TÚLIO NEME DE AZEVEDO

Orientador: Marco Antonio Moreira Carvalho

UMA AVALIAÇÃO PRECISA DA MODELAGEM DO PROBLEMA DE MINIMIZAÇÃO DE TROCA DE FERRAMENTAS COMO O PROBLEMA DO CAIXEIRO VIAJANTE

Ouro Preto

Março de 2018

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

UMA AVALIAÇÃO PRECISA DA MODELAGEM DO PROBLEMA DE MINIMIZAÇÃO DE TROCA DE FERRAMENTAS COMO O PROBLEMA DO CAIXEIRO VIAJANTE

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

TÚLIO NEME DE AZEVEDO

Ouro Preto
Março de 2018

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Uma Avaliação Precisa da Modelagem do Problema de Minimização de Troca de Ferramentas como o Problema do Caixeiro Viajante

TÚLIO NEME DE AZEVEDO

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Marco Antonio Moreira Carvalho – Orientador
Universidade Federal de Ouro Preto

Dr. Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto

Dr. Puca Huachi Vaz Penna
Universidade Federal de Ouro Preto

Ouro Preto, Março de 2018

Resumo

O Problema de Minimização de Troca de Ferramentas (MTSP) é um problema combinatório encontrado em linhas de produção industriais que se caracteriza por processar um conjunto de diferentes tarefas em uma única máquina flexível. O processamento de cada tarefa requer que um conjunto específico de ferramentas seja alocado na máquina flexível, que possui um compartimento com capacidade fixa para comportá-las. Como as tarefas exigem conjuntos de ferramentas diferentes, é necessário que as ferramentas alocadas na máquina sejam trocadas entre o processamento de duas tarefas diferentes, respeitando a capacidade máxima da máquina e de forma que todas as tarefas sejam concluídas. A cada troca de ferramentas realizada, a máquina deve ser desligada, interrompendo a linha de produção, o que não é desejável. Assim, é necessário sequenciar as tarefas de forma a minimizar o número de trocas de ferramentas necessárias, minimizando o tempo ocioso da máquina flexível. O MTSP foi definido como $\mathcal{NP}$-Difícil, e vários autores tentaram solucionar o problema utilizando diversas modelagens. A modelagem mais utilizada é a que envolve o clássico Problema do Caixeiro Viajante (PCV). Neste trabalho, é revisitada a modelagem do MTSP via PCV, considerando cinco definições de custo diferentes, encontradas na literatura. Para realizar uma análise precisa desta modelagem, o problema resultante foi resolvido de maneira exata utilizando-se o resolvedor Concorde. Experimentos computacionais foram realizados e os resultados comparados com as melhores soluções presentes na literatura. Os tempos de execução reportados se mostraram muito competitivos, entretanto, em termos de qualidade das soluções, todos os métodos foram inferiores ao estado da arte.

Abstract

The Minimization of Tool Switch Problem (MTSP) is a combinatorial problem found in industrial production lines that is characterized by processing a set of different tasks in a single flexible machine. The processing of each task requires a specific set of tools to be allocated to the flexible machine, which has a magazine of fixed capacity to handle them. As the tasks require different tool sets, it is necessary to switch the tools allocated in the machine between the processing of two different tasks, respecting the maximum capacity of the machine and such that all tasks are fully processed. At each tool switch, the machine must be switched off, interrupting the production line, which is not desirable. Thus, it is necessary to determine the sequence of the tasks in order to minimize the number of tool switches required, minimizing the idle time of the flexible machine. The MTSP was defined as $\mathcal{NP}$-Hard, and several authors tried to solve it using several modelings. The most widely used modeling is the one that involves the classic Traveling Salesman Problem (TSP). In this work, the modeling of MTSP via TSP is revisited, considering five different cost definitions found in the literature. To perform an accurate analysis of this modeling, the resulting problem was solved exactly using the Concorde solver. Computational experiments were carried out and the results were compared with the best solutions reported in the literature. The running times reported were very competitive, however, in terms of solution quality, all methods were inferior to the state-of-the-art method.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a evolução frequente da tecnologia, as indústrias que utilizam máquinas flexíveis em suas linhas de produção recorrem a métodos computacionais para resolverem problemas operacionais. Por sua vez, esses métodos, com suas respectivas especificidades, podem contribuir significativamente para otimização da eficiência das linhas de produção.

Uma *máquina flexível* tem como característica proporcionar dinamismo à linha de produção ao fabricar diversos tipos de produtos não relacionados entre si. Este tipo de máquina comporta um número fixo de *ferramentas* instaladas, por exemplo, chaves de fenda, alicates e brocas, entre outras. Diferentes conjunto de ferramentas produzem diferentes tipos de produtos. Cada produto exige que um determinado conjunto de ferramentas esteja instalado no momento de sua produção.

A capacidade da máquina de produção é suficiente para suportar o conjunto de ferramentas necessárias para a fabricação de cada produto isoladamente, porém, não comporta todas as ferramentas necessárias para fabricar todos os produtos simultaneamente. Durante a fabricação dos produtos, trocas de ferramentas serão necessárias para que a capacidade não seja ultrapassada e o produto subsequente seja processado. Desta maneira, cada troca exige que a máquina seja desligada, interrompendo a linha de produção, ocasionando aumento de custo de produção e ociosidade.

Um dos problemas presentes nas indústrias que empregam máquinas flexíveis, é o *Problema de Minimização de Troca de Ferramentas* (MTSP). Este problema consiste em determinar a sequência de fabricação dos produtos a cada estágio nas linhas de produção. A fabricação de um produto é considerada uma *tarefa* a ser realizada por uma máquina flexível. O objetivo do MSTP é otimizar a linha de produção pela minimização das trocas de ferramentas necessárias. Desta forma, a produção é interrompida minimamente, tornando-a mais eficiente.

O MTSP se divide em duas partes: o problema de carregamento de ferramentas e o problema de sequenciamento de tarefas, como definido por Tang e Denardo (1988). O problema de carregamento de ferramentas, dada uma sequência fixa de tarefas, consiste em determinar o menor número de trocas de ferramentas para processar o conjunto de tarefas. O método *Keep*

Tool Neeeded Soonest (KTNS), proposto por Tang e Denardo (1988), soluciona este problema em tempo determinístico polinomial. Quando troca de ferramentas são necessárias, a política KTNS garante que as ferramentas que serão necessárias mais brevemente pelas tarefas ainda não processadas sejam mantidas na máquina. O problema de sequenciamento de tarefas é $\mathcal{NP}$-Difícil (Crama et al., 1994), significando que não se conhece algoritmo eficiente para sua solução. Então, é faz necessário resolver o problema de sequenciamento das tarefas.

O MTSP se apresenta de diferentes formas na literatura. Este trabalho aborda o caso geral, no qual o tempo de instalação é o mesmo para todas ferramentas e o tamanho e custo das ferramentas são uniformes, ou seja, não importa qual a posição as ferramentas são alocadas na máquina flexível.

Abordando o caso geral, este trabalho propõe a análise crítica de uma modelagem amplamente adotada na literatura. Em trabalhos anteriores, após a modelagem utiliza-se métodos heurísticos para solução do problema, o que impede a análise precisa da modelagem adotada. Para conferir precisão à analise, a modelagem é realizada conforme reportado em diferentes trabalhos da literatura e resolvida pela primeira vez de maneira exata. A introdução de um resolvedor exato permite que conclusões inequívocas sobre a qualidade da modelagem sejam realizadas.

1.1 Motivação

O problema considerado neste trabalho possui ampla aplicação prática em sistemas de manufatura flexível. Além da aplicabilidade prática, trata-se de um problema $\mathcal{NP}$-Difícil, o que gera interesse teórico.

Especificamente, a modelagem considerada neste trabalho é utilizada em diversos trabalhos da literatura, porém, sem a realização de uma análise aprofundada de sua qualidade. Desta maneira, não é possível determinar se os resultados reportados na literatura são devidos à modelagem utilizada ou à qualidade dos métodos empregados para solução da mesma.

1.2 Objetivos

O trabalho de pesquisa abordado neste projeto se propõe a estudar o Problema de Minimização de Trocas de Ferramentas em sistemas de manufatura flexíveis, revisitando sua modelagem pelo Problema do Caixeiro Viajante e utilizando o resolvedor exato Concorde para sua solução. Desta forma, é possível avaliar precisamente a qualidade da modelagem empregada amplamente na literatura sobre o MTSP. São os objetivos específicos:

1. Realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre o tema tratado;

2. Realizar pesquisa sobre a modelagem do Problema de Minimização de Trocas de Ferramentas como o Problema do Caixeiro Viajante;

3. Realizar pesquisa para geração de embasamento teórico a respeito do resolvedor Concorde;

4. Implementar a modelagem e solução exata do Problema de Minimização de Trocas de Ferramentas modelado como o Problema do Caixeiro Viajante; e

5. Avaliar o método implementado considerando as definições disponíveis na literatura e problemas teste publicamente disponíveis, realizando uma análise crítica considerando outros métodos da literatura.

1.3 Organização

O restante deste trabalho está organizado como descrito a seguir. O Capítulo 2 relata os trabalhos apresentados na literatura. Os fundamentos teóricos do MTSP e do PCV, juntamente com a descrição do resolvedor Concorde são apresentados no Capítulo 3. O Capítulo 4 detalha as definições de custo propostas na literatura para a modelagem do MTSP baseada no PCV. Os experimentos computacionais envolvendo a modelagem analisada e o método que represente o atual estado da arte são reportados no Capítulo 5. Por fim, as conclusões deste trabalho são apresentadas no Capítulo 6.

Capítulo 2

Revisão da Literatura

Tang e Denardo (1988) foram os primeiros a relatarem na literatura uma abordagem ao MTSP, sugerindo a modelagem do problema como o Problema do Caminho Hamiltoniano Mínimo em um grafo completo em que os vértices representam as tarefas e cujos pesos das arestas foi definido como um limitante inferior para o número de troca de ferramentas entre duas tarefas específicas. Contudo, a principal contribuição deste trabalho foi a política ótima *Keep Tool Needed Soonest* (KTNS) para troca de ferramentas. Dada uma sequência fixa de tarefas a serem realizadas, esta política define o menor número de troca de ferramentas correspondente. Se trocas forem necessárias, a política garante que as ferramentas que serão necessárias mais brevemente pelas tarefas ainda não realizadas sejam mantidas na máquina flexível.

No mesmo ano, Bard (1988) formulou o MTSP como um problema de programação inteira não linear e o resolveu com uma heurística baseada na relaxação dual do modelo. O conjunto de instâncias utilizados nos experimentos não teve sua origem relatada. Todavia, os resultados demonstraram a necessidade de um tempo de execução relativamente pequeno para determinar boas soluções locais.

Crama et al. (1994) relacionaram o MTSP com o Problema do Caixeiro Viajante (PCV), o que inovou a modelagem do MTSP. Neste trabalho foram aplicadas heurísticas específicas para o PCV na resolução do MTSP. Também foi definida a complexidade computacional do problema, $\mathcal{NP}$-Difícil para capacidade da máquina maior ou igual a 2.

Posteriormente, Hertz et al. (1998) utilizaram a mesma modelagem, embutindo a política KTNS em heurísticas existentes para solução do PCV. Estas heurísticas incluem GENI e GENIUS (Gendreau et al., 1992) e *Farthest Insertion* (Golden e Stewart, 1985). As heurísticas GENI e *Farthest Insertion* consistem em métodos construtivos, ou seja, geram soluções gradativamente, baseando-se em métodos de inserção. A heurística GENIUS utiliza o mesmo princípio de GENI, porém, com uma fase de pós-otimização, denominadas US. Neste mesmo trabalho foram apresentadas definições formais para o cálculo das distâncias entre as tarefas, utilizadas como pesos das arestas nos métodos de Crama et al. (1994) e foi definida uma fun-

ção objetivo para o PCV que considera o KTNS como um de seus componentes. Dentre os métodos citados, a versão do GENIUS que incorpora o KTNS obteve o melhor desempenho.

