67.4%
ft70	10000	38640.1	38735.1	0.01637	432.3	236	46.3	478.6	86.9%

redução no número de subcircuitos. Por exemplo, para o problema ftv70, exibimos o número de subcircuitos encontrados ao longo das 1000 iterações do AV na Figura 3.

5.2 Resolução do modelo inteiro

Nesta subsecção, procuramos averiguar a adequação do AV como mecanismo de identificação de arcos que não participam na solução óptima inteira. Mais precisamente, com a solução resultante da aplicação do algoritmo volumétrico foram fixadas a zero todas as variáveis cujo valor da solução primal seja igual ou inferior a 0.0005 e cujo custo reduzido associado à restrição dual correspondente seja superior ou igual a 0.5. Depois, executou-se o CPLEX MIP com base na formulação de fluxo desagregado. Aplicámos esta ideia apenas a algumas instâncias estudadas anteriormente. Os resultados são apresentados na Tabela 3.

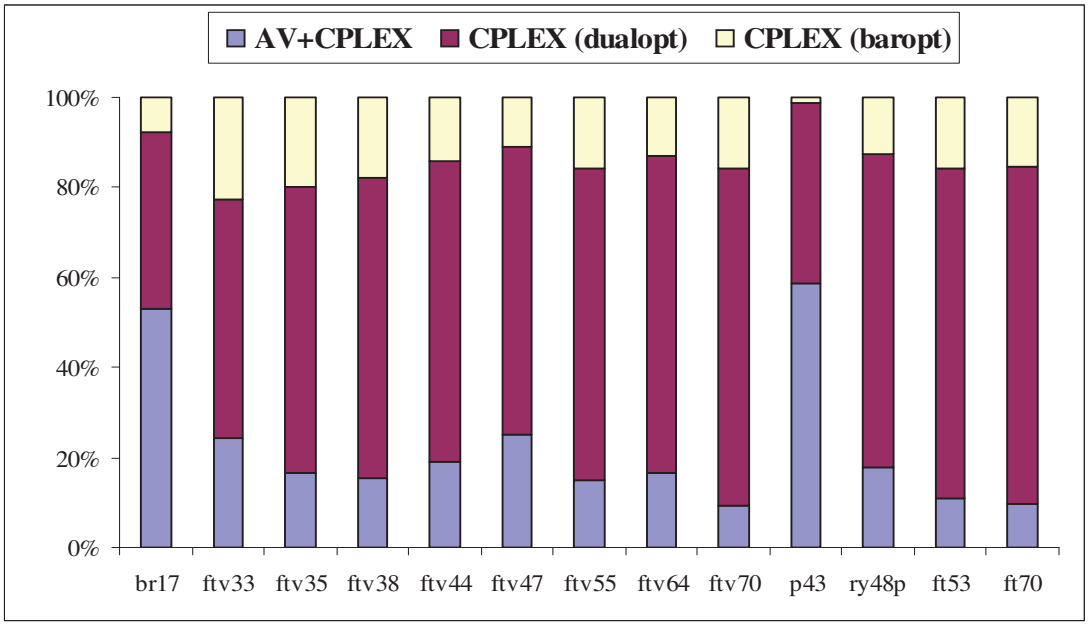

Figura 2: Comparação de tempos referidos nas Tabelas 1 e 5.1.

