Problema do Caixeiro Viajante com Entrega e Coleta Sobre Carregamento LIFO: Uma Abordagem Computacional

Afonso H. Sampaio, Sebastián A. Urrutia
Departamento de Ciência da Computação - Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, Pampulha, Belo Horizonte, MG CEP 31270-901, Brasil.
{afonsohs, surrutia}@dcc.ufmg.br

RESUMO

Nesse trabalho, tratamos do Problema do Caixeiro Viajante com Coleta e Entrega Sobre Carregamento LIFO (*Last-in First-out*) sem capacidades. O objetivo é analisar uma implementação para os métodos de separação das inequações de um modelo em programação inteira proposto por Cordeau et al. (2010) que modelam as restrições deste problema, com um número exponencial de desigualdades, dentro de algoritmos de planos de cortes. Mostramos que os critérios utilizados para a inclusão de desigualdades violadas no modelo e a hierarquia dos métodos de separação podem melhorar substancialmente o desempenho dos algoritmos no que diz respeito ao tempo de execução para determinar um limite inferior para o problema.

PALAVRAS CHAVE. Caixeiro Viajante, Restrições de Carregamento, Planos de Corte, Métodos de Separação

ABSTRACT

In this work, we address the uncapacitated Travelling Salesman Problem with Pickup and Delivery with LIFO Loading. The main goal is to study the implementation details in the separation methods for the inequalities used in a integer programming model proposed by Cordeau et al. (2010) to formulate this problem, using a exponential number of inequalities, together with cutting planes algorithms. We show that some implementation decisions can substantially improve the performance of the algorithms concerning the execution time until achieve a lower bound to the model.

KEYWORDS. Travelling Salesman, Loading Constraints, Cutting Plane, Separation Methods

1. Introdução

Um dos problemas em otimização combinatória mais conhecidos e extensivamente estudados é o Problema do Caixeiro Viajante (TSP - *Travelling Salesman Problem*). Entre suas diversas aplicações está o problema de roteamento de veículos (Iori e Martello, 2010). Uma variação desse problema consiste de um único veículo que deve servir a um conjunto de requisições compostas por um ponto de origem, onde um determinado item é coletado, e por um ponto de destino, onde esse mesmo item deve ser entregue. O Problema do Caixeiro Viajante com Coleta e Entrega (TSPPD - *Travelling Salesman Problem with Pcikup and Delivery*) consiste, então, em determinar um caminho que visite cada um dos pontos uma única vez, partindo de um depósito inicial e chegando num depósito final, de forma que cada ponto de coleta seja visitado antes de seu correspondente ponto de entrega, com o menor custo possível. Nesse trabalho, pretendemos analisar uma variante desse problema denominada TSPPD com restrições de carregamento *Last-in First-out* (LIFO), no qual o veículo possui uma (TSPPDL) ou múltiplas (TSPPDMS) entradas independentes para o carregamento dos itens. Um item quando carregado nesse veículo é colocado no topo de uma das pilhas e um item só pode ser entregue se ele estiver no topo de uma das pilhas de carregamento de itens do veículo. Nesse trabalho, consideraremos apenas o caso em que uma pilha de carregamento está disponível (TSPPDL) e que a pilha de carregamento tem capacidade ilimitada.

Aplicações surgem, por exemplo, no roteamento de veículos de transporte onde os itens transportados são carregados e descarregados pela parte traseira do veículo. A realocação dos itens durante uma operação de carregamento ou descarregamento pode ser proibitiva para itens pesados ou frágeis, ou aqueles cujo manuseio é perigoso. Dai a restrição que somente o último item carregado em uma pilha é acessível e, consequentemente, ele deve ser entregue antes de todos os demais itens que foram carregados antes dele. Ladany e Mehrez (1984) apresentam uma aplicação real em uma companhia de expedição em Israel. Levitin e Abezgaouz (2003) mostram o problema num contexto onde um veículo guiado automaticamente é utilizado para mover itens entre diversos pontos de um armazém.

Recentemente, Cordeau et al. (2010) propuseram 3 formulações em programação inteira para o TSPPDL, duas para o caso capacitado e uma para o caso sem capacidades, embora o foco do trabalho seja no problema sem capacidades. Os autores afirmam que são os primeiros modelos apresentados na literatura. Basicamente, o modelo sem capacidades consiste na formulação clássica do TSP com restrições de *eliminação de subciclos*, de um conjunto de restrições para impor as relações de precedência entre os pontos de coleta e entrega, e de um conjunto exponencial de restrições proposto pelos autores para impor a política de carregamento LIFO do veículo. As formulações são fortalecidas através de uma série de desigualdades válidas, algumas que são válidas para o TSPPD - e, portanto são válidas para o TSPPDL - e outras propostas pelos autores observando a estrutura particular do TSPPDL.

Nesse trabalho, abordamos apenas o TSPPDL sem capacidades. O objetivo é analisar, do ponto de vista computacional, uma implementação para os métodos de separação das desigualdades do modelo em programação inteira proposto por Cordeau et al. (2010) que modelam as restrições deste problema. Embora o trabalho dos autores contenha um vasto material detalhando as desigualdades válidas e os procedimentos de separação propostos, além de um estudo computacional avaliando

as contribuições de cada desigualdade na obtenção de um limite inferior para as formulações baseado na relaxação linear dos mesmos, poucos detalhes sobre os critérios utilizados na implementação das ideias propostas são mencionados. Algumas decisões de implementação influenciam sobremaneira o comportamento do algoritmo de geração de cortes. Em particular, estratégias diferentes para a inclusão dos cortes gerados, ou a ordem de inclusão destes cortes no modelo relaxado, tem impacto considerável no desempenho do algoritmo: o número de cortes violados computados com os métodos de separação pode variar bastante, assim como o tempo de execução do algoritmo até obter um limite inferior para o modelo.