O primeiro trabalho a utilizar dados reais oriundos das indústrias é devido a Shirazi e Frizelle (2001), provando que os métodos existentes na literatura eram melhores que os utilizados nas indústrias. Observou-se que as instâncias reais possuíam máquinas com capacidade elevada em relação as instâncias da literatura, de forma que as instâncias reais possuíam solução mais fácil do que as da literatura.

Diferentes problemas de sequenciamento de produção foram relacionados por Linhares e Yanasse (2002), o Problema de Pilhas Abertas (*Minimization of Open Stacks Problem*, MOSP), o Problema de Minimização de Descontinuidades (*Minimization of Discontinuities Problem*, MDP) e Problema de Minimização de Espalhamento de Ordens (*Minimization of Order Spread Problem*, MORP). Entre os resultados teóricos deste trabalho, destaca-se a prova de que MTSP e MOSP, apesar de relacionados, não são equivalentes entre si.

Al-Fawzan e Al-Sultan (2003) propuseram implementações de Busca Tabu para solução do MTSP, utilizando diferentes estratégias e métodos probabilísticos para reduzir o espaço de busca. Adicionalmente, foram utilizados dois métodos para geração de soluções aleatórias, *Random Swapping*, que realiza trocas entre pares de tarefas, e *Random Block Insertion*, que realiza trocas entre pares de blocos de tarefas consecutivas. Comparados com a Busca Tabu original, a versão mista, ou seja, a que utilizou estratégias probabilísticas e de oscilação, apresentou melhores resultados.

Os métodos *branch-and-cut* e *branch-and-bound* foram aplicados à solução do MTSP por Laporte et al. (2004). Neste trabalho foi proposta uma nova função objetivo, no intuito de melhorar a relaxação linear de um modelo baseado no PCV. Essa nova proposta introduziu restrições oriundas do PCV à formulção do MTSP reportada por Tang e Denardo (1988), resultando assim na nova função objetivo. O *branch-and-bound* foi capaz de solucionar instâncias com até 25 tarefas e 25 ferramentas em tempo computacional aceitável. Um novo *branch-and-bound* foi a base para a implementação do *beam search* de Zhou et al. (2005), que reduziu o espaço de busca ao limitar o número de nós na árvore de busca. Os resultados apresentados foram melhores que os resultados reportados por Bard (1988).

Privault e Finke (2000) seguiram a linha de pesquisa que utiliza métodos específicos para resolução do PCV para solucionar o MTSP. Foi proposta uma adaptação de um método para o problema de localização de k-servidores móveis para grandes lotes de requisições, gerando o método *Bulk-Service Partitioning Algorithm* (BSPA). Desta forma, os servidores foram representados pelos *slots* de ferramentas da máquina e os pedidos em grandes lotes pelos conjuntos de ferramentas necessárias a uma tarefa. Os resultados obtidos foram comparados aos de métodos específicos para o PCV, sendo a heurística *Farthest Insertion* utilizada para instâncias esparsas, e os clássicos *2-opt* e *Shortest Edge* aliados a um limite inferior utilizados para instâncias densas. O método BSPA apresentou boas soluções, porém o tempo computacional

foi muito maior que os comparados.

Djellab et al. (2000) propuseram uma modelagem do MTSP através de hipergrafos, associada ao método *Iterative Best Insertion*, além de dois métodos baseados no KTNS para definição do menor número de troca de ferramentas dada a sequência de tarefas. Em seus experimentos, foram consideradas apenas instâncias geradas pelos próprios autores, sem comparação com nenhum outro método da literatura.

Novamente modelando o MTSP com base em outros problemas combinatórios, Yanasse e Pinto (2002) propuseram uma modelagem, porém, associada ao Problema de Fluxo de Custo Mínimo em Redes. Não foram realizados experimentos computacionais, porém, foram realizadas observações sobre algumas instâncias presentes na literatura e relatou-se que a proposta demonstra ser promissora. Posteriormente, Yanasse e Lamosa (2005) enunciaram uma proposta de modelagem utilizando o Problema do Caixeiro Viajante Generalizado, implementada por Yanasse e Lamosa (2006), porém, experimentos computacionais não foram realizados, sendo este apenas um artigo teórico.

Uma extensão do MTSP que considera ferramentas de tamanho não uniforme, foi abordada poe Tzur e Altman (2004). Neste trabalho, foi proposto o método *Aladdin*, baseado no melhor método existente até então para resolver o MTSP original, o GENIUS. Este mesmo método fora utilizado anteriormente por Hertz et al. (1998). Neste mesmo trabalho foi proposta a política *Keep Smaller Tools Needed Soonest*, para determinação do número de troca de ferramentas para a extensão do MTSP abordada.

Posteriormente, Crama et al. (2007) consideraram esta mesma extensão do MTSP. Concluiu-se que, quando a capacidade de ferramentas na máquina é fixa, o problema pode ser resolvido em tempo determinístico polinomial, utilizando um método proposto no mesmo trabalho. Apesar da complexidade polinomial, este método apresenta alto custo (polinômio de ordem alta) para resolução do problema.

Um algoritmo memético foi utilizado por Amaya et al. (2008), cuja implementação utilizou uma variação de algoritmo genético e *Hill Climbing*. Os resultados foram comparados com o *Beam Search* de Zhou et al. (2005) e considerados melhores.

Outro *branch-and-bound* foi proposto por Yanasse et al. (2009), juntamente com um método para determinar limitantes inferiores para o valor das soluções. Os resultados foram comparados com os resultados obtidos por Laporte et al. (2004). O *branch-and-bound* foi capaz de resolver instâncias não resolvidas anteriormente pelo método comparado. Entretanto, este método não foi capaz de resolver outras instâncias, como alguns casos com 25 tarefas.

Baseados neste último *branch-and-bound*, Senne e Yanasse (2009) empregaram o método *Beam Search*, que seleciona regiões promissoras do espaço de busca, com o objetivo de determinar limitantes superiores para acelerar os métodos já existentes. Os resultados obtidos foram comparados com os obtidos por Laporte et al. (2004). O método foi capaz de obter 92% de soluções entre ótimas e melhores soluções existentes.

A pesquisa em algoritmos meméticos teve sequência por Amaya et al. (2012). Neste trabalho, o algoritmo memético foi relacionado com técnicas utilizadas em problemas de otimização em redes. Os resultados foram comparados com os de Amaya et al. (2008), superando-os.

Uma nova modelagem em grafos, porém, não relacionada ao PCV, foi apresentada por Chaves et al. (2012). A solução do MTSP foi dividida em duas fases: construtiva e refinamento. Na fase construtiva gera-se uma solução viável e, posteriormente, a fase de refinamento é responsável por encontrar o máximo local para a solução inicial encontrada na primeira fase, através de uma Busca Local Iterada (*Iterated Local Search*,ILS). As soluções encontradas foram utilizadas como limitantes superiores para o algoritmo *Branch-and-Bound*, proposto anteriormente por Yanasse et al. (2009). O algoritmo apresentou uma diminuição de 74% dos nós gerados na árvore de busca e de 76% no tempo computacional.

Novamente, Amaya et al. (2013) propuseram diferentes formulações para algoritmos meméticos e genéticos. Desta vez, os algoritmos foram combinados com entropia cruzada e diferentes buscas locais. Os resultados foram comparados com os reportados em Amaya et al. (2012), superando-os. Foi também relatado que os algoritmos que utilizam a entropia cruzada são melhores que os que não utilizam, não havendo um método dominante.

Outra extensão do MTSP foi estudado por Raduly-Baka e Nevalainen (2015), considerando posições fixas para as ferramentas na máquina. Desta forma, uma ferramenta não pode ser instalada em qualquer posição livre não máquina flexível. Foi provado que esse problema é mais complexo que o MTSP original, sendo $\mathcal{NP}$-difícil no sentido forte (da definição $\mathcal{NP}$-hard). No caso em que o número de módulos alimentadores é fixo, pode-se encontrar a solução em tempo determinístico polinomial, porém, a ordem do polinômio é alta.

Dois modelos de programação linear inteira específicos para solucionar o MTSP foram propostos por Catanzaro et al. (2015). Os resultados reportados indicam que ambos os métodos apresentam melhor relaxação linear do que os métodos anteriores da literatura, que têm a mesma base conceitual. Uma política alternativa ao KTNS foi recentemente proposta por Adjiashvili et al. (2015). Embora o novo método exija tempo determinístico polinomial para determinar um plano de troca de ferramentas, assim como o KTNS, não é realizada comparação entre os dois.

Chaves et al. (2016) apresentaram um método baseado em *Clustering Search* e Algoritmo Genético de Chaves Aleatórias Viciadas (*Biased Random Key Genetic Algorithm* – BRKGA). A metaheurística *Clustering Search* identifica regiões promissoras dentro do espaço de busca e realiza a busca local pelo método Descida em Vizinhança Variável (*Variable Neighborhood Descent* – VND) nessas regiões. A metaheurística BRKGA foi utilizada para gerar soluções dentro da região do espaço de busca determinada pelo *Clustering Search*. Os resultados obtidos por Chaves et al. (2012) nas instâncias propostas por Yanasse et al. (2009) e Crama et al. (1994) foram comparados nos experimentos computacionais conduzidos. O novo método foi capaz de igualar ou superar todos os resultados anteriores para estas instâncias.

Mais recentemente, uma nova abordagem realizada sobre a estratégia de busca local iterada foi proposta por Paiva e Carvalho (2017). Neste trabalho foram propostos uma nova representação em grafos para o problema, uma nova heurística construtiva para a geração de soluções iniciais e um novo método de busca local. Os resultados do ILS foram comparados aos resultados obtidos por Chaves et al. (2016) nas instâncias de Yanasse et al. (2009) e Crama et al. (1994), além dos melhores resultados obtidos para as instâncias de Catanzaro et al. (2015). O novo método foi capaz de igualar ou superar todo os resultados anteriores para essas instâncias, sendo considerado o atual estado da arte para solução do MTSP.

Capítulo 3

Fundamentação Teórica

Neste capítulo são descritos formalmente o Problema de Minimização de Trocas de Ferramentas e o clássico Problema do Caixeiro Viajante. Também é apresentado brevemente o método exato que representa o estado da arte para solução do Problema do Caixeiro Viajante.

3.1 O Problema de Minimização de Trocas de Ferramentas

Formalmente, o MTSP é definido como descrito a seguir. Dados uma máquina flexível com capacidade de comportar até C ferramentas, um conjunto de tarefas $T = \{1, \ldots, n\}$, um conjunto de ferramentas $F = \{1, \ldots, m\}$, o subconjunto de ferramentas F_i ($F_i \in F$) necessárias para processar a tarefa i ($i \in T$), é necessário determinar uma permutação ϕ dos elementos de T tal que o número de trocas de ferramentas, obtido pela aplicação do algoritmo KTNS, seja minimizado.

Uma instância do MTSP apresenta informações sobre as ferramentas necessárias para o processamento de cada uma das tarefas. A Tabela 3.1 apresenta os dados de um cenário relacionado ao MTSP em que tem-se $n = 5$, $m = 5$, $C = 3$.

Tabela 3.1: Exemplo de instância MTSP.