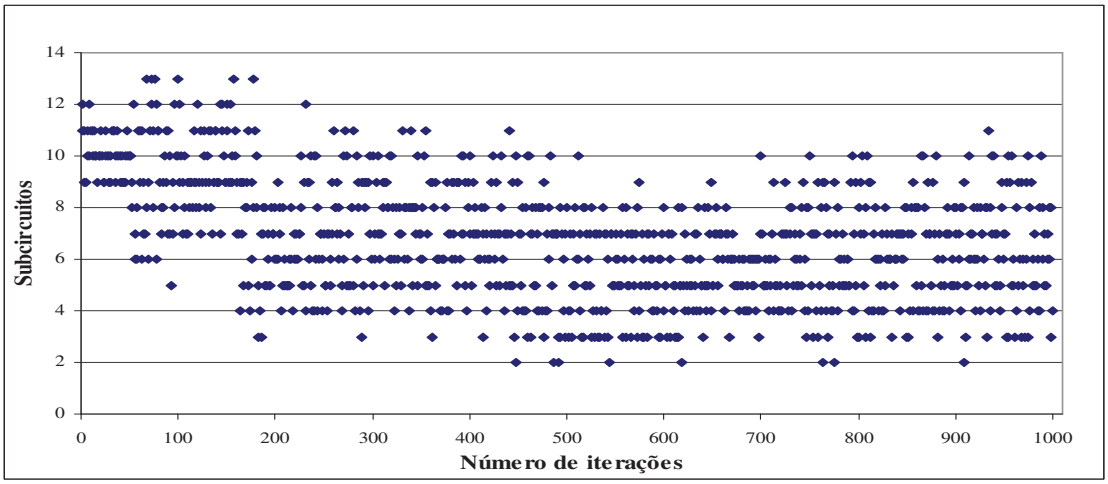

Figura 3: Número de subcircuitos para o Problema ftv70.

A primeira parte da Tabela 3 repete informação apresentada na Tabela 5.1. A segunda parte inclui os resultados da aplicação do CPLEX MIP ao problema reduzido, respectivamente, o número de iterações executadas e o tempo de resolução. A penúltima coluna diz respeito ao tempo total no cálculo da solução inteira, ou seja, a soma do tempo na aplicação do AV adicionado do tempo da aplicação do CPLEX MIP. A coluna final indica o ganho de tempo relativamente à aplicação do CPLEX MIP apenas. Em todas estas instâncias obteve-se o mesmo valor óptimo que o problema original (sem a fixação prévia de variáveis).

Comparando estes resultados com os obtidos na Tabela 1, podemos verificar que para o

Tabela 3: Resultados obtidos com o CPLEX MIP após a execução do AV.

PCVA ID	VOLUMÉT.		CPLEX MIP		Tempo Total	Tempo Ganho
	AV It.	Tempo (seg.)	Simplex It.	Tempo (seg.)		
br17	500	0.7	48837	158.4	159.1	1.9%
br17	1000	1.4	35331	121.4	122.8	24.3%
ftv33	500	2.8	2508	1.1	3.9	93.1%
ftv33	1000	5.5	2879	1.4	6.8	88.0%
ftv35	500	3.2	5028	27.9	31.1	98.5%
ftv35	1000	6.4	4106	24.4	30.8	98.5%
ftv38	500	3.6	5997	36.9	40.5	99.8%
ftv38	1000	7.1	11662	64.0	71.2	99.7%

problema br17 o tempo despendido pelo CPLEX MIP na obtenção da solução inteira foi grande relativamente ao requerido pelo AV. Mesmo assim, após 1000 iterações do AV e fixando 1071 variáveis (dum total de 4624 variáveis) conseguiu-se um ganho de tempo de 24.3%. Para os outros problemas conseguiram-se ganhos de tempo na ordem dos 90%, como podemos verificar de seguida. Para o problema ftv33, após fixar a zero 10709 dum total de 38148 variáveis, foi encontrada a solução inteira óptima num tempo total de 3.9s, que é inferior ao tempo despendido quando se utiliza apenas o CPLEX MIP que foi de 56.8s. Isto mostra, novamente, que o AV acelerou o processo de encontrar a solução. Porém, como era de esperar, na resolução da mesma instância com 1000 iterações do AV, que resultou em fixar a zero 7779 variáveis, não se verificou uma melhoria no tempo total. Para o problema ftv35, com 500 iterações do AV (fixando a zero 5990 dum total de 45360 variáveis) foi encontrada a solução óptima inteira. Para a instância ftv38, executando 500 iterações do AV (fixando a zero 6972 dum total de 57798 variáveis) foi encontrada a solução óptima inteira. De notar, que o tempo total despendido na obtenção desta solução foi muito menor que o obtido através da execução apenas do CPLEX MIP no problema completo.