No restante do trabalho, ilustramos o modelo de programação inteira para o TSPPDL apresentado por Cordeau et al. (2010) na seção 2. A seção 3 mostra os métodos de separação exatos das desigualdades desse modelo dentro de diferentes algoritmos de geração de planos de corte, com uma breve análise de alguns aspectos que surgem na implementação desses métodos. Analisamos 5 estratégias diferentes para geração e inclusão dos cortes no modelo relaxado e, na seção 4, apresentamos um estudo computacional sobre o desempenho de cada estratégia. Apresentamos nossas conclusões e trabalhos futuros na seção 5.

2. Modelo em Programação Inteira

Utilizando terminologia de teoria dos grafos, o problema pode ser descrito formalmente como a seguir (Côté et al., 2012). Seja $G(V, A)$ um grafo completo, direcionado, onde A é o conjunto de arcos, $V = P \cup D \cup \{0, 2n+1\}$, $P = \{1, ..., n\}$ é o conjunto de pontos de coleta, $D = \{n+1, ..., 2n\}$ é o conjunto de pontos de entrega, considerando que o item coletado no ponto $i \in P$ deve ser entregue no ponto $n+i \in D$. Os pontos 0 e $2n+1$ correspondem aos depósitos inicial e final, respectivamente, os quais podem ser iguais. A cada arco $(i, j) \in A$ está associado um custo c_{ij}. Um único veículo com um conjunto $K = \{1, ..., k\}$ de pilhas para o carregamento dos itens é utilizado para atender a todas as requisições. Além disso, seja $\pi(i)$ a ordem com que o item no ponto $i \in V$ é coletado/entregue pelo veículo e $T_j = \{i_1, ..., i_k\}$ o conjunto dos itens no topo de cada uma das pilhas quando o veículo chega ao ponto $j \in V$. O objetivo é atender a todas as requisições percorrendo um caminho hamiltoniano de menor custo entre os depósitos 0 e $2n+1$ satisfazendo às restrições de precedência, $\pi(i) < \pi(n+i)\ \forall i \in P$, e de carregamento, $j \in T_j\ \forall j \in D$.

Uma formulação em programação inteira para o problema pode, então, ser descrita da seguinte forma: a cada arco $(i, j) \in A$ é associado uma variável $x_{ij} \in \{0, 1\}$, onde $x_{ij} = 1$ se o ponto $j \in V$ é visitado imediatamente depois do ponto $i \in V$. Seja $\mathcal{S}$ a coleção de todos os conjuntos $S \subset V$ com $0 \in S$, $2n+1 \notin S$ tal que existe um $i \in P$ com $i \notin S$ e $n+i \in S$, e Ω a coleção de todos os conjuntos $S \subset P \cup D$ tais que existe ao menos um $j \in P$ tal que $j \in S$ e $n+j \notin S$ ou $n+j \in S$ e $j \notin S$. A formulação proposta por Cordeau et al. (2010) é apresentada na formulação 1 (observe que há uma única pilha de carregamento, com capacidade ilimitada).

As restrições (2), (3), (4) são as restrições clássicas para estabelecer um caminho hamiltoniano com os pontos em V, começando por 0 e terminando em $2n+1$. As restrições (5) foram introduzidas por Balas et al. (1995) no contexto do problema do Caixeiro Viajante com Precedências (TSPP), e modelam as relações de precedências entre os pontos de coleta e entrega. As restrições (6) são as restrições introduzidas pelo autores da formulação para impor as restrições de carregamento LIFO utilizando apenas as variáveis originais x_{ij}. Além disso, os autores apresentam algumas famílias

Formulação 1 Modelo em programação inteira para o TSPPDL

$$\text{(TSPPDL) minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$\text{s. t.} \quad \sum_{j\neq i} x_{ij} = 1 \qquad \forall i \in P \cup D \cup \{0\} \tag{2}$$

$$\sum_{j\neq i} x_{ji} = 1 \qquad \forall i \in P \cup D \cup \{2n+1\} \tag{3}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq P \cup D, |S| \geq 2 \tag{4}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{5}$$

$$\sum_{j\in S} x_{ij} + \sum_{i,j\in S} x_{ij} + \sum_{j\in S} x_{j,n+i} \leq |S| \qquad \forall S \in \Omega, \forall i, n+i \notin S, i \in P \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{7}$$

de restrições válidas para fortalecer essa formulação. Utilizam métodos de separação exatos - com base na solução de uma série de problemas de fluxo máximo - para identificar restrições violadas de subtour (4), de precedência (5) ou de carregamento LIFO (6), além de métodos de separação heurísticos para algumas famílias de restrições propostas. Todos esses resultados são utilizados dentro de um algoritmo *branch-and-cut* capaz de resolver instâncias com até 17 requisições e algumas instâncias com 25 requisições. TSPPDL é um problema $\mathcal{NP}$-Difícil, como mostrado em Cordeau et al. (2010).

3. Algoritmo de Planos de Corte

Dantizig, Fulkerson e Johnson (Chvátal et al., 2010) introduziram um método de planos de corte para o TSP criando e resolvendo uma sequência de relaxações lineares para o problema original. A ideia fundamental consiste em obter uma solução para a relaxação linear do problema original e utilizá-la para melhorar o limite (inferior) fornecido pelas relaxações seguintes. O modelo relaxado para o TSPPDL, TSPPDLR, consiste de (1) − (3), onde as variáveis inteiras são relaxadas (i.e. $0 \leq x_{ij} \leq 1$) e de um conjunto $A\mathbf{x} \leq \mathbf{b}$ de restrições satisfeitas por todas as soluções do TSPPDL.