Tarefas	Ferramentas
1	2, 3, 5
2	1, 3
3	1, 4, 5
4	1, 2
5	2, 3, 4

Na referida Tabela 3.1, a primeira coluna representa as tarefas a serem processadas pela máquina flexível, enumeradas de 1 a 5. A segunda coluna representa as ferramentas necessárias para que a tarefa referida seja processada. Computacionalmente, esta instância pode ser modelada por uma matriz binária Q, conforme apresentado pela Tabela 3.2. As linhas de Q

representam as ferramentas e as colunas representam as tarefas, ambas numeradas de 1 a 5. Os elementos q_{ij} da matriz Q são definidos $q_{ij} = 1$ caso a ferramenta i ($i \in F$) seja necessária para processar a tarefa j ($j \in T$). Caso contrário, $q_{ij} = 0$.

Tabela 3.2: Matriz Q.

Ferramentas\Tarefas	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	1	1
3	1	1	0	0	1
4	0	0	1	0	1
5	1	0	1	0	0

Uma solução para este problema é dada por um sequenciamento ϕ das colunas de Q. Por exemplo, considere $\phi = [3, 4, 2, 1, 5]$, representada pela Tabela 3.3. Esta solução resulta em 7 troca de ferramentas, explicitadas a seguir:

Tabela 3.3: Solução MTSP.

Tarefas	Ferramentas carregadas na máquina
3	1, 4, 5
4	1, 2, 5
2	1, 2, 3
1	2, 3, 5
5	2, 3, 4

1. Três trocas para carregar as ferramentas iniciais na máquina (ferramentas 1,4,5);
2. Uma troca entre a tarefa 3 e 4 (ferramenta 4 por 2);
3. Uma troca entre a tarefa 4 e 2 (ferramenta 5 por 3);
4. Uma troca entre a tarefa 2 e 1 (ferramenta 1 por 5); e
5. Uma troca entre a tarefa 1 e 5 (ferramenta 5 por 4).

Considerando a representação por matrizes binárias, a solução ϕ induz uma matriz permutação R^{ϕ}, vide Tabela 3.4, cujas colunas são as colunas de Q na ordem estabelecida por ϕ e uma coluna adicional 0 com todos os elementos nulos. Na referida Tabela 3.4, as ferramentas inseridas a cada instante na máquina estão sublinhadas e as ferramentas que permanecem na máquina, mesmo sem serem utilizadas, estão indicadas em negrito. Na troca da tarefa 3 para a tarefa 4, a ferramenta 2 foi inserida na máquina, portanto sublinhada. A ferramenta 5 foi mantida na máquina mesmo sem ser utilizada na tarefa 4, portanto, em negrito.

Tabela 3.4: Matriz R^{ϕ}.

Ferramentas\ϕ	0	3	4	2	1	5
1	0	$\underline{1}$	1	1	0	0
2	0	0	$\underline{1}$	**1**	1	1
3	0	0	0	$\underline{1}$	1	1
4	0	$\underline{1}$	0	0	0	$\underline{1}$
5	0	$\underline{1}$	**1**	0	$\underline{1}$	0

O valor dos elementos r_{ij}^{ϕ} é definido de acordo com a Equação 3.1.

$$r_{ij}^{\phi} = \begin{cases} 1, \text{ se a ferramenta } i \in F \text{ estiver na máquina durante o processamento da tarefa } j \in T \\ 0, \text{caso contrário.} \end{cases}$$

Uma solução representada pela matriz R^{ϕ} é avaliada de acordo com a Equação 3.1, proposta por Crama et al. (1994). Esta função calcula, dada uma sequência de tarefas, o número de inversões de 0 para 1, que representam a inserção de ferramentas na máquina.

$$Z_{MTSP}^{\phi}(R) = \sum_{j \in T} \sum_{i \in F} r_{ij}^{\phi}(1 - r_{ij-1}^{\phi}) \tag{3.1}$$

O objetivo do MTSP é determinar a permutação $\phi \in \Phi$ das colunas da matriz Q que resulte no menor número de trocas de ferramentas, em que Φ é o conjunto de todas as permutações possíveis. A função objetivo correspondente é exibida na Equação 3.2.

$$\min_{\phi \in \Phi} Z_{MTSP}^{\phi}(Q) \tag{3.2}$$

Conforme mencionado anteriormente, o Problema de Minimização de Trocas de Ferramentas pertence à classe $\mathcal{NP}$-Difícil para os casos em que $C \geq 2$ (Crama et al., 1994).

3.2 O Problema do Caixeiro Viajante

Em seu livro, Cook (2012), uma referência para o estudo do PCV, relata que a origem do Problema do Caixeiro Viajante não possui datamento preciso, e faz uma análise das origens históricas do mesmo. Um relato indica que em 1925 uma empresa dos EUA, a *Page Seed Company*, deixou para seu funcionário, *Mr. Cleveland*, uma lista de cidades para visitar em busca de negócios. *Mr. Cleveland* e a empresa fizeram observações que deixaram claro o objetivo de minimizar o tempo gasto na rota. Outro relato cita que em 1832, um livro alemão fez a descrição do PCV, em que é indicado que a idéia principal do problema é sempre visitar o máximo de cidades possíveis sem ter que visitá-las duas vezes. Um terceiro relato indica que os primeiros estudos matemáticos relacionados ao PCV foram datados de 1930, em Harvard e

Viena, apresentando o problema para a comunidade matemática.

O PCV é um problema combinatório que em sua versão de otimização consiste em, dadas uma lista de n cidades e as distâncias entre elas, determinar uma rota que comece e termine em uma mesma cidade e passe pelas demais cidades exatamente uma vez, com o menor custo possível. A Figura 3.1 ilustra o problema, através de um grafo, no qual os vértices representam as cidades e as arestas representam as ligações entre duas cidades, cujas distâncias são representadas pelos pesos das arestas.

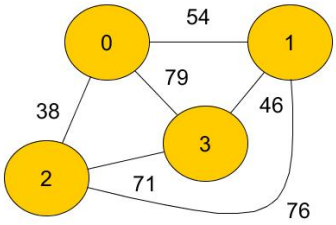

Figura 3.1: Representação de uma instância do PCV.

Computacionalmente, uma instância do PCV pode ser modelada por uma matriz W $n \times n$, exemplificada na Tabela 3.5. Na matriz, as linhas e colunas representam cada uma das n cidades do problema, e cada elemento w_{ij} da tabela, indica a distância entre as cidades i e j $(i, j = 1 \ldots n)$. Note que $w_{ij} = w_{ji}$ no caso simétrico e quando $i = j$, w_{ij} possui valor nulo.

Tabela 3.5: Matriz W.

	0	1	2	3
0	-	54	38	79
1	54	-	76	46
2	38	76	-	71
3	79	46	71	-

Dado o exemplo da Tabela 3.5, uma solução para o exemplo de instância referida é construída da seguinte maneira: seleciona uma cidade qualquer, e a partir dela seleciona as demais, de forma a construir um ciclo sem repetições de cidades. A primeira cidade escolhida é a que termina o ciclo. Por exemplo, uma solução $S = [0, 2, 1, 3, 0]$ indica que o ciclo começa e termina na cidade 0, visitando as cidades 2, 1 e 3, nesta ordem, durante o percurso. O custo total da solução S é 239, calculada pela soma das distâncias entre as cidades, de acordo com a Tabela 3.5, apresentadas a seguir:

1. Distância da cidade 0 para 2: 38;
2. Distância da cidade 2 para 1: 76 (acumulado, 114);
3. Distância da cidade 1 para 3: 46 (acumulado, 160); e

4. Distância da cidade 3 para 0: 79 (acumulado, 239).

O objetivo do PCV é determinar a rota que possua o menor custo, ou seja, a solução ótima é aquela que minimiza a distância total percorrida. Para a instância referida na Tabela 3.5, a solução ótima é dado pela sequência $S = [0, 1, 3, 2, 0]$, que possui custo total 209.

O PCV, em sua versão de otimização é de complexidade $\mathcal{NP}$-Difícil (Laporte, 1992), significando que não se conhece algoritmo eficiente para sua solução. Embora haja diversas heurísticas clássicas para sua solução, estas heurísticas não possuem bom desempenho à medida em que o número cidades aumenta, tendendo ao aumento rápido do tempo de execução. Da mesma forma, métodos exatos exigem muito tempo para a solução exata, tornando-se impraticáveis.

Novamente, Cook (2012) destaca a evolução dos algoritmos criados para solução do PCV. O primeiro, que permaneceu como estado da arte por 17 anos, resolveu uma instância do problema contendo 49 cidades. Atualmente, o estado da arte é o resolvedor *Concorde*[1], utilizado para resolver o PCV entre outros problemas, como o roteamento de veículos, o mapeamento genético e outros. Em se tratando do PCV, este resolvedor foi utilizado para resolver problemas existentes na *TSPLIB*[2], uma biblioteca específica de casos de testes para o PCV. Em seus resultados, se destaca a solução ótima de todos os 110 casos de teste disponíveis, incluindo um caso de teste com 85.900 cidades, assim sendo o maior número de cidades resolvidas atualmente.

3.3 O Resolvedor Concorde

O resolvedor *Concorde* é uma ferramenta desenvolvida para solução do PCV disponível também para vários problemas de otimização relacionados. A ferramenta pode ser utilizada gratuitamente para fins acadêmicos, havendo também um servidor que permite a resolução de instâncias do problema *online*.

Codificado em ANSI C, apesar de contar com métodos heurísticos em seu código com mais de 700 funções, o Concorde é um método exato. A exatidão do resolvedor é devida à utilização do método de plano de cortes, que por sua vez resolve os problemas utilizando técnicas de relaxação linear. Verifica-se também o uso de técnicas para o cálculo de limitantes inferiores: Triangulação de *Delaunay*, Árvore Geradora Mínima e várias heurísticas do tipo *vizinho mais próximo*. Aliadas a esses métodos, são disponibilizadas também cinco heurísticas geradoras de soluções alternativas para o PCV: gulosa (não especificada), *Boruvka* (Boruvka, 1926), *Quick Boruvka* (Applegate et al., 2003), *Nearest Neighbor*, *Chained Lin-Kernighan* (Kernighan e Lin, 1970) e solução aleatória.

[1] `http://www.math.uwaterloo.ca/tsp/concorde`
[2] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

O resolvedor também possui suporte para programação em ambientes paralelos de memória compartilhada e uma interface gráfica opcional. Estas funções permitem ao usuário a criar códigos especializados para problemas que se relacionam com o PCV, como por exemplo o MTSP e problemas de otimização em redes. Também há suporte para os resolvedores de programação linear *QSopt*[3] e *CPLEX*[4].

Apesar de o resolvedor apresentar resultados expressivos para o PCV, a sua documentação está incompleta e não há suporte oficial para sua utilização, o que dificulta o entendimento e a sua utilização. De fato, o pacote é disponibilizado sem garantias de precisão ou corretude de resultados. Uma contribuição deste trabalho é a criação e disponibilização de um relatório técnico que descreve a instalação do resolvedor Concorde utilizando os resolvedores *CPLEX* ou *QSopt* em ambientes Linux e também a modelagem para os problemas e utilização das rotinas básicas para solução dos mesmos. Todos os tópicos são acompanhados de exemplos minimamente funcionais. Este relatório técnico está incluso no Apêndice B.

Com efeito, durante os experimentos computacionais, erros de execução e solução foram verificados. Por exemplo, na documentação do resolvedor estão relacionadas duas funções indicadas como equivalentes entre si para solução dos problemas devidamente modelados (as funções *CCtsp_solve_sparse* e *CCtsp_solve_dat*). Porém, durante os experimentos computacionais, constatou-se o contrário. Uma das funções (*CCtsp_solve_sparse*) não foi capaz de resolver boa parte das instâncias consideradas, ao passo que a outra (*CCtsp_solve_dat*) resolveu todas.

[3] `http://www.math.uwaterloo.ca/~bico/qsopt`
[4] `http://www.ibm.com/software/commerce/optimization/cplex-optimizer/`

Capítulo 4

Relacionando o MTSP com o PCV

Uma maneira comum para solucionar problemas de otimização consiste em realizar modelagens de problemas correlatos entre si que possuam métodos de solução de boa qualidade. Desta forma, a solução do problema original é composta da solução gerada pela modelagem para um segundo problema.