5.3 Resolução por métodos heurísticos

Nesta subsecção, descrevemos experiências computacionais obtidas com a aplicação de várias heurísticas aproveitando os custos reduzidos gerados no decurso do AV. Além das instâncias apresentadas anteriormente, foram consideradas outras de maior dimensão, umas obtidas na TSPLib e outras geradas aleatoriamente através dum código disponível na página web de *DIMACS Implementation Challenge* [24]. As instâncias geradas pertencem à classe denominada *Random Asymmetric Matrices Closed under Shortest Paths* (TMAT). Nesta classe as instâncias são geradas através do gerador de matrizes de distâncias assimétricas aleatórias que escolhe cada distância $d(i,j)$ como um valor inteiro aleatório x, tal que $0 \leq x \leq 10^6$. Depois, para cada distância se $d(i,j) > d(i,k) + d(k,j)$ então $d(i,j) = d(i,k) + d(k,j)$ e repete-se até não se poderem fazer mais trocas. De notar, que estas instâncias consideram a desigualdade triangular das distâncias. Na Tabela 4, encontra-se uma descrição das instâncias retiradas da TSPLib (kro124p e ftv100 até ftv170), incluindo o valor da solução óptima inteira conhecida, e das instâncias geradas aleatoriamente (tmat100 até tmat200).

Considerámos uma redefinição dos custos $cr_{ij} \equiv c_{ij} - (u_i + v_j)$, onde u e v denotam as

variáveis duais óptimas no problema de afectação de uma dada iteração do AV. De 100 em 100 iterações, executamos uma heurística que vai permitir obter um circuito Hamiltoniano com base nessa estrutura de custos. No final do AV, observamos o melhor limite superior encontrado. Testámos heurísticas de construção e a heurística de Lin-Kernighan tal como é implementada em [21].

Tabela 4: Problemas de dimensões superiores.

PCVA			
ID	$\|V\|$	$\|A\|$	Int. óptimo
kro124p	100	9900	36230
ftv100	101	10100	1788
ftv110	111	12210	1958
ftv120	121	14520	2166
ftv130	131	17030	2307
ftv140	141	19740	2420
ftv150	151	22650	2611
ftv160	161	25760	2683
ftv170	171	29070	2755
tmat100	100	9900	
tmat110	110	11990	
tmat120	120	14280	
tmat130	130	16770	
tmat140	140	19460	
tmat150	150	22350	
tmat160	160	25440	
tmat170	170	28730	
tmat180	180	32220	
tmat190	190	35910	
tmat200	200	39800	

5.3.1 Heurísticas de construção

Testámos quatro das heurísticas que estão disponíveis no código TSP_SOLVE [22], nomeadamente, as heurísticas *addition*, *assign*, *loss* e *patching*. A heurística *addition*, descrita em [23], funciona de modo semelhante à heurística do vizinho mais próximo. As heurísticas *assign* e *patching* funcionam de modo semelhante, nomeadamente, usam uma solução do correspondente problema de afectação, que define um conjunto de subcircuitos, para definir um circuito Hamiltoniano de forma heurística. A heurística *patching*, foi inicialmente apresentada por Karp e Steele em [25], é também denominada por *Karp-Steele Patching* (KSP) em [13]. A heurística *loss* é uma implementação do método de Loss ([9]), que foi inicialmente proposta para o problema do caixeiro viajante simétrico e posteriormente adaptada para o caso assimétrico. A Tabela 5 reproduz os resultados obtidos para todos os problemas da TSPLIB e da classe TMAT, anteriormente apresentados. Nessa tabela apresentamos, para cada instância, o número total de iterações do AV, e para cada heurística o melhor limite superior, denotado LS, encontrado. Indicamos, também, a primeira iteração em que se encontrou esse LS. Os tempos de execução das heurísticas são negligenciáveis.