Se $\mathbf{x}^*$ é uma solução ótima para o modelo relaxado e $\mathbf{a}'\mathbf{x}^* > b$, então a relaxação linear pode ser melhorada se adicionarmos ao modelo a restrição $\mathbf{a}'\mathbf{x} \leq b$ satisfeita por todas as soluções do problema TSPPDL. $\mathbf{a}'\mathbf{x} \leq b$ é um plano de corte e sua adição ao modelo elimina $\mathbf{x}^*$ do conjunto de soluções na relaxação linear e, possivelmente, a solução da relaxação linear seguinte terá um limite melhorado. Para mais sobre métodos de planos de corte e separação das restrições relacionadas ao TSP, veja Applegate et al. (2007).

O *grafo suporte*, G^*, de uma solução $\mathbf{x}^*$ para modelo relaxado é um grafo com vértices V e arcos $A_r = \{(i,j) \in A : x^*_{ij} > 0\}$ com capacidades $w_{ij} = x_{ij}$. A ideia básica para identificar desigualdades violadas consiste na resolução de problemas de fluxo máximo $s - t$ no grafo G^*. Para as desigualdades (4), $2n$ problemas são resolvidos: n com origem em $s = 0$ e destino $t = i \in P$ e outros n com origem em

$s = n + i \in D$ e destino em $t = 2n + 1$. Se o valor de fluxo é menor do que 1, então um conjunto $S \subset P \cup D$ pode ser obtido, tal que S viola a desigualdade (4). Algumas modificações no grafo G^* são necessárias para encontrar conjuntos $S \in \mathcal{S}$ e $S \in \Omega$, para definir restrições (5) e (6), respectivamente. Em particular, para as desigualdades (5) n problemas de fluxo, com origem em 0 e destino em $2n + 1$, são solucionados, um para cada $i \in P$. Para as desigualdades (6), o processo consiste da solução de dois problemas de fluxo para cada par $i, j \in P$ (um para identificar $S \not\ni i, n + i$ tal que $\exists j \in P \cap S$ e $n + j \notin S$ e, caso não seja encontrado, $S \not\ni i, n + i$ tal que $\exists n + j \in D \cap S$ e $j \notin S$). Veja Cordeau et al. (2010) para os detalhes das modificações propostas no grafo G^*.

3.1. Implementação

Na implementação do algoritmos de planos cortes, $2n$, n e $O(n^2)$ problemas de fluxos são resolvidos no intuito de identificar desigualdades (4), (5) e (6), respectivamente, violadas por uma dada solução $\mathbf{x}^*$. Portanto, potencialmente uma grande quantidade de cortes pode ser gerada. Os critérios para decidir quais deles acrescentar ao modelo relaxado antes da reotimização pode alterar drasticamente o desempenho do algoritmo. Além disso, uma outra questão relevante diz respeito ao critério de seleção das classes de desigualdades a serem separadas antes de cada processo de reotimização do modelo: dado uma solução $\mathbf{x}^*$, podemos executar o algoritmo de separação para uma dada classe $\mathcal{C}$ de desigualdades, identificar cortes violados para esta classe, decidir quais deles acrescentar ao modelo e, então, decidir por identificar cortes de uma outra classe ou reotimizar o modelo e seguir com a separação das desigualdades da classe $\mathcal{C}$, até que o algoritmo de separação não possa mais identificar cortes dessa classe. A seguir, descrevemos os critérios de inclusão de cortes e as políticas de reotimização que avaliamos para o TSPPDL. Ao todo, implementamos algoritmos de planos de cortes com 5 estratégias diferentes, combinando cada um dos critérios e políticas.

3.1.1. Critérios de Seleção dos Cortes

Um critério comumente utilizado em implementações de algoritmos de planos de corte para o TSP clássico consiste em resolver uma série de problemas de fluxo máximo, variando-se o destino do fluxo t, para obter conjuntos S_{st} que determinam desigualdades de eliminação de subciclos violadas. Aquele conjunto S_{st} que define o corte mínimo de menor capacidade, entre todas as escolhas de t, define, então, o corte mínimo global e é adicionado ao modelo relaxado. Para o TSPPDL, seguindo essas ideias, um critério semelhante seria resolver os problemas de fluxo para identificar desigualdades (4), (5) ou (6) violadas por uma solução $\mathbf{x}^*$ mas adicionar ao modelo TSPPDLR apenas a desigualdade de cada classe que fornecer o menor valor de fluxo.

Para as restrições (4) e (5), identificar a desigualdade mais violada é uma tarefa relativamente simples. Em particular, $2n$ e n problemas de fluxo devem ser resolvidos, respectivamente, para a separação dessas desigualdades. Por outro lado, a separação das desigualdades (6) requer a solução de $O(n^2)$ problemas de fluxo. Dessa forma, um outro critério adotado foi o de adicionar ao modelo somente a primeira desigualdade violada encontrada para cada classe. Com esse critério, um maior número de restrições é adicionado ao modelo relaxado, como mostraremos nos resultados computacionais.