Embora o MTSP e o PCV não sejam equivalentes entre si no caso geral, uma modelagem comum, conforme mencionado no Capítulo 2, relaciona ambos os problemas considerando um grafo completo em que cada tarefa é representada por um vértice. Os pesos das arestas podem ser definidos de diferentes maneiras, de forma a converter o número de troca de ferramentas entre duas tarefas processadas contiguamente, característica do MTSP, na distância entre cidades, característica do PCV. Neste capítulo, revisitamos as diferentes formas de cálculo de distâncias propostas na literatura ao modelar o MTSP como o PCV.

Além da dificuldade introduzida pela definição dos pesos das arestas, a análise da literatura realizada na Seção 2 indica que somente métodos heurísticos foram considerados para solução dos modelos. Desta forma, as soluções geradas possuem, além da degradação da modelagem por si só, a degradação oriunda da solução por heurísticas, o que dificulta uma avaliação precisa da modelagem.

Hertz et al. (1998) descrevem cinco definições diferentes para o cálculo das distâncias d_{ij} $(i, j \in T)$ dada uma instância do MTSP. As três primeiras são conhecidas anteriormente na literatura e outras duas são novas propostas. Nas equações a seguir, C é a capacidade máxima da máquina e F_i $(F_i \in F)$ é conjunto de ferramentas necessárias a processar a tarefa i $(i \in T)$.

4.1 Definição 1

A primeira definição, apresentada na Equação 4.1, relaciona a capacidade máxima da máquina e o número de ferramentas em comum entre duas tarefas. Deste modo, determina-se quantas ferramentas permanecerão na máquina e a capacidade restante da máquina é utilizada como um limitante superior para o número de trocas.

$$d_{ij} = C - \mid F_i \cap F_j \mid \tag{4.1}$$

Esta definição foi originalmente proposta por Crama et al. (1994). Hertz et al. (1998) constatam que o MTSP se reduz ao PCV caso todas as tarefas T_i ($T_i \in T$) necessitem de exatamente C ferramentas para serem processadas. Além disso, reportaram que o caso geral apresenta $\mid T_i \mid < C$ ($T_i \in T$) e ainda assim tem-se algoritmos para o PCV que resultam em bons resultados heurísticos para o problema.

4.2 Definição 2

A definição apresentada pela Equação 4.2, também foi proposta originalmente por Crama et al. (1994). Considera-se como limitante superior para o número de troca de ferramentas entre duas tarefas a diferença simétrica entre os conjuntos de ferramentas necessárias para processar duas tarefas diferentes (i.e., a união dos conjuntos subtraída da interseção entre os mesmos).

$$d_{ij} = \mid F_i \cup F_j \mid - \mid F_i \cap F_j \mid \tag{4.2}$$

Uma observação válida sobre as Equações 4.1 e 4.2 é que quando $\mid F_i \mid = C$ as equações se tornam equivalentes. Adicionalmente, ambas resultam em valores altos caso duas tarefas i e j compartilhem poucas ferramentas entre si.

4.3 Definição 3

A terceira definição se baseia na proposta original por Tang e Denardo (1988) para cálculo de limitantes inferiores para o número de trocas de ferramentas, apresentada na Equação 4.3.

$$d_{ij} = \mid F_i \cup F_j \mid - C \tag{4.3}$$

Esta definição considera o número de ferramentas utilizadas por duas tarefas que excede a capacidade de máquina. Entretanto, é possível que valores negativos sejam produzidos, quando duas tarefas i e j exigirem um número baixo de ferramentas quando comparado à capacidade da máquina.

Crama et al. (1994) propuseram uma adaptação da definição de custo de Tang e Denardo (1988) para evitar valores negativos. A adaptação consiste em definir a distância como zero caso a definição resulte em um valor negativo. A Equação 4.4 apresenta esta definição.

$$d_{ij} = \max\{0, \mid F_i \cup F_j \mid - C\} \tag{4.4}$$

Esta definição de distância é utilizada pela maioria dos autores que solucionaram o MTSP modelando-o como o PCV (Tang e Denardo, 1988; Crama et al., 1994; Hertz et al., 1998; Laporte et al., 2004; Privault e Finke, 2000; Catanzaro et al., 2015). Adicionalmente, Fathi e Barnette (2002) utilizaram esta definição como limitante inferior para o Problema de Sequenciamento em Máquinas Idênticas Paralelas com Restrições de Ferramentas.

As três primeiras definições (Equações 4.1, 4.2 e 4.4) levam em consideração somente duas tarefas subsequentes i e j, sem considerar os conjuntos de ferramentas carregadas na máquina antes do processamento da tarefa i e depois do processamento da tarefa j. Levando em consideração esta análise, Hertz et al. (1998) propôs duas novas definições, descritas nas seções a seguir.

4.4 Definição 4

A definição expressada pela Equação 4.5, aperfeiçoa a definição de custo anterior, adotando o critério de frequência de utilização para a permanência das ferramentas na máquina, carregadas anteriormente. Sendo assim, subtrai-se uma quantidade menor que C de ferramentas caso as ferramentas requeridas pelas tarefas i ou por j não forem necessárias antes de i ou depois de j. Caso as ferramentas requeridas por i ou por j sejam requeridas antes de i ou depois de j, o número de ferramentas subtraídas será maior.

$$d_{ij} = \max\left\{0, \mid F_i \cup F_j \mid - \left[\theta \frac{\Lambda(ij)}{(n-2) \mid F_i \cup F_j \mid}\right] C\right\} \tag{4.5}$$

São utilizadas duas funções auxiliares. A primeira $\lambda_k(ij)$, indica o número de tarefas, excluindo-se as tarefas i e j, que requerem a ferramenta $k \in F_i \cup F_j$. A função $\Lambda(ij) = \sum_{k \in F_i \cup F_j} \lambda_k(ij)$, representa o total das frequência de todas as ferramentas utilizadas para processar duas tarefas i e j. Além disso, tem-se o parâmetro θ, com valor entre 0 e 1. Assim, a definição subtrai de $\mid F_i \cup F_j \mid$ uma quantidade entre $[0, C]$, sendo esta quantidade alta quando as ferramentas de $F_i \cup F_j$ são utilizadas com frequência.

4.5 Definição 5

Seguindo o mesmo raciocínio anterior, a definição representada pela Equação 4.6 também considera a frequência de utilização de ferramentas.

$$d_{ij} = \left(\left[\frac{c+1}{c}\right] \mid F_i \cup F_j \mid - \mid F_i \cap F_j \mid\right) \left[\frac{(n-2) \mid F_i \cup F_j \mid}{\max\{\Lambda(ij), 0.5\}}\right] \tag{4.6}$$

O fator $\frac{c+1}{c}$ é um valor entre $[1, 2]$ que proporciona um peso maior para o termo $\mid F_i \cup F_j \mid$ caso a capacidade máxima da máquina seja pequena, i.e., se mais trocas de ferramentas são prováveis. A segunda parte da equação é no mínimo 1, tornando-se maior se as ferramentas

presentes em $F_i \cup F_j$ forem raramente utilizadas. O parâmetro 0.5 é utilizado para evitar a divisão por 0, quando $\Lambda(ij) = 0$. Em experimentos realizados com as heurísticas consideradas em seu trabalho, Hertz et al. (1998) reportaram que esta definição para o custo das arestas apresentou os melhores resultados dentre todas as descritas neste capítulo.

4.6 Implementação

As cinco diferentes definições para cálculo da distância na modelagem do MTSP como o problema do Caixeiro Viajante mencionadas foram utilizadas para derivar cinco métodos diferentes para a solução do MTSP. São elas: d_1 (descrita pela Equação 4.1), d_2 (descrita pela Equação 4.2), d_3 (descrita pela Equação 4.4), d_4 (descrita pela Equação 4.5) e d_5 (descrita pela Equação 4.6).

Experimentos preliminares foram conduzidos para ajustar o valor do parâmetro θ, de maneira a gerar os melhores resultados quando utilizada a método d_4. Entretanto, concluiu-se que a variação do valor de θ no intervalo $[0, 1]$ não gerou alteração significativa nas soluções obtidas para as instâncias consideradas.

Todos os métodos foram implementados utilizando a linguagem C++, além de incorporar as funções do resolvedor Concorde e do resolvedor de programação linear CPLEX, versão 12.6.3.

Capítulo 5

Experimentos Computacionais

Os experimentos foram realizados em um computador com processador *Intel i3-2350M Quad Core* de 2.30GHz, 4 GB de RAM e sistema operacional Ubuntu 16.04.02. Os resultados obtidos foram comparados com os obtidos pelo atual estado da arte relacionado ao MTSP, a meta-heurística *Iterated Local Search* (ILS) proposta por Paiva e Carvalho (2017). Os resultados desse método foram obtidos em um computador com processador *Intel i5 Quad Core* de 3.2GHz e 8 GB de RAM.

As imagens a seguir ilustram os valores das soluções geradas para cada um dos grupos de instâncias considerados em forma de gráficos *boxplot*. No eixo x as siglas d_1, d_2, d_3, d_4, d_5 e ILS indicam qual algoritmo está evidenciado, representado os métodos d_1, d_2, d_3, d_4 e d_5 respectivamente, enquanto o eixo y representa o valor da soluções encontrada. As demais informações sobre os grupos de instâncias são definidas no início de cada seção a seguir. O valor das soluções e o tempo de execução para cada grupo de instâncias são apresentados detalhadamente no Apêndice A, em forma de tabelas.

Boxplot é um gráfico utilizado para avaliar a distribuição empírica de dados, e é poderoso por concentrar muita informação com simplicidade. Pode ser utilizado também para uma comparação visual entre dois ou mais grupos de dados, como os resultados obtidos por dois algoritmos. Os *boxplots* apresentam uma linha vermelha, que representa a mediana das soluções. Desta forma, metade dos valores estão abaixo da linha vermelha e a outra metade acima da linha, e cada uma das divisões é denominada *quartil*. As retas que surgem a partir dos quartis, denominadas *whiskers*, indicam a variabilidade dos valores fora dos quartis inferior e superior. Os pontos vermelhos correspondem aos *outliers*, valores que apresentam um grande afastamento dos demais, ou seja, valores atípicos.

Adicionalmente, foram levados em consideração o *gap* (distância percentual em relação à melhor solução existente) e o *ranking* ordinal. Dada uma solução S e uma solução de referência S^*, o *gap* é calculado como $100\times(S - S^*)/S^*$. O *ranking* ordinal estabelece uma ordem entre os métodos comparados, indicando quais algoritmos geram as melhores soluções com mais frequência, embora não indique a qualidade das soluções geradas. Em outras palavras, o

ranking ordinal atribui valores ordinais aos métodos de acordo com a qualidade das soluções associadas. Em caso de empate, o mesmo valor ordinal é atribuído aos métodos.

Após a apresentação dos resultados gerados para cada conjunto de instâncias e respectivas análises individuais, a Seção 5.4 apresenta uma discussão geral sobre o desempenho dos métodos.

5.1 Instâncias de Yanasse et al. (2009)

Yanasse et al. (2009) propuseram um total de 1.350 instâncias para o MTSP, divididas em 5 grupos (A, B, C, D, e E). Entre os grupos, varia-se a capacidade da máquina, o número de tarefas, o número de ferramentas e o número de instâncias. A Figura 5.1 ilustra os resultados obtidos para Grupo A, em que o número de tarefas é fixado em 8 para todas as instâncias consideradas.