Tabela 5: Soluções obtidas com as heurísticas do TSP_SOLVE.

PCVA	AV	addition		assign	loss		patching	
ID	It.	LS	It.	LS	LS	It.	LS	It.
br17	1000	39	100	39	39	100	39	100
ftv33	1000	1482	200	1286	1372	600	1409	200
ftv35	1000	1491	100	1473	1508	300	1489	200
ftv38	1000	1634	200	1530	1547	700	1546	100
ftv44	5000	1733	3400	1613	1673	300	1699	100
ftv47	5000	1793	700	1776	1787	1600	1846	300
ftv55	5000	1781	3400	1608	1747	200	1657	100
ftv64	5000	2054	500	1839	1890	500	1871	900
ftv70	5000	2168	400	1950	2074	100	2004	600
p43	5000	5623	800		5638	1100	5640	200
ry48p	5000	14939	200	14422	15254	1000	14857	100
ft53	5000	8088	300	6905	7383	500	7847	100
ft70	5000	40566	500	38673	39065	1900	39197	100
kro124p	5000	40524	200	36230	41121	500	40106	100
ftv100	5000	2119	3000	1788	1969	900	1878	400
ftv110	5000	2335	800	1958	2156	4400	2036	500
ftv120	5000	2618	3000	2166	2402	100	2244	1100
ftv130	5000	2900	3300	2307	2519	100	2402	1000
ftv140	5000	2965	5000	2420	2590	300	2500	1200
ftv150	5000	3166	3400	2611	2971	200	2691	1500
ftv160	5000	3318	4200	2683	2839	100	2729	200
ftv170	5000	3509	4500	2755	2938	300	2809	600
tmat100	5000	1628486	1700	1377025	1379385	3500	1395601	500
tmat110	5000	1579919	200	1362004	1371800	700	1367459	100
tmat120	5000	1684696	2800	1390478	1396470	600	1419536	100
tmat130	5000	1716897	4500	1429864	1428864	3400	1444402	300
tmat140	5000	1719640	4800	1433486	1443556	300	1449585	100
tmat150	5000	1598899	3500	1364821	1364821	100	1389641	300
tmat160	5000	1855854	1300	1484992	1486164	2200	1487017	100
tmat170	5000	1674560	1600	1405055	1405273	4400	1405172	800
tmat180	5000	1941760	4400		1558217	500	1562008	100
tmat190	5000	1670791	500		1411658	1000	1424933	200
tmat200	5000	1968195	1400		1548893	400	1547919	100

Os LS encontrados para a heurística *assign* coincidem todos com os valores óptimos inteiros conhecidos e foram todos encontrados ao fim de 100 iterações do AV. Queremos também revelar que executando esta heurística com os custos originais do problema, os valores dos LS também coincidem com os valores óptimos inteiros. Para o problema p43, com a heurística *assign*, não se conseguiram obter quaisquer soluções após 5 horas de tempo de execução.

Para as heurísticas *addition, loss* e *patching*, só ao fim de um certo número de iterações é que se encontraram os melhores LS. Salientamos, também, que para estas heurísticas os LS obtidos com os custos originais do problema são piores que os encontrados com os custos reduzidos. Podemos observar, pela tabela, que os valores dos melhores LS, para a heurística

addition ficam, em geral, mais afastados dos valores determinados com as outras heurísticas.

Relativamente aos problemas da classe TMAT, verificamos que os melhores LS são obtidos com a heurística *assign*, apesar de não se ter conseguido obter resultados para as instâncias tmat180 até tmat200, porque, neste caso, o programa não permite resolver instâncias do PCVA com mais de 175 vértices. Nesta classe de instâncias, também se verifica que a heurística que apresenta piores resultados é a *addition*.