Algoritmo 1: Plano de cortes para o TSPPDL com a política 1

1 Seja $A\mathbf{x} \leq \mathbf{b}$, um sistema linear satisfeito $\forall \mathbf{x}$ vetores solução do TSPPDL;
2 $\mathcal{P} \leftarrow \{A\mathbf{x} \leq \mathbf{b}\}$;
3 **enquanto** *Critério de parada não satisfeito* **faça**
4 $\quad \mathbf{x}^* \leftarrow \mathbf{x} \in \mathcal{P}$ que minimiza $\mathbf{c}'\mathbf{x}$;
5 $\quad A_1\mathbf{x} > \mathbf{b}_1$: desigualdades (4) violadas por $\mathbf{x}^*$;
6 $\quad A_2\mathbf{x} > \mathbf{b}_2$: desigualdades (5) violadas por $\mathbf{x}^*$;
7 $\quad A_3\mathbf{x} > \mathbf{b}_3$: desigualdades (6) violadas por $\mathbf{x}^*$;
8 $\quad$ Determine os cortes a serem inseridos no modelo, de acordo com um dos critérios de seleção de cortes: $\mathcal{P} \leftarrow \mathcal{P} \cap \{\mathbf{x} : \mathbf{a}_1'\mathbf{x} \leq b_1, \mathbf{a}_2'\mathbf{x} \leq b_2, \mathbf{a}_3'\mathbf{x} \leq b_3\}$

3.1.2. Políticas de Reotimização

Dado uma solução $\mathbf{x}^*$, podemos utilizar o procedimento de separação para cada classe de desigualdades $\mathcal{C}_i$ do TSPPDL, identificar os cortes de cada classe que separam $\mathbf{x}^*$ do conjunto de soluções de TSPPDLR e selecionar aqueles que serão inseridos no modelo relaxado. Alternativamente, podemos utilizar o método de separação de apenas uma classe $\mathcal{C}_i$, identificar as desigualdades violadas e definir o corte dessa classe que elimina a solução $\mathbf{x}^*$, incluí-la no modelo e reotimizá-lo. Enquanto o procedimento de separação consiga identificar desigualdades violadas pela solução do modelo, apenas os cortes de $\mathcal{C}_i$ são adicionados. Quando nenhuma desigualdade é violada pela solução $\mathbf{x}^*$ corrente, então executamos o procedimento de separação para outra classe de desigualdades.

Para o modelo do TSPPDL (1)-(7), temos as classes de desigualdades (4), (5) e (6). Uma restrição básica a ser atendida por qualquer solução do problema é que essa defina um grafo G^* conectado. As desigualdades (5) são um fortalecimento das restrições de eliminação de subciclo para $S \in \mathcal{S}$ (Balas et al., 1995) e, portanto, também contribuem para forçar uma solução $\mathbf{x}^*$ a ser um caminho conexo com os pontos em V. Por outro lado, um conjunto $S \in \Omega$ que não viole (6) ainda pode definir um subciclo (e portanto viola (4)). Dessa forma, e considerando a maior complexidade do método de separação das desigualdades (6), para a segunda política de adição de cortes, na qual apenas cortes de uma dada classe são adicionadas ao modelo enquanto o método de separação conseguir identificar desigualdades violadas, apenas o primeiro corte ou o corte mais violado para (6) é adicionado ao modelo. De fato, nos experimentos computacionais que executamos, seguir com o procedimento de separação, como para as classes (4) e (5), mostrou-se extremamente ineficiente do ponto de vista do tempo de execução.

Os algoritmos 1 e 2 ilustram as ideias aplicadas num método de planos de cortes para o TSPPDL considerando as classes de desigualdades (4), (5) e (6).

4. Experimentos Computacionais

Implementamos os algoritmos de planos de cortes 1 e 2 com cada um dos critérios de seleção de cortes (a primeira desigualdade violada encontrada ou a mais violada) em C++ utilizando o CPLEX 12.4 como solver de programação linear, e os executamos num Intel i5 3Ghz, 8GB, sobre a plataforma Ubuntu 12.04. O conjunto de instâncias utilizado é o mesmo utilizado por Cordeau et al. (2010) para avaliar o algoritmo *branch-and-cut* proposto pelos autores. Em particular, as instâncias foram criadas

Algoritmo 2: Plano de cortes para o TSPPDL com a política 2

Seja $A\mathbf{x} \leq \mathbf{b}$, um sistema linear satisfeito $\forall \mathbf{x}$ vetores solução do TSPPDL;
$\mathcal{P} \leftarrow \{A\mathbf{x} \leq \mathbf{b}\}$;
enquanto *Critério de parada não satisfeito* **faça**
  $\mathbf{x}^* \leftarrow \mathbf{x} \in \mathcal{P}$ que minimiza $\mathbf{c}'\mathbf{x}$;
  enquanto $\mathbf{x}^*$ viola alguma desigualdade (4) **faça**
    $A_1\mathbf{x} > \mathbf{b}$: desigualdades (4) violadas por $\mathbf{x}^*$;
    Determine o corte a ser inserido no modelo, de acordo com um dos critérios de seleção de cortes: $\mathcal{P} \leftarrow \mathcal{P} \cap \{\mathbf{x} : \mathbf{a}_1'\mathbf{x} \leq b_1\}$;
    $\mathbf{x}^* \leftarrow \mathbf{x} \in \mathcal{P}$ que minimiza $\mathbf{c}'\mathbf{x}$;
  enquanto $\mathbf{x}^*$ viola alguma desigualdade (5) **faça**
    $A_2\mathbf{x} > \mathbf{b}$: desigualdades (5) violadas por $\mathbf{x}^*$;
    Determine o corte a ser inserido no modelo, de acordo com um dos critérios de seleção de cortes: $\mathcal{P} \leftarrow \mathcal{P} \cap \{\mathbf{x} : \mathbf{a}_2'\mathbf{x} \leq b_2\}$;
    $\mathbf{x}^* \leftarrow \mathbf{x} \in \mathcal{P}$ que minimiza $\mathbf{c}'\mathbf{x}$;
  $A_3\mathbf{x} > \mathbf{b}$: desigualdades (6) violadas por $\mathbf{x}^*$;
  Determine o corte a ser inserido no modelo, de acordo com um dos critérios de seleção de cortes: $\mathcal{P} \leftarrow \mathcal{P} \cap \{\mathbf{x} : \mathbf{a}_3'\mathbf{x} \leq b_3\}$;

a partir de 9 instâncias da TSPLIB (Reinelt, 1991), onde os pontos de coleta e entrega foram atribuídos aleatoriamente, e possuem 9, 13, 17, 21 e 25 requisições (respectivamente 20, 28, 36, 44 e 52 vértices).