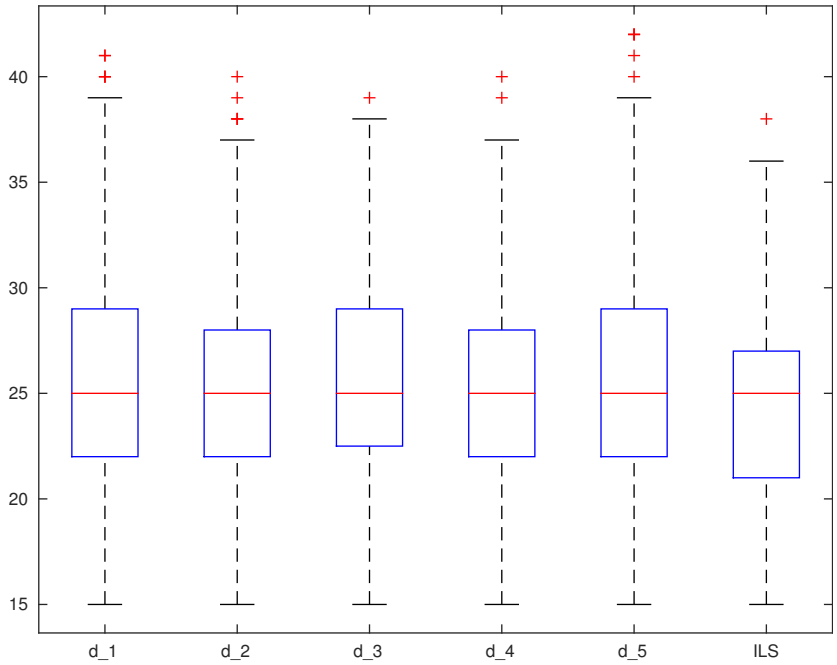

Figura 5.1: *Boxplot* referente as instâncias do Grupo A.

Por ser um grupo com instâncias consideradas fáceis, o desempenho dos métodos baseados no PCV foram parecidos. Mesmo assim, é evidente que o ILS apresentou melhores soluções, com mais valores abaixo da mediana e menos *outliers*. Comparando somente os métodos analisados na modelagem via PCV, todos obtiveram desempenhos parecidos, porém os valores acima da mediana e a quantidade de *outliers* apresentados pelos métodos d_1, d_3 e d_5 indicam pior desempenho dos métodos. Além disso, os métodos d_2 e d_4 são os que mais se aproximam do ILS. Ressalta-se também que o método d_4 alcançou, dentre os métodos comparados, o menor *gap* em 66,67% dos subgrupos, ou seja, em 6 dos 9 subgrupos. Os métodos baseados

no PCV mostrou agilidade na geração das soluções, com média de 0,00 segundos de tempo de execução para todos os métodos.

A Figura 5.2 ilustra os resultados obtidos para as instâncias do Grupo B. Neste grupo, as instâncias do grupo possuem o número de tarefas fixo em 15 unidades.

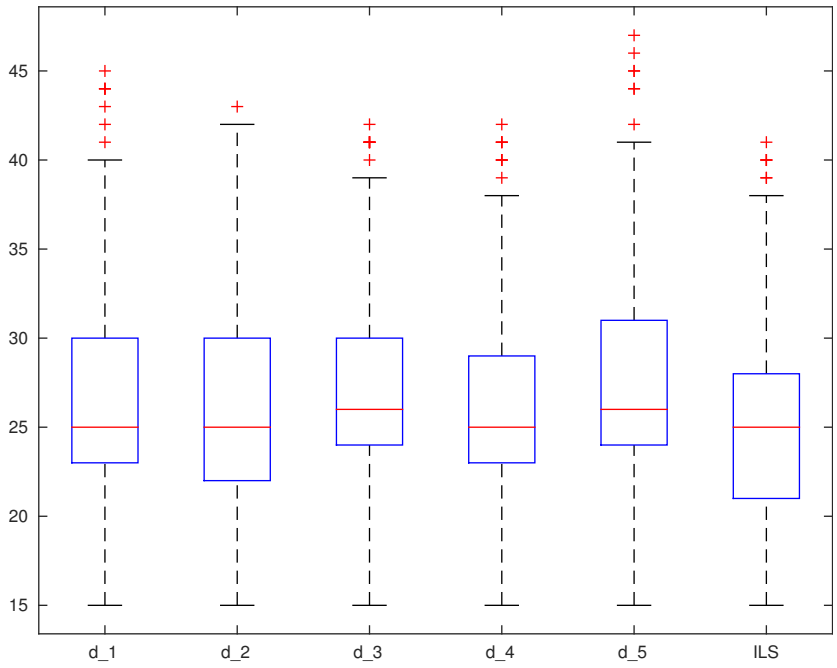

Figura 5.2: *Boxplot* referente as instâncias do Grupo B.

Analogamente aos resultados apresentados no Grupo A, com instâncias consideradas fáceis, o gráfico indica novamente desempenho parecido para todos os métodos, embora o ILS tenha se mostrado superior. É fácil perceber a má qualidade do método d_5 pela quantidade de *outliers* e pelo espaçamento nos *quartis* inferior e superior. Apesar de serem muito parecidos, o *quartil* inferior dos métodos d_2 e d_4 indica novamente que estes foram os métodos que chegaram mais próximo das soluções apresentadas pelo ILS. O tempo de execução médio de 0,00 segundos para todos os métodos baseados no PCV indica mais uma vez a agilidade em gerar soluções para o problema, mesmo utilizando um resolvedor exato.

Os resultados obtidos para o Grupo C são ilustrados na Figura 5.3. O número de tarefas em cada instância do grupo é fixado em 15.

A discrepância dos resultados obtidos pelo ILS em comparação com os métodos baseados no PCV se torna mais clara. O método de referência apresenta valores mais compactos, valor de mediana menor e menor amplitude dos *quartis* e *whiskers*. Entre os métodos baseados no PCV, novamente os métodos d_2 e d_4 apresentaram os melhores desempenhos e os métodos d_3 e d_5 apresentaram os piores desempenhos. Em relação ao tempo de execução, a característica observada nos conjuntos anteriores é mantida, com médias em torno dos 0,03 segundos em todos os métodos.

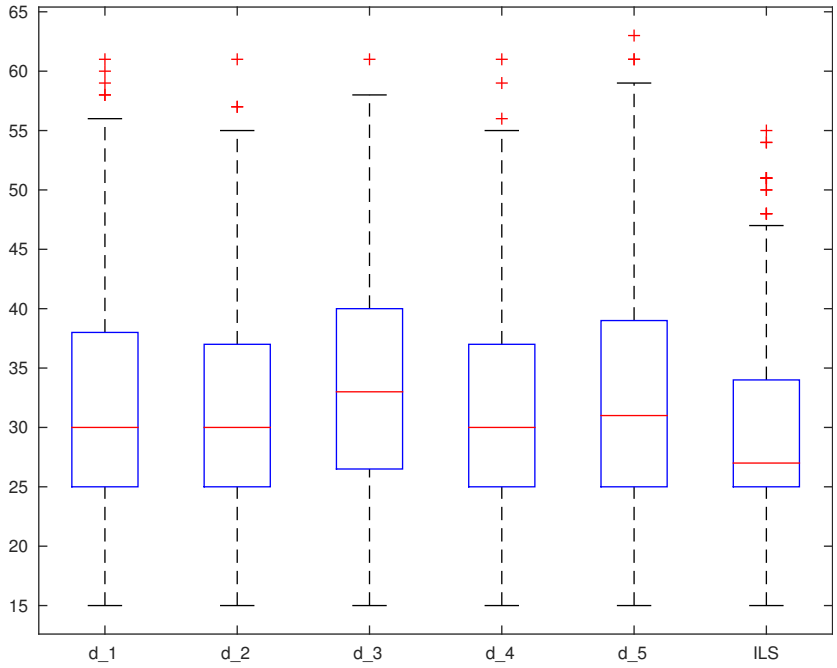

Figura 5.3: *Boxplot* referente as instâncias do Grupo C.

O grupo D apresenta mais variações entre número de tarefas, número de ferramentas e capacidade da máquina. Os respectivos resultados para este grupo são ilustrados na Figura 5.4.

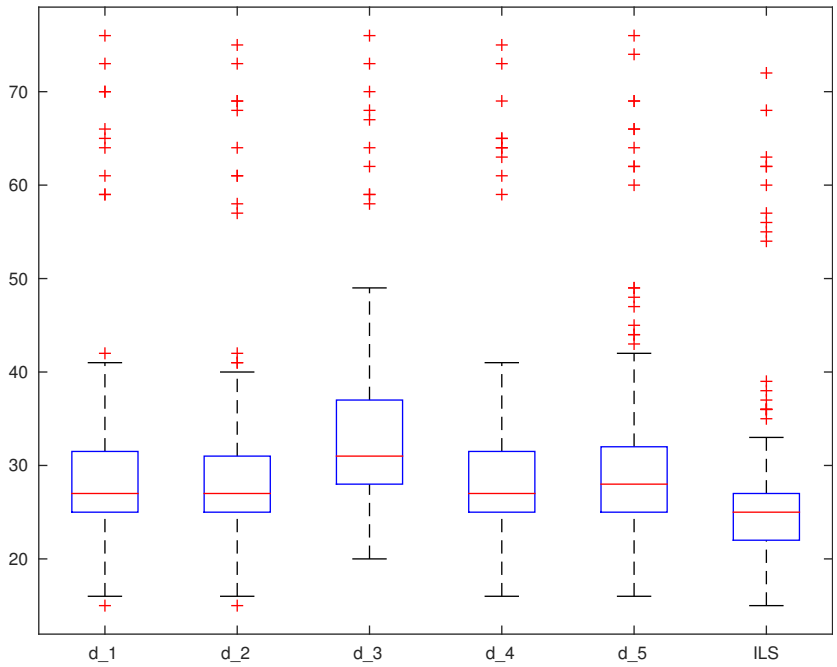

Figura 5.4: *Boxplot* referente as instâncias do Grupo D.

Por este ser considerado um grupo de maior dificuldade, a distância das soluções apresentados por ILS em comparação com os métodos baseados no PCV é evidenciada. O ILS apresenta valor da mediana próximo do inicio do *quartil* inferior dos métodos baseados no PCV e distribuição de valores compacta. Os métodos d_1, d_2, d_4 e d_5 apresentaram soluções parecidas, enquanto d_3 apresentou o pior desempenho. Os tempos de execução se mantém baixos, sendo altamente competitivos com a média encontrada pelo ILS: 6,05 segundos.

Os resultados obtidos para o grupo E são ilustrados na Figura 5.5. As instâncias do grupo possuem 10 e 15 tarefas.

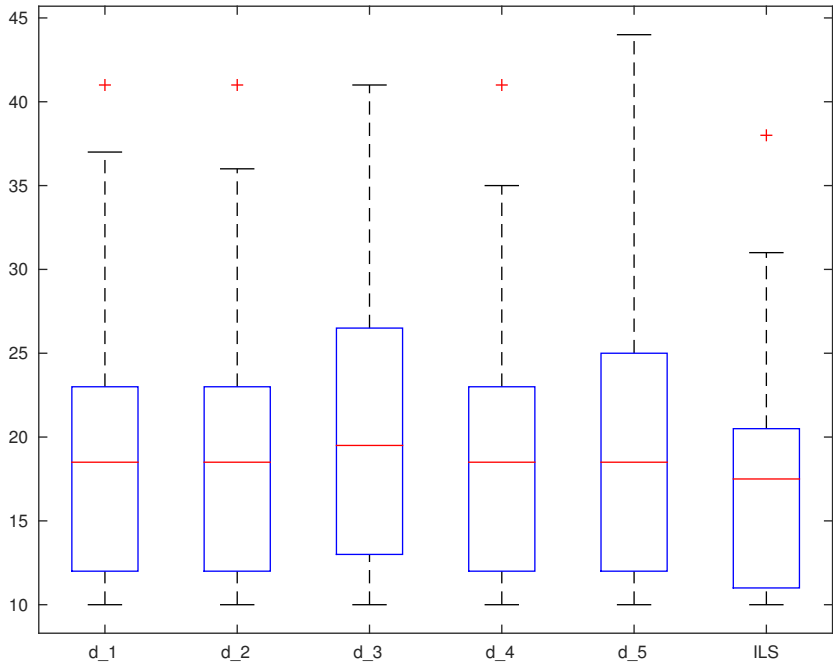

Figura 5.5: *Boxplot* referente as instâncias do Grupo E.