Na Figura 4, encontra-se um gráfico que mostra os diversos LS encontrados, com a heurística *addition*, para o problema ftv44, durante a execução de 5000 iterações do AV. Neste caso, o melhor LS foi encontrado quando se invocou a heurística na iteração 3400 e voltou a este valor na iteração 4800. De notar, que na iteração zero do AV, em que os custos reduzidos coincidiam com os custos originais do problema, o LS tinha o valor de 1829. Por isso, podemos concluir que os custos reduzidos, gerados pelo AV, ajudam a encontrar melhores LS.

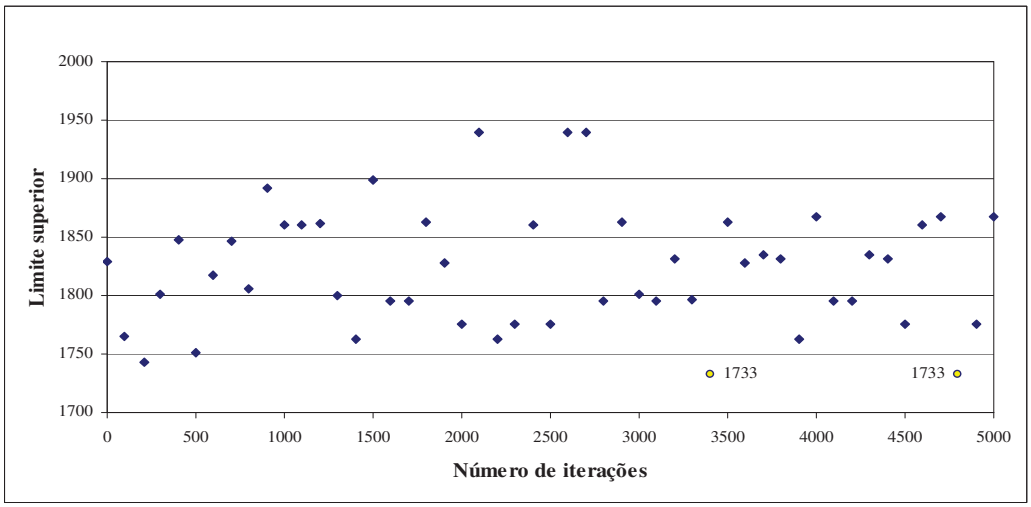

Figura 4: Limites superiores determinados pela heurística *addition* para o problema ftv44.

5.3.2 Heurística de Lin-Kernighan modificada por Helsgaun

Keld Helsgaun descreve em [21] a implementação de uma versão modificada do algoritmo de Lin-Kernighan [28]. A nova implementação difere da original em vários detalhes. A principal diferença encontra-se na estratégia de procura. O novo algoritmo usa passos de procura maiores e mais complexos que o original. Usa também a análise de sensibilidade para direccionar e restringir a procura. As experiências computacionais que efectuámos vieram confirmar o reconhecido sucesso desta heurística. Nomeadamente, os circuitos encontrados com esta heurística para os problemas da TSPLIB (usando os custos originais) são sempre óptimos, excepto para o problema p43 que encontra um circuito de valor 5621.

Tal como fizemos para as heurísticas da subsecção anterior, executámos esta heurística com os custos reduzidos, em cada 100 iterações do AV. Apresentamos na Tabela 6 os valores dos LS encontrados. Para as instâncias da TSPLIB, os circuitos encontrados pelo heurística são sempre óptimos. Para os vários problemas, os melhores LS foram encontrados na iteração 100

do AV, excepto para os problemas ftv150, ftv170 e p43, que foram encontrados, respectivamente, nas iterações 400, 500 e 400. Tal como nos casos anteriores, os tempos de execução da heurística são negligenciáveis.

Tabela 6: Soluções obtidas com a heurística de Helsgaun.