Antes de iniciar o algoritmo proposto por Cordeau et al. (2010), os autores incluem uma série de desigualdades $A\mathbf{x} \leq \mathbf{b}$, satisfeitas por todas soluções do TSPPDL, no modelo relaxado. Além disso, excluem as variáveis $x_{i,n+j}$, $i \neq j$, observando que um ponto de entrega $n + j \in D$ não pode ser sucessor direto do ponto de coleta $i \in P$ quando $i \neq j$. Utilizamos o mesmo conjunto de desigualdades iniciais nos algoritmos de planos de cortes que propomos, mas também observamos uma outra particularidade da estrutura do problema: o sucessor do ponto 0 e o antecessor do ponto $2n + 1$ deve ser, necessariamente, um ponto de coleta $i \in P$ e um ponto de entrega $n + j \in D$, respectivamente. Embora essas condições estejam implicadas pelas desigualdades de precedência (5), a inclusão das restrições $\sum_{j \in D} x_{0j} = 0$ e $\sum_{i \in P} x_{i,2n+1} = 0$ no modelo inicial contribuíram para o desempenho dos algoritmos de plano de cortes. Em particular, o limite inferior obtido com os algoritmos só não foi melhor após a inclusão destas restrições apenas para 3 instâncias (*att532*, *fnl4461* e *pr1002*).

O procedimento de identificação das desigualdades (6) violadas por uma solução $\mathbf{x}^*$ requer o cômputo de $O(n^2)$ problemas de fluxo máximo no grafo suporte G^*. Dessa forma, os algoritmos de planos de cortes que utilizam como critério de seleção de cortes a desigualdade mais violada entre todas identificadas pelo método de separação, apenas consideram um conjunto com no máximo 10 desigualdades (6) violadas e determinam a mais violada dentro desse conjunto.

Para a solução dos problemas de fluxo máximo, utilizamos a solução dada por um modelo de programação linear para este problema. Embora algoritmos especializados na solução deste problema (Goldberg e Tarjan, 1986) possibilitem a aplicação dos

métodos de separação para instâncias muito grandes, nos testes que executamos, as instâncias são relativamente pequenas e os ganhos foram pouco significativos. Os conjuntos S que determinam desigualdades violadas podem ser obtidos notando-se que o problema de corte mínimo $s-t$ é o dual do problema de fluxo máximo $s-t$.

Outro aspecto relevante na implementação de algoritmos de planos de cortes diz respeito à convergência do método. Em particular, num dado momento do algoritmo a inclusão dos cortes ao modelo relaxado pode não contribuir para um aumento considerável do limite inferior obtido. Dessa forma, para tratar os efeitos de *tailing-off*, o critério de parada dos algoritmos leva em consideração os aumentos no limite inferior para o modelo relaxado após cada iteração de adição de cortes. Se esse aumento é muito pequeno, o algoritmo para.

Na tabela 1, mostramos os tempos de execução para cada algoritmo de plano de cortes até alcançar um limite inferior para o modelo (1)-(7). Para cada instância, apresentamos a instância original da TSPLIB (*nome*), o número de requisições (*n)* e o limite superior (*UB)* utilizado por (Cordeau et al., 2010). Para cada algoritmo, reportamos o limite inferior obtido como $100 * (UB - LB)/UB$ na coluna *gap*, onde LB é o limite inferior obtido pelos algoritmos, e o tempo de execução em segundos na coluna *t(s)*. Colunas em branco na tabela ilustram casos em que o algoritmo de planos de corte em questão falhou em retornar um resultado antes de 3600 segundos. As estratégias 1 e 2 consistem dos algoritmos 1 e 2, respectivamente, onde o critério de seleção de cortes consiste apenas da primeira desigualdade violada encontrada pelo método de separação de uma dada classe de desigualdades. Nas estratégias 3 e 4, implementamos os algoritmos 1 e 2, respectivamente, com o critério de seleção de cortes definido como a desigualdade mais violada entre todas aquelas que o método de separação de uma dada classe pode identificar para uma dada solução $\mathbf{x}^*$. Na estratégia 5, utilizamos o critério de desigualdade mais violada e implementamos o algoritmo 1 mas, no lugar de escolher a desigualdade mais violada de cada classe, determinamos a desigualdade mais violada entre todas as desigualdades violadas encontradas, considerando todas as classes. Note que no método de separação das desigualdades (4), uma desigualdade é violada se o fluxo máximo é menor do que 1 e, para as desigualdades (5) e (6), se o fluxo é menor do que 2. Portanto, para a comparação da desigualdade mais violada na estratégia 5, consideramos a razão entre o valor de fluxo encontrado pelos métodos de separação das desigualdades (5) e (6) e o valor máximo de fluxo, 2.