Assim como em todos os grupos de instâncias de Yanasse et al. (2009), o ILS mostrou desempenho superior também no Grupo E, com os *quartis* com menor amplitude quando comparado aos métodos baseados no PCV. Também é evidente o pior desempenho dos métodos d_3 e d_5, e o desempenho similar entre d_1, d_2 e d_4. Neste conjunto de instâncias, novamente verifica-se o baixo tempo de execução para todos os métodos baseados no PCV e a dificuldade dos métodos em obter soluções próximas das melhores conhecidas.

A Tabela 5.1 apresenta o *gap* dos métodos em todos os grupos de instâncias. Nenhum dos métodos baseados no PCV foi capaz de igualar a qualidade das soluções apresentadas pelo ILS, que obteve todas as melhores soluções para o problema. Comparando somente os métodos baseados no PCV, tem-se que os métodos d_4 e d_2 apresentaram melhores soluções com mais frequência, com *gap* variando entre 3,26% e 13,51% para d_4. Para d_2 a variação se deu entre 3,33% e 13,51%. Adicionalmente, o método d_4 apresentou os menores valores de *gap* em 60,00% no total dos subgrupos de instâncias, ou seja, 30 em 50 subgrupos. O método

d_3 obteve o pior desempenho, com *gap* até 19,37%.

Tabela 5.1: *Gap* obtidos em cada grupo.

	ILS	d_1	d_2	d_3	d_4	d_5
Grupo A	0,00%	4,51%	3,33%	4,21%	3,26%	8,89%
Grupo B	0,00%	5,58%	4,18%	5,43%	3,83%	10,99%
Grupo C	0,00%	14,02%	12,15%	17,72%	12,00%	17,62%
Grupo D	0,00%	14,18%	13,90%	33,27%	13,51%	17,73%
Grupo E	0,00%	9,00%	8,29%	19,37%	7,58%	14,61%

O *ranking* ordinal médio dos métodos comparados é apresentado na Tabela 5.2, que corrobora o argumento de má qualidade das soluções descritas acima. A tabela indica que nenhum método conseguiu as melhores soluções em nenhum dos grupos.

Tabela 5.2: *Ranking* obtidos em cada grupo.

	ILS	d_1	d_2	d_3	d_4	d_5
Grupo A	1,00	1,85	1,63	1,78	1,61	2,32
Grupo B	1,00	2,05	1,79	2,00	1,74	2,61
Grupo C	1,00	2,85	2,46	3,33	2,42	3,31
Grupo D	1,00	2,65	2,57	4,22	2,52	3,13
Grupo E	1,00	2,10	2,01	2,99	1,95	2,73

A análise comparativa dos métodos baseados no PCV confirma os métodos d_4 e d_2 com melhores resultados e também destaca-se o mau desempenho dos métodos d_3 e d_5. Curiosamente, o método d_3, conforme mencionado na Seção 2 é o mais utilizado pelos trabalhos da literatura.

5.2 Instâncias de Crama et al. (1994)

Divididas em 4 grupos (C_1, C_2, C_3 e C_4), as 160 instâncias propostas por Crama et al. (1994) são diferenciadas pelo número de tarefas, número de ferramentas e capacidade máxima da máquina flexível. Cada grupo possui 40 instâncias.

A Figura 6.6 ilustra as soluções encontradas para as instâncias do grupo C_1. Ressalta-se que o número de tarefas e o número de ferramentas é fixado para todas as instâncias do grupo em 10. Sendo assim, varia-se somente a capacidade da máquina.

Apesar de as instâncias neste conjunto possuírem o número de tarefas similar ao das instâncias do Grupo A, apresentado anteriormente, o desempenho dos métodos baseados no PCV é pior. Embora os melhores resultados não tenham sido alcançados, o método d_2 apresentou melhor desempenho para o conjunto analisado. Todos os métodos apresentaram baixo tempo de execução, sempre abaixo de 0,02 segundos.

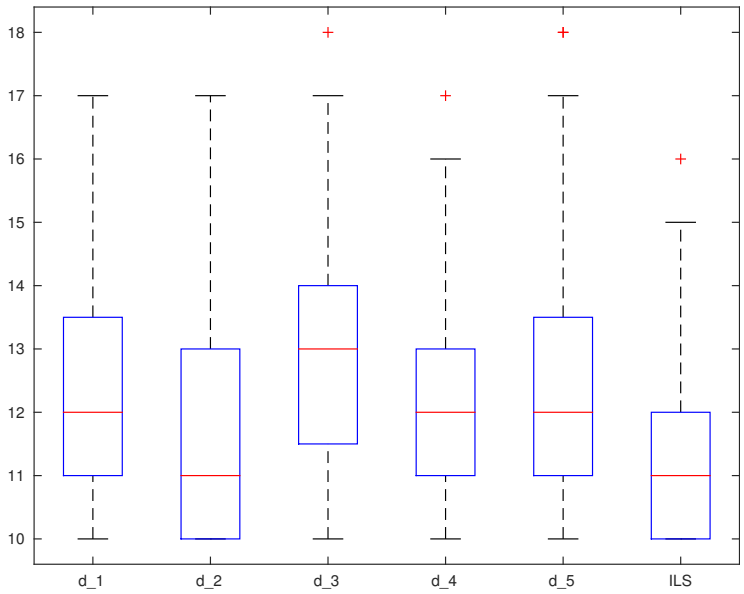

Figura 5.6: *Boxplot* referente as instâncias do Grupo C_1.

A Figura 6.7 ilustra os resultados obtidos para as instâncias do grupo C_2. Novamente, as instâncias variam apenas a capacidade da máquina, uma vez que o número de tarefas e o número de ferramentas são fixadas em 15 e 20, respectivamente.

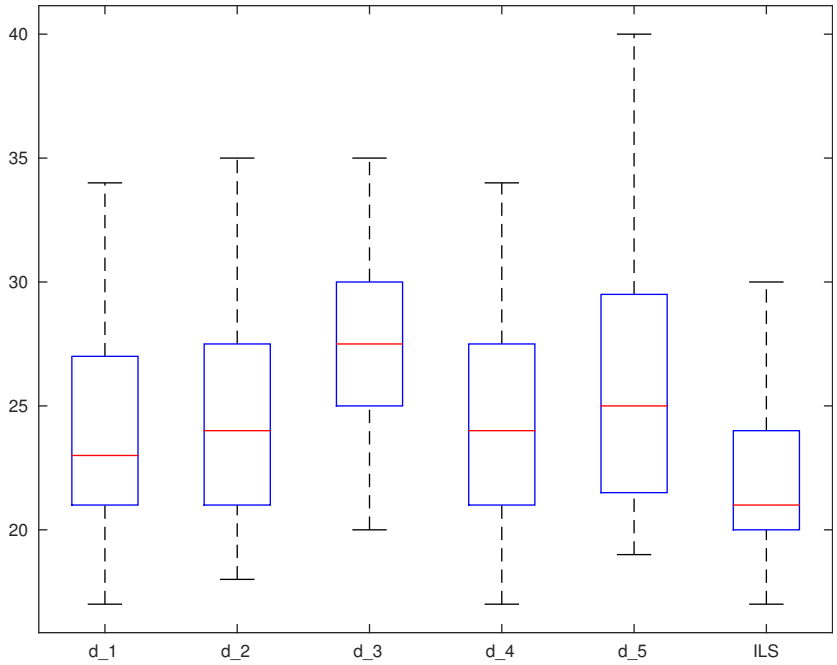

Figura 5.7: *Boxplot* referente as instâncias do Grupo C_2.

Novamente, o ILS apresentou os melhores resultados de forma evidente. Pela primeira

vez nos experimentos realizados, o método d_1 obteve melhor desempenho dentre os métodos baseados no PCV, embora todos estes tenham apresentado resultados distantes dos melhores conhecidos na literatura. Os métodos d_2 e d_4 obtiveram resultados comparáveis. Analogamente aos grupos analisados anteriormente, os métodos d_3 e d_5 obtiveram desempenhos ruins, muito distantes dos demais métodos baseados no PCV. O tempo de execução se manteve baixo para todos os métodos baseados no PCV, reforçando a eficiência do resolvedor Concorde para estas instâncias.

O Grupo C_3 fixa o número de tarefas e o número de ferramentas em 30 e 40 respectivamente, variando a capacidade da máquina. A Figura 6.8 ilustra todos os resultados obtidos para esse grupo.

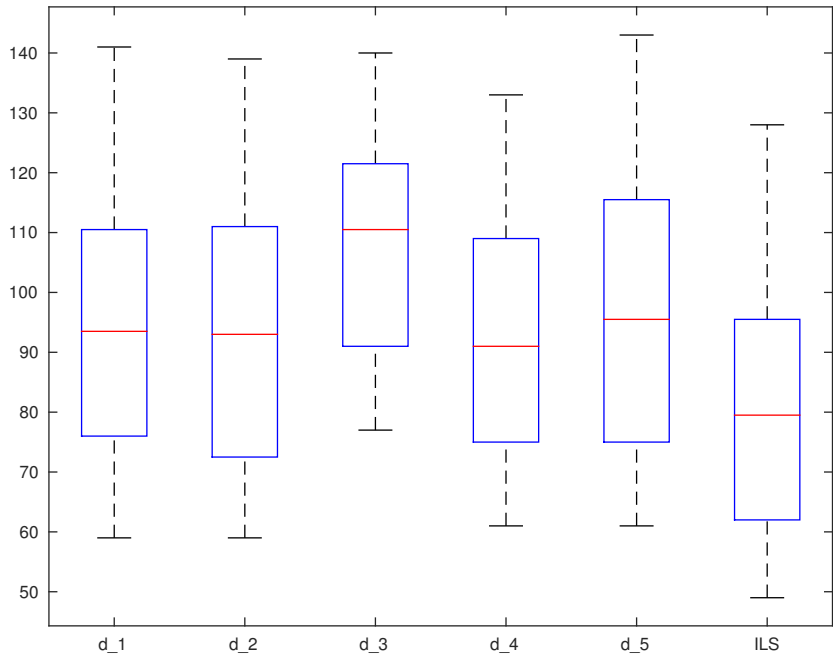

Figura 5.8: *Boxplot* referente as instâncias do Grupo C_3.

A disparidade do ILS em comparação com os métodos baseados no PCV pode ser verificada pelo valor da mediana, equiparado ao inicio do *quartil* inferior dos demais métodos. Comparando somente os métodos baseados no PCV, d_2 e d_4 permanecem apresentando as melhores soluções e d_3 e d_5 permanecem com desempenhos ruins.

Neste grupo, os métodos baseados no PCV apresentam as piores performances para as instâncias analisadas até aqui. De fato, nenhuma das definições foi capaz de gerar soluções próximas das melhores conhecidas, o que indica que a modelagem do MTSP via PCV não produz heurísticas de boa performance para as instâncias *benchmark* atuais.

Assim como no Grupo C_3, as instâncias do Grupo C_4 apresentam um nível de dificuldade de resolução alto. A Figura 6.9 ilustra os resultados obtidos para estas instâncias, que possuem número de tarefas e número de ferramentas fixados em 40 e 60 respectivamente, ao passo em

que a capacidade da máquina varia.