PCVA ID	LS	PCVA ID	LS	PCVA ID	LS
br17	39	kro124p	36230	tmat100	1377025
ftv33	1286	ftv100	1788	tmat110	1362004
ftv35	1473	ftv110	1958	tmat120	1390478
ftv38	1530	ftv120	2166	tmat130	1429864
ftv44	1613	ftv130	2307	tmat140	1433486
ftv47	1776	ftv140	2420	tmat150	1364821
ftv55	1608	ftv150	2611	tmat160	1484992
ftv64	1839	ftv160	2683	tmat170	1405055
ftv70	1950	ftv170	2755	tmat180	1554371
p43	5620			tmat190	1411658
ry48p	14422			tmat200	1545211
ft53	6905				
ft70	38673				

6 Conclusões

Este estudo incidiu, essencialmente, na resolução da relaxação linear subjacente à formulação de fluxo desagregado para o problema do caixeiro viajante assimétrico através do algoritmo volumétrico.

O nosso estudo mostrou que vale a pena utilizar o algoritmo volumétrico como mecanismo de identificação de uma base inicial para posterior aplicação de um método dual-simplex. Em geral, um número maior de iterações do algoritmo volumétrico corresponde a um menor número de iterações do método dual-simplex (CPLEX). Quase sempre o tempo total de execução desta estratégia combinada é bastante inferior ao tempo despendido na execução do método dual-simplex. Os casos em que este comportamento não foi observado devem-se a uma interrupção demasiado prematura do algoritmo volumétrico.

O algoritmo volumétrico foi bastante eficiente na identificação dos arcos que não participam na solução óptima inteira, para as instâncias pequenas que se testaram. Com a consequente redução no tamanho do problema o método *branch-and-bound* (CPLEX MIP) requereu muito menos tempo de execução. Para o futuro, pretendemos testar esta estratégia com instâncias de dimensão maior da TSPLib.

Os custos reduzidos gerados no decurso do algoritmo volumétrico ajudaram a melhorar o comportamento de certas heurísticas. Verificámos, ainda, que para a instância p43, a abordagem combinada permitiu encontrar uma solução inteira de melhor qualidade do que aquela que é obtida com a implementação de Helsgaun da heurística de Lin-Kernighan (usando apenas os custos originais), que é considerada como um heurística muito eficiente.

Com base nas experiências efectuadas, pensamos poder afirmar que o algoritmo volumétrico permite acelerar consideravelmente a resolução do problema do caixeiro viajante assimétrico.

Agradecimentos

Os autores agradecem os comentários e sugestões do revisor que contribuíram para melhorar e clarificar alguns aspectos do artigo.

7 Referências

[1] R. Anbil, J. J. Forrest, and W. R. Pulleyblank. Column generation and the airline crew pairing problem. *Documenta Mathematica*, Extra Volume ICM III:677–686, 1998.

[2] L. Bahiense, F. Barahona, and O. Porto. Solving steiner tree problems in graphs with lagrangian relaxation. Journal of Combinatorial Optimization, 3(7):259–282, 2003.

[3] L. Bahiense, N. Maculan, and C. Sagastizábal. The volume algorithm revisited: Relation with bundle methods. Mathematical Programming, 94:41–70, 2002.

[4] F. Barahona and R. Anbil. The volume algorithm: Producing primal solutions with a subgradient method. Mathematical Programming, 87:385–399, 2000.

[5] F. Barahona and R. Anbil. On some difficult linear programs coming from set partitioning. Discrete Applied Mathematics, 118(1-2):3–11, 2002.

[6] F. Barahona and F. A. Chudak. Near-optimal solutions to large scale facility location problems. Technical report, IBM Watson Research Center, 1999.

[7] F. Barahona and L. Ladanyi. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. Technical report, IBM Watson Research Center, 2001.

[8] A. Claus. A new formulation for the travelling salesman problem. SIAM Journal of Algebraic and Discrete Methods, 5:21–25, 1984.

[9] P. V. D. Cruyssen and M. Rijckaert. Heuristic for the asymmetric travelling salesman problem. Journal of the Operational Research Society, 29(7):697–701, 1978. 18A. M. Rocha, E. M. G. P. Fernandes, J. Soares / Investigação Operacional, 25 (2005) 1-19

[10] G. Dantzig, D. Fulkerson, and S. Johnson. Solution of a large-scale traveling salesman problem. Operations Research, 2:393–410, 1954.