Observe que nem todos os algoritmos conseguem retornar um limite inferior para todas as instâncias. Em particular, o algoritmo com a estratégia 1 não conseguiu retornar uma solução para a instância $d18112$, ao passo que a estratégia 2 pôde retornar um limite inferior para todas as instâncias. Embora os valores de *gaps* obtidos pelos algoritmos de plano de cortes com as estratégias 1 e 2 sejam próximos (médias de 7.80% e 8.05%, respectivamente, desconsiderando a instância não resolvida por 1), os tempos de execução para a estratégia 2 foram substancialmente menores (médias de $32,61s$ e $19.72s$). Para os algoritmos com as estratégias 3 e 4, os tempos de execução foram menores para 4. Além disso, o algoritmo com a estratégia 3 não conseguiu retornar soluções para 8 instâncias, enquanto o algoritmo com a estratégia 4 retorna um limite inferior para todas as instancias. Os resultados sugerem que, aparentemente, os algoritmos de planos de cortes têm um melhor desempenho quando a política de planos de cortes mostrada no algoritmo 2 é utilizada na implementação,

ou seja, ao incluir os cortes para as classes de desigualdades (4) e (5) e forçar as soluções do modelo relaxado a se aproximar de um caminho com todos os pontos $P \cup D$, começando em 0 e terminado em $2n + 1$, os algoritmos conseguem chegar a um limite inferior de qualidade semelhante para o modelo num menor tempo computacional. As diferenças no tempo de execução entre as estratégias 2 e 4 se justificam pela tarefa adicional para determinar a desigualdade mais violada encontrada pelo método de separação. Nesse sentido, com a estratégia 2, os limites inferiores obtidos tem qualidade semelhante aos obtidos com a estratégia 4 (médias de 8.05% e 8.01%, respectivamente), mas os tempos de execução são bem menores com a estratégia 2 (médias de $19.72s$ e $56.01s$, respectivamente). O algoritmo com a estratégia 5 obteve os piores resultados. Observe que limites inferiores foram obtidos apenas para 23 das 45 instâncias e, embora esses limites sejam próximos aos obtidos pelos demais algoritmos, o tempo de execução ainda foi superior.

Uma outra evidência que parece comprovar o melhor desempenho dos algoritmos de planos de cortes para o TSPPDL quando a política mostrada no algoritmo 2 é utilizada, está relacionada ao número de cortes de cada classe de desigualdades adicionado ao modelo por cada estratégia. Na tabela 2 mostramos o número de iterações de adição de cortes (*iter*) e a quantidade de desigualdades violadas de cada classe (*sec*, (4); *prec*, (5); *lifo*, (6)) adicionadas ao modelo relaxado para os algoritmos implementados com cada uma das estratégias. Observe que com as estratégias 1 e 3, os algoritmos de cortes adicionam muito mais restrições (5) e (6) ao modelo, enquanto que com as estratégias 2 e 4 o número de restrições (6) adicionadas é muito pequeno. Em particular, observe que apenas adicionando desigualdades (4) ao modelo, 3 instâncias (*fnl4461* e *pr1002* com $n = 9$, e *pr1002* com $n = 13$) foram solucionadas pelos algoritmos, isto é, a solução $\mathbf{x}^*$, com todas as variáveis $x_{ij} \in \{0, 1\}$, representa um caminho desde 0 até $2n+1$ passando por todos os pontos $P \cup D$, respeitando as precedências entre os pontos de coleta e entrega e também a política de carga e descarga do veículo. Por fim, considerando todas as classes de desigualdades, menos cortes no total são adicionados ao modelo pelo algoritmos com as estratégias 2 e 4.

5. Conclusões

Apresentamos algoritmos de planos de cortes para um problema de otimização combinatória para o qual apenas recentemente formulações em programação inteira foram elaboradas. Mostramos que algumas decisões de implementação influenciam sobremaneira o comportamento de um algoritmo de geração de cortes que utilize os métodos de separação exatos propostos por Cordeau et al. (2010). Em particular, estratégias diferentes para a inclusão dos cortes gerados, ou a hierarquia de utilização dos métodos de separação, tem impacto considerável no desempenho do algoritmo. Dado a maior complexidade do processo de separação das desigualdades (6), e uma vez que elas podem não contribuir para que os vetores solução $\mathbf{x}^*$ do modelo relaxado definam grafos G^* conectados, uma alternativa que parece favorecer uma implementação de um algoritmo de planos de cortes para o TSPPDL seria a de considerar as restrições (4) e (5) e utilizar um algoritmo *branch-and-cut* para alcançar uma solução $\mathbf{x}^*$ com entradas inteiras, para só então utilizar o método de separação para as desigualdades (6) e identificar violações na política de carregamento do veículo para essa solução.

Futuramente, pretendemos adicionar essas ideias dentro de um algoritmo *branch-and-cut* e fazer uma comparação com os resultados apresentados por Cordeau et al.

(2010).

Referências

Applegate, D. L., Bixby, R. E., Chvatal, V. e Cook, W. J. (2007), *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*, Princeton University Press, Princeton, NJ, USA.

Balas, E., Fischetti, M. e Pulleyblank, W. R. (1995), 'The precedence-constrained asymmetric traveling salesman polytope', *Math. Program.* **68**(3), 241–265.

Carrabs, F., Cordeau, J. F. e Laporte, G. (2007), 'Variable Neighborhood Search for the Pickup and Delivery Traveling Salesman Problem with LIFO Loading', *INFORMS J. on Computing* **19**(4), 618–632.

Chvátal, V., Cook, W., Dantzig, G., Fulkerson, D. e Johnson, S. (2010), Solution of a Large-Scale Traveling-Salesman Problem, *in* M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi e L. A. Wolsey, eds, '50 Years of Integer Programming 1958-2008', Springer Berlin Heidelberg, pp. 7–28.

Cordeau, J.-F., Iori, M., Laporte, G. e Salazar González, J. J. (2010), 'A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading', *Networks* **55**(1), 46–59.

Côté, J.-F., Archetti, C., Speranza, M. G., Gendreau, M. e Potvin, J.-Y. (2012), 'A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks', *Networks* **60**(4), 212–226.

Fügenschuh, A. e Martin, A. (2005), *Computational Integer Programming and Cutting Planes*, Vol. 12, Elsevier, pp. 69–121.