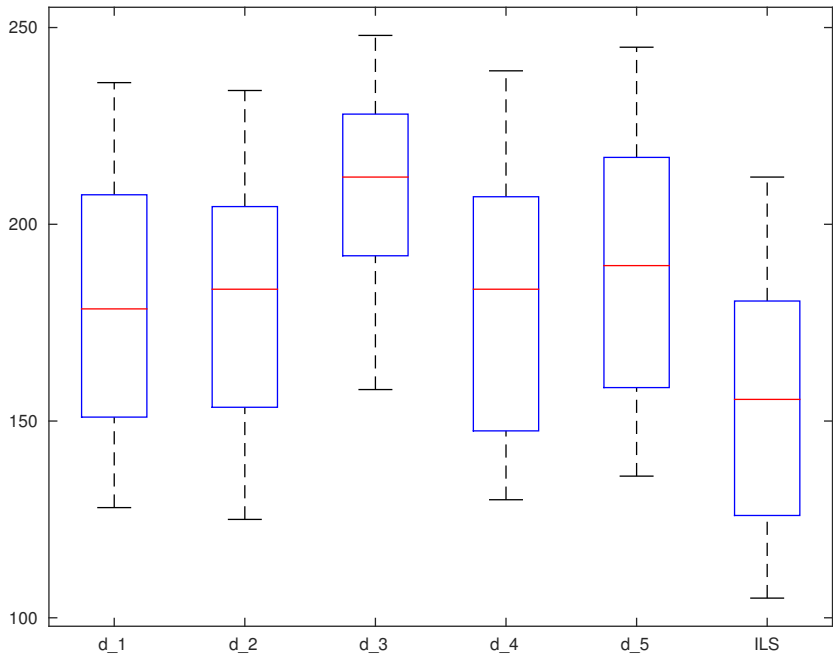

Figura 5.9: *Boxplot* referente as instâncias do Grupo C_4.

A análise dos gráficos *boxplots* indicam novamente que nenhum dos métodos foi capaz de apresentar soluções competitivas frente ao ILS. Analogamente ao Grupo C_3, os métodos d_3 e d_5 apresentaram piores desempenhos, enquanto que d_1, d_2 e d_4 apresentaram desempenhos parecidos e melhores, quando comparados os métodos baseados no PCV.

Embora estas instâncias possuam um número maior de tarefas e ferramentas, os resultados obtidos foram levemente melhores do que os obtidos para o conjunto anterior. Entretanto, os indicadores se mantêm altos, mantendo a tendência para os grupos de instâncias deste conjunto. A análise comparativa dos métodos baseados no PCV também mantém a mesma tendência anterior.

A Tabela 5.3 apresenta os valores de *gap* dos métodos comparados. Novamente, nenhum dos métodos conseguiu se aproximar razoavelmente do ILS.

Tabela 5.3: *Gap* obtidos em cada grupo.

	ILS	d_1	d_2	d_3	d_4	d_5
Grupo C_1	0,00%	8,42%	6,63%	15,80%	7,42%	11,69%
Grupo C_2	0,00%	9,17%	10,22%	25,06%	10,97%	15,40%
Grupo C_3	0,00%	18,78%	18,01%	37,72%	16,42%	21,14%
Grupo C_4	0,00%	16,83%	15,57%	34,65%	16,27%	21,10%

Os dados apresentados indicam valores de *gap* de até 18,78% para os três melhores métodos (d_1, d_2 e d_4), enquanto o método d_3, que apresenta o pior desempenho, atingiu 37,72% de

gap. Comparando os melhores métodos, d_4 se mostrou ligeiramente superior do que d_2 e d_1, respectivamente segundo e terceiro melhor desempenho. A Tabela 5.4 apresenta o *ranking* ordinal, o qual comprova as análises dos *boxplots* e *gap* apresentados anteriormente.

Tabela 5.4: *Ranking* obtidos em cada grupo.

	ILS	d_1	d_2	d_3	d_4	d_5
Grupo C_1	1,00	1,90	1,70	2,55	1,80	2,23
Grupo C_2	1,00	2,15	2,35	3,65	2,35	2,93
Grupo C_3	1,00	3,40	3,03	5,33	2,58	3,98
Grupo C_4	1,00	3,25	2,63	5,53	3,08	4,25

Destaca-se o desempenho ruim dos métodos baseados na modelagem analisada neste trabalho, principalmente em relação aos Grupos C_2, C_3 e C_4. Nos métodos d_2 e d_4, que são os de melhor desempenho, o *ranking* ordinal médio varia entre 2,35 e 3,08, indicando que não se aproximaram das soluções obtidas pelo ILS. Os métodos d_3 e d_5 ficaram muito aquém dos demais, com *ranking* ordinal médio de até 5,53.

De maneira geral, mesmo com os tempos de execução baixos, sempre em torno de décimos de segundos, os métodos apresentaram baixa qualidade das soluções, mesmo para instâncias de dimensões semelhantes às do primeiro conjunto de instâncias consideradas. A qualidade das soluções se deteriorou de maneira mais acentuada na resolução de instâncias dos grupos C_3 e C_4. Comparando-se os métodos baseados no PCV, o método d_2 se destacou dos demais apresentando menores soluções em 75,00% dos subgrupos de instâncias, seguido pelo método d_4. Novamente, a pior performance foi relacionada ao método d_3, embora o método d_5 também tenha apresentado soluções de baixa qualidade em geral.

5.3 Instâncias de Catanzaro et al. (2015)

As 160 instâncias propostas por Catanzaro et al. (2015) foram divididas em 4 grupos (*datA*, *datB*, *datC* e *datD*) com 40 instâncias cada. Os grupos de instâncias se diferenciam pelo número de tarefas, número de ferramentas e capacidade da máquina.

Todas instâncias presentes no grupo *datA* possuem 10 tarefas e ferramentas. Os resultados obtidos no grupo *datA* são ilustrados na Figura 6.10.

Neste primeiro grupo de instâncias, o ILS demonstrou-se muito superior, apresentando uma distribuição de valores compacta, com praticamente com mesmos valores de mediana e menor valor. Quanto aos métodos baseados no PCV, d_3 e d_5 possuem pior desempenho e d_1, d_2 e d_4 possuem qualidade das soluções semelhantes. Levando em consideração o tempo de execução, todos os métodos baseados no PCV apresentam médias em torno de 0,02 segundos.

A Figura 6.11 ilustra os resultados para as instâncias do grupo *datB*, as quais possuem fixados o número de tarefas e o de ferramentas em 15 e 20 unidades.

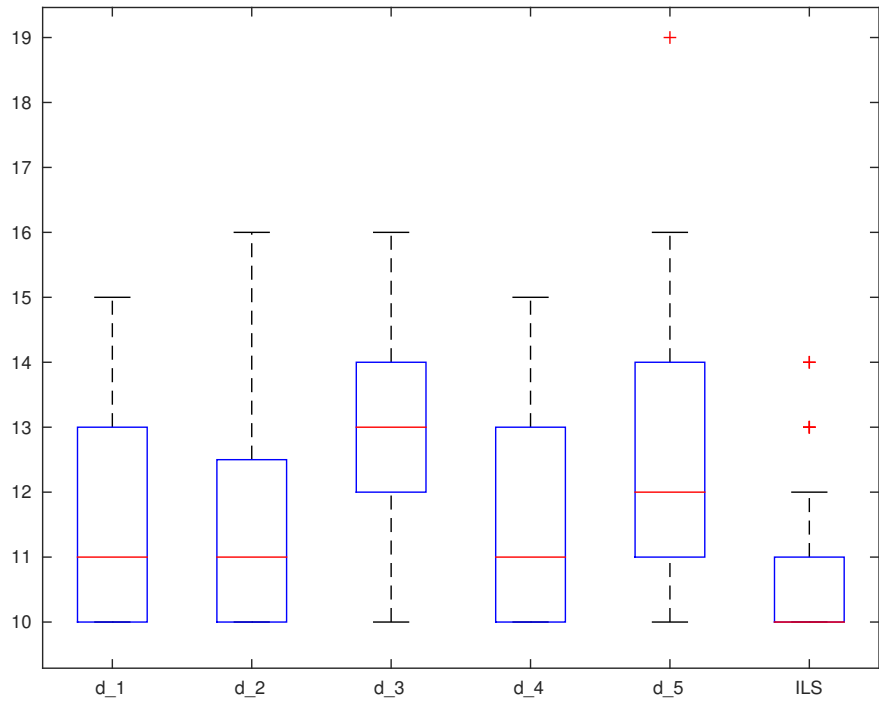

Figura 5.10: *Boxplot* referente as instâncias do Grupo *datA*.

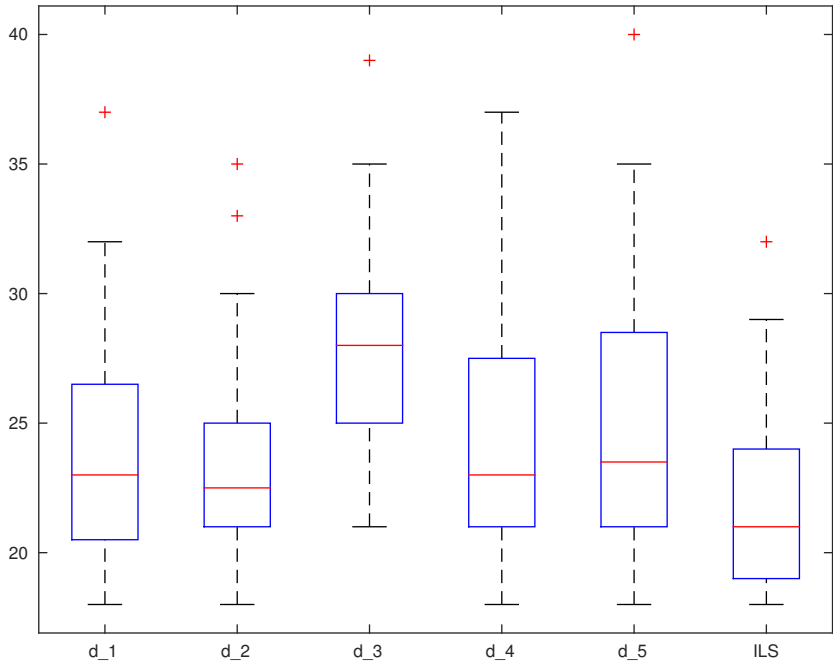

Figura 5.11: *Boxplot* referente as instâncias do Grupo *datB*.

À exceção dos métodos d_2 e d_5, os demais métodos tiveram uma piora significativa na

qualidade das soluções geradas. Apesar de o método d_2 se destacar frente aos demais, a mesma análise anterior se aplica a este grupo de instâncias, inclusive com melhor desempenho de d_1 em relação a d_4.

Os resultados ilustrados na Figura 6.12 são as soluções encontradas para as instâncias do grupo *datC*, em que o número de tarefas e de ferramentas são fixados em 30 e 40 unidades.