[11] M. Fischetti and P. Toth. A polyhedral approach to the asymmetric traveling salesman problem. Management Science, 43(11):1520–1536, 1997.

[12] B. Gavish and S. Graves. The traveling salesman problem and related problems. Technical report, Operations Research Center, MIT, 1978. Working Paper OR-078-78.

[13] F. Glover, G. Gutin, A. Yeo, and A. Zverovich. Construction heuristics for the asymmetric TSP. European Journal of Operational Research, 129:555–568, 2001.

[14] L. Gouveia and J. M. Pires. Uma análise comparativa de formulações para o problema do caixeiro viajante assimétrico. Investigação Operacional, 16:89–114, 1996.

[15] L. Gouveia and J. M. Pires. The asymmetric travelling salesman problem and a reformulation of the miller-tucker-zemlin constraints. European Journal of Operational Research, 112(1):134–146, 1999.

[16] L. Gouveia and J. M. Pires. The asymmetric travelling salesman problem: on generalizations of disaggregated Miller-Tucker-Zemlin constraints. Discrete Applied Mathematics, 112(1-3):12–145, 2001.

[17] G. Gutin and A. P. Punnen, editors. The traveling salesman problem and its variations, volume 12 of Combinatorial Optimization. Kluwer Academic Publishers, Dordrecht, 2002.

[18] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees. Operations Research, 18:1138–1162, 1970.

[19] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. Mathematical Programming, 1:6–25, 1971.

[20] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. Mathematical Programming, 6:62–68, 1974.

[21] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman problem. European Journal of Operations Research, 126:106–130, 2000. Code available at http://www.akira.ruc.dk/ keld/research/LKH/.

[22] C. Hurwitz and R. Craig. GNU Tsp_solve. Version 1.3.8, 1994. Available at http://www.cs.sunysb.edu/ algorith/implement/tsp/distrib/tsp_solve/.

[23] D. Johnson and C. Papadimitriou. Performance guarantees for heuristics. In E. Lawler, J. Lenstra, A. R. Kan, and D. Shmoys, editors, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, pages 145–180. John Wiley & Sons, Chichester, 1985. A. M. Rocha, E. M. G. P. Fernandes, J. Soares / Investigação Operacional, 25 (2005) 1-1919

[24] D. Jonhson, L. McGeoch, F. Glover, and C. Rego. Website for the DIMACS implementation challenge on the traveling salesman problem. http://www.research.att.com/ dsj/chtsp.

[25] R. Karp and J. Steele. Probabilistic analysis of heuristics. In E. Lawler, J. Lenstra, A. R. Kan, and D. B. Shmoys, editors, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, pages 181–205. John Wiley & Sons, Chichester, 1985.

[26] A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman problem formulations. Operations Research Letters, 9:127–132, 1990.

[27] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. The traveling salesman problem. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Ltd., Chichester, 1990. A guided tour of combinatorial optimization, Reprint of the 1985 original, A Wiley-Interscience Publication.

[28] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. Operations Research, 21:498–516, 1973.

[29] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of traveling salesman problems. Journal of ACM, 7:326–329, 1960.

[30] M. Padberg and T. Sung. An analytical comparison of different formulations of the travelling salesman problem. Mathematical Programming, 52:315–357, 1991.

[31] J. M. O. Pires. Formulações para o problema do caixeiro viajante assimétrico e sua aplica ção a um problema de desenho de redes com topologia em forma de anel. PhD thesis, Universidade de Lisboa, Setembro 2001.

[32] G. Reinelt. TSPLIB - a traveling salesman problem library. ORSA Journal on Computing, 3:376–384, 1991. Available at http://www.iwr.uniheidelberg. de/groups/comopt/software/TSPL IB95.

[33] R. Wong. Integer programming formulations of the travelling salesman problem. pages 149–152. Proceedings of the IEEE International Conference of Circuits and Computers, 1980.