Goldberg, A. V. e Tarjan, R. E. (1986), A new approach to the maximum flow problem, *in* 'Proceedings of the eighteenth annual ACM symposium on Theory of computing', STOC '86, ACM, New York, NY, USA, pp. 136–146.

Gouveia, L. e Pesneau, P. (2006), 'On extended formulations for the precedence constrained asymmetric traveling salesman problem', *Netw.* **48**(2), 77–89.

Iori, M. e Martello, S. (2010), 'Routing problems with loading constraints', *TOP* **18**(1), 4–27.

Ladany, S. P. e Mehrez, A. (1984), 'Optimal routing of a single vehicle with loading and unloading constraints', *Transportation Planning and Technology* **8**(4), 301–306.

Levitin, G. e Abezgaouz, R. (2003), 'Optimal routing of multiple-load AGV subject to LIFO loading constraints', *Computers & Operations Research* **30**(3), 397–410.

Reinelt, G. (1991), 'Tsplib - a traveling salesman problem library', *INFORMS Journal on Computing* **3**(4), 376–384.

Instância			Estratégia 1		Estratégia 2		Estratégia 3		Estratégia 4		Estratégia 5	
nome	n	UB	gap	t(s)	gap	t(s)	gap	t(s)	gap	t(s)	gap	t(s)
a280	9	402	10.80	0.16	10.52	0.3	9.11	1.92	9.11	1.99	10.80	1.45
att532	9	4250	3.11	0.20	3.11	0.18	2.96	1.38	3.11	0.91	3.11	2.02
brd14051	9	4555	2.34	0.77	2.21	0.68	2.43	1.52	2.21	1.07	2.43	4.47
d15112	9	76203	4.45	0.36	3.14	0.55	4.45	1.55	3.14	1.03	3.14	3.42
d18512	9	4446	1.77	1.03	1.77	0.25	1.77	2.19	1.77	0.98	1.77	2.83
fnl4461	9	1866	0.00	1.02	0.00	0.18	0.00	3.79	0.00	1.08	0.00	5.14
nrw1379	9	2691	2.83	0.41	0.71	0.39	0.71	2.62	0.71	1.06	0.71	5.16
pr1002	9	12947	0.00	0.50	0.00	0.24	0.00	1.49	0.00	0.91	0.00	2.42
ts225	9	21000	9.52	3.5 0	9.52	1.06	9.52	3.54	9.52	1.13	9.52	4.10
a280	13	505	9.65	1.14	9.65	1.25	9.37	15.01	9.65	6.36		
att532	13	5800	3.84	1.15	3.84	1.09	3.84	7.76	3.84	5.29	3.84	14.80
brd14051	13	4936	4.92	2.98	4.70	2.54	4.90	6.39	4.70	5.45	4.70	21.72
d15112	13	93158	8.25	1.37	7.94	2.57	8.25	7.65	7.94	5.37	8.46	13.62
d18512	13	4704	5.51	4.12	5.49	2.08	5.50	16.31	5.49	5.95	5.51	22.85
fnl4461	13	2483	10.89	6.36	10.52	1.43	10.52	12.51	10.52	6.10	10.54	24.93
nrw1379	13	3366	10.35	1.64	9.41	1.58	9.85	6.01	9.43	5.63		
pr1002	13	15566	0.00	3.87	0.00	1.36	0.00	18.61	0.00	5.78	0.00	22.65
ts225	13	32395	6.43	17.54	6.43	4.32	6.43	36.29	6.43	7.16	6.43	44.02
a280	17	647	0.91	5.65	0.91	5.02	0.91	82.1	0.91	24.50		
att532	17	6361	5.71	2.95	5.71	3.69	5.71	45.88	5.71	20.99	5.71	63.54
brd14051	17	5196	4.93	12.67	4.88	7.75	4.94	33.91	4.91	20.53		
d15112	17	115554	10.02	4.32	10.02	7.54	10.02	18.83	10.02	18.88	10.18	27.83
d18512	17	5186	8.70	4.88	8.65	2.59	8.65	80.00	8.65	23.55		
fnl4461	17	2852	9.72	11.58	9.71	18.93	9.73	53.09	9.71	22.59	9.71	97.83
nrw1379	17	3652	10.93	10.11	10.48	6.71	10.81	33.58	10.48	24.88		
pr1002	17	17564	0.47	24.62	0.47	5.62	0.47	84.51	0.47	24.58	0.47	147.59
ts225	17	36703	5.78	35.96	5.79	14.96	5.89	134.46	5.79	29.29		
a280	21	752	8.87	19.11	8.84	16.7	8.57	240.37	8.57	80.27		
att532	21	10714	5.46	8.27	5.46	8.93	5.48	95.86	5.48	60.92	5.48	164.73
brd14051	21	5719	9.90	36.51	9.90	20.07	9.93	84.16	9.90	64.77	10.27	57.79
d15112	21	128798	12.41	14.99	12.41	10.48	12.42	96.61	12.41	35.63		
d18512	21	5634	9.64	133.81	9.62	55.62	9.62	251.43	79.61	70.38		
fnl4461	21	3269	14.20	32.39	14.02	58.43	14.33	125.91	14.03	69.00		
nrw1379	21	4282	13.21	10.22	13.03	12.1	13.05	42.75	13.02	61.41		
pr1002	21	20173	2.06	73.53	2.10	23.55			2.02	81.11		
ts225	21	43082	11.13	56.57	11.05	15.64			11.05	80.14		
a280	25	845	8.81	41.79	8.74	39.55			8.51	202.50		
att532	25	11484	5.70	26.82	5.73	18.01	5.73	313.02	5.73	162.42		
brd14051	25	7539	27.17	75.17	27.17	40.79	27.23	279.89	27.17	160.84	27.33	354.32
d15112	25	143654	13.85	182.47	13.84	62.29			13.84	173.63		
d18512	25	7291			26.33	150.92			26.34	181.39		
fnl4461	25	3860	18.87	266.4	18.84	153.28			18.84	179.46		
nrw1379	25	4836	13.38	39.88	13.26	32.12	13.27	322.48	13.25	173.29		
pr1002	25	22774	2.74	175.69	2.75	40.82			2.75	214.34		
ts225	25	54629	13.77	80.14	13.66	33.46			13.64	196.10		
Média			7.80	32.61	8.05	19.72	7.03	62.30	8.01	56.01	6.10	48.22

Tabela 1: Tempo de execução dos algoritmos de planos de cortes e limites inferiores obtidos

Instância		Estratégia 1				Estratégia 2				Estratégia 3				Estratégia 4				Estratégia 5			
nome	n	iter	sec	prec	lifo	iter	sec	prec	lifo	iter	sec	prec	lifo	iter	sec	prec	lifo	iter	sec	prec	lifo
a280	9	4	6	3	4	3	8	0	3	6	10	5	6	4	14	0	4	5	4	0	1
att532	9	3	4	2	1	2	4	0	2	3	4	3	1	2	6	0	2	5	3	0	2
brd14051	9	5	8	5	2	2	8	9	0	4	6	4	4	2	8	6	2	11	6	3	2
d15112	9	4	6	4	1	2	8	6	0	4	6	4	4	2	8	4	2	9	4	3	2
d18512	9	4	6	4	0	2	6	1	0	5	8	5	4	2	8	1	2	7	4	1	2
fnl4461	9	8	14	7	5	2	14	0	0	10	18	9	8	2	16	0	0	14	12	0	0
nrw1379	9	4	6	4	3	2	8	8	2					2	8	5	2	13	6	5	2
pr1002	9	4	6	3	0	2	6	0	0	4	6	3	2	2	8	0	0	6	4	0	0
ts225	9	11	20	9	0	2	20	0	0	8	14	7	2	2	16	0	0	10	9	0	0
a280	13	9	16	8	9	2	18	0	2	10	18	8	10	2	22	0	2				
att532	13	5	8	4	4	2	8	1	2	5	8	4	5	2	10	1	2	9	6	1	2
brd14051	13	4	6	4	1	2	6	6	0	3	4	3	3	2	6	6	0	10	2	6	0
d15112	13	5	8	5	4	2	8	6	2	4	6	4	4	2	6	6	2	7	4	1	2
d18512	13	6	10	6	4	2	12	4	0	8	14	8	7	2	14	4	2	11	7	2	2
fnl4461	13	8	14	8	3	2	16	4	0	6	10	6	6	2	16	4	2	12	7	3	2
nrw1379	13	4	6	4	1	2	6	8	0	9	16	7	6	2	6	7	2				
pr1002	13	7	12	5	0	2	14	0	0					2	18	0	0	11	9	0	0
ts225	13	13	23	12	0	2	27	0	0	17	32	16	6	2	32	0	0	20 17		1	0
a280	17	14	26	12	14	2	28	0	2	12	22	11	12	2	26	0	2				
att532	17	6	10	5	2	2	12	1	0	6	10	5	6	2	12	1	2	8	5	1	2
brd14051	17	5	8	5	0	2	8	5	0	5	8	5	5	2	8	3	2				
d15112	17	4	6	4	2	2	6	1	2	3	4	3	3	2	4	1	2	4	2	0	2
d18512	17	9	16	9	2	2	18	7	0	9	16	8	4	2	16	6	2				
fnl4461	17	9	15	9	2	2	17	3	0	7	12	7	4	2	16	3	0	12	8	2	0
nrw1379	17	8	13	8	2	2	14	12	2					2	16	9	2				
pr1002	17	11	20	10	0	2	24	2	0	12	22	11	8	2	26	2	0	19	16	1	0
ts225	17	21	39	20	3	2	36	11	0	16	30	16	10	2	34	10	0				
a280	21	18	34	17	18	2	38	0	2	16	30	15	16	2	38	0	2				
att532	21	7	12	6	2	2	14	2	0	5	8	4	5	2	8	1	2	8	5	1	2
brd14051	21	7	11	7	0	2	11	2	0	5	7	4	5	2	11	2	2	6	3	1	2
d15112	21	5	8	5	4	2	8	2	2	5	8	5	5	2	10	2	2				
d18512	21	7	12	7	3	2	12	5	0	9	16	9	2	2	18	5	0				
fnl4461	21	11	19	11	2	2	15	8	0	6	10	6	4	2	14	6	0				
nrw1379	21	3	4	3	0	2	5	4	0					2	4	4	2				
pr1002	21	17	32	17	0	3	34	7	0					2	28	12	2				
ts225	21	16	30	16	2	2	30	5	0					2	30	6	0				
a280	25	18	34	15	18	2	44	0	2					2	40	0	2				
att532	25	8	14	7	0	2	10	0	0	6	10	5	6	2	10	0	2				
brd14051	25	5	8	5	2	2	6	4	0	4	6	4	4	2	6	4	2	5	3	0	2
d15112	25	7	12	7	4	2	12	3	0					2	15	3	2				
d18512	25					2	14	7	0					2	18	6	0				
fnl4461	25	11	19	11	2	2	19	3	0					2	20	3	0				
nrw1379	25	5	8	5	1	2	10	7	0	7	12	7	7	2	12	6	2				
pr1002	25	19	36	19	0	2	30	14	0					2	35	10	0				
ts225	25	13	23	12	0	2	24	10	0					2	28	7	0				

Tabela 2: Número de desigualdades violadas para cada classe e estratégia de separação.