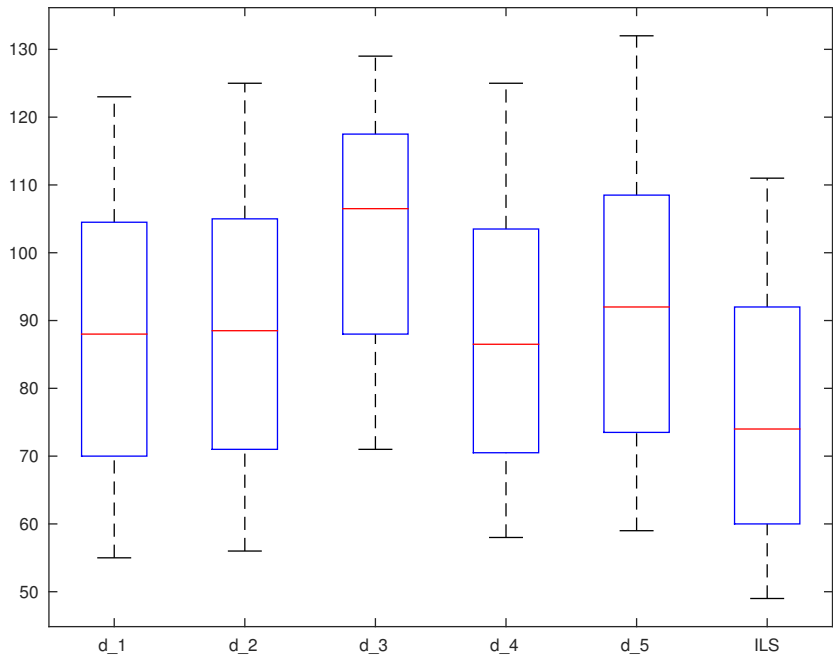

Figura 5.12: *Boxplot* referente as instâncias do Grupo *datC*.

Neste que é considerado um dos conjuntos de instâncias mais difíceis da literatura, os métodos baseados no PCV obtiveram as piores performances em todos os experimentos. Comparando os métodos baseados no PCV, somente d_3 de distanciou dos demais, enquanto que os demais métodos apresentaram desempenho semelhantes, com d_4 destacando ligeiramente.

Para o último grupo de instâncias proposto por Catanzaro et al. (2015), as soluções são ilustradas na Figura 6.13. O número de tarefas e o número de ferramentas são idênticos para todas as instâncias do grupo: 40 e 60 unidades.

O comportamento dos métodos baseados no PCV, apesar de levemente melhor do que o apresentado em relação ao grupo de instâncias anterior, segue o mesmo padrão, com d_1, d_2 e d_4 apresentando qualidade parecidas em suas soluções e d_3 e d_5 com pior desempenho evidente. A Tabela 5.5 indica os valores de *gap* para cada um dos grupos de instâncias considerados.

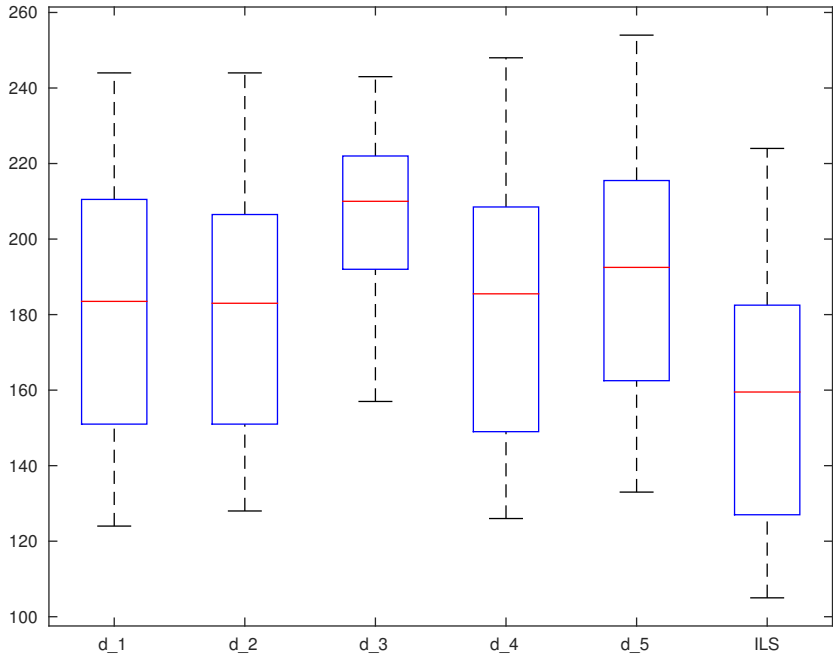

Figura 5.13: *Boxplot* referente as instâncias do Grupo *datD*.

Tabela 5.5: *Gap* obtidos em cada grupo.

	ILS	d_1	d_2	d_3	d_4	d_5
datA	0,00%	6,72%	7,23%	18,96%	7,24%	14,53%
datB	0,00%	10,62%	7,71%	26,83%	11,11%	14,62%
datC	0,00%	20,23%	19,71%	39,64%	18,74%	23,48%
datD	0,00%	16,40%	15,38%	33,90%	16,06%	20,57%

Novamente os dados apresentados confirmam o desempenho ruim dos métodos baseados no PCV quando comparados ao ILS. O destaque negativo se refere ao método d_3, que atingiu 39,64% de *gap*, enquanto que o destaque positivo continuam sendo as definições d_2 e d_4, com médias de até 20,00% de *gap*. A Tabela 5.6 indica os valores do *ranking* ordinal médio para cada um dos grupos de instâncias considerados.

Tabela 5.6: *Ranking* obtidos em cada grupo.

	ILS	d_1	d_2	d_3	d_4	d_5
datA	1,00	1,70	1,78	2,73	1,78	2,40
datB	1,00	2,35	1,93	3,95	2,53	2,98
datC	1,00	3,28	3,00	5,40	2,75	3,95
datD	1,00	3,15	2,80	5,35	3,00	4,45

Uma avaliação dos dados da tabela reforça a análise realizada sobre a Tabela 5.5. Os valores de 5,35 e 5,40 para o *ranking* ordinal foram os piores, considerando a comparação entre 6 métodos. Comparando os métodos de melhor desempenho, os resultados obtidos em

d_2 foram um pouco melhor do que os resultados obtidos em d_4, apresentando dois resultados melhores (grupos *datB* e *datD*), um resultado igual (grupo *datA*) e um resultado pior (grupo *datC*).

Neste conjunto de instâncias, de maneira geral, assim como nos demais conjuntos considerados, apesar do baixo tempo de execução, os resultados não são sequer razoavelmente próximos daqueles reportados na literatura recente. Particularmente nos grupos *datC* e *datD*, considerados como os mais difíceis atualmente, a performance dos métodos baseados no PCV foi muito inferior ao atual estado da arte. Comparando-se os métodos entre si, novamente o método d_2 obteve os melhores indicadores, seguido pelo método d_4, ao passo que o método d_3 obteve os piores indicadores, muito aquém dos demais métodos comparados.

5.4 Análise Geral

À luz dos experimentos e com as respectivas análises específicas nas seções anteriores, é possível realizar uma análise geral sobre a modelagem do Problema de Minimização de Trocas de Ferramentas como o Problema do Caixeiro Viajante. Devido à utilização do resolvedor exato Concorde, é possível verificar a consistência e qualidade da modelagem em si, dado que os resultados são ótimos para a modelagem utilizada. Isto não foi possível anteriormente na literatura devido à utilização de métodos heurísticos para solução dos modelos, o que por si só poderia gerar degradação da qualidade da solução.

Os dados indicam que as soluções obtidas utilizando a modelagem proposta e as definições encontradas na literatura possuem baixa qualidade, já que nenhum dos métodos baseados no PCV conseguiu igualar ou se aproximar razoavelmente dos resultados obtidos pelo atual estado da arte, mesmo para instâncias mais fáceis. Muito embora os métodos tenham sido capazes de gerar soluções em tempo de execução desprezível, esta característica não é suficiente para que conferir relevância aos mesmos.

Desta forma, conclui-se que a modelagem do MTSP utilizando PCV foi levada ao limite neste trabalho e que a mesma não é adequada, considerando-se o que já foi proposto neste tema. Todavia, é possível concentrar esforços em novas formas de cálculo de distância que sejam capazes de suprir as deficiências das cálculos propostos anteriormente na literatura. Dentre esses métodos destaca-se a utilização de heurísticas matemáticas e outras relacionadas a evolução das soluções dentro do próprio método.

Capítulo 6

Conclusões

Neste trabalho considerou-se o Problema de Minimização de Trocas de Ferramentas (ou MTSP), um problema de sequenciamento de tarefas em linhas de produção industrial. Na tentativa de solucionar este problema de aplicação prática, foram apresentadas diferentes modelagens desde o final da década de 80. Na literatura, uma forma comum de modelar o MTSP baseia-se no clássico Problema do Caixeiro Viajante (PCV). Embora os dois problemas não sejam equivalentes entre si no caso geral, considera-se cada tarefas como uma cidade, tal que existem diferentes formas para adaptação do número de troca de ferramentas, característica do MTSP, para a distância entre cidades, característica do PCV. Na literatura, estas diferentes adaptações foram aplicadas na modelagem do MTSP pelo PCV, que, via de regra, era resolvido por heurísticas clássicas, como 2-*opt* e *nearest-neighbor*, por exemplo.

Neste trabalho, propôs-se revisitar esta tradicional modelagem e também as cinco definições tradicionais para conversão do número de trocas de ferramentas em distâncias entre cidades. Entretanto, utilizou-se o resolvedor exato Concorde para solução do PCV. Desta forma, foi possível avaliar a qualidade da modelagem empregada, dado que a solução pelo Concorde para o PCV é exata, para cada uma das definições de distância encontradas na literatura. Os experimentos computacionais consideraram três grandes conjuntos de dados, em um total de 1.670 instâncias, divididas em treze subgrupos, e os resultados foram comprados a uma implementação do método *Iterated Local Search* (ILS), considerado o estado da arte para o problema em questão.

Os resultados computacionais reportaram tempo de execução baixo para os métodos implementados, com médias em torno dos décimos de segundos para todos os métodos em todos os grupos de instâncias, sendo sempre menores que as médias reportada pelo ILS. Entretanto, nenhum método proposto foi capaz de igualar ou se aproximar razoavelmente dos melhores resultados reportados na literatura. As análises dos resultados indicam que duas definições de distância apresentaram melhores resultados e se destacaram das demais e também, que a métrica mais comumente utilizada na literatura obteve os piores indicadores em relação aos demais métodos comparados. De maneira geral, foi possível observar que a modelagem do

MTSP pelo PCV não é adequada para as instâncias atuais, quando consideramos as definições existentes para adaptação do MTSP ao PCV.

Os trabalhos futuros se concentrarão em produzir definições de distância que apresentem resultados com melhor precisão e melhor desempenho. Dentre as opções destaca-se a utilização de métodos evolutivos que, utilizadas em conjunto com o resolvedor Concorde, darão origem a heurísticas matemáticas para resolução do MTSP.

Apêndice A

Resultados Detalhados

Nas tabelas a seguir são apresentados os valores das soluções e os respectivos tempos de execução para cada grupo de instâncias consideradas neste trabalho. São apresentados o número de ferramentas (m), a capacidade da máquina (C), a melhor solução (S^*) encontrada pelo *ILS* (Paiva e Carvalho, 2017), a solução encontrada pelos métodos propostos neste trabalho (S) e o tempo médio de execução (T) de cada método em segundos. As siglas d_1, d_2, d_3, d_4 e d_5 e *ILS* indicam qual algoritmo está evidenciado na coluna.

As Tabelas 6.1 a 6.5 são referentes aos grupos A, B, C, D, e E, respectivamente, das instâncias propostas por de Yanasse et al. (2009). Os grupos de instância de Crama et al. (1994), C_1, C_2, C_3 e C_4, são referenciadas pelas Tabelas 6.6 a 6.9, respectivamente. Por último, os grupos de instâncias propostas por Catanzaro et al. (2015), *datA*, *datB*, *datC* e *datD*, estão presentes nas Tabelas 6.10 a 6.13, respectivamente.

Tabela 6.1: Resultados obtidos para o Grupo A (Yanasse et al., 2009).

m	C	*ILS*		d_1		d_2		d_3		d_4		d_5	
		S^*	T	S	T	S	T	S	T	S	T	S	T
15	5	17,00	0,06	17,80	0,00	17,70	0,00	17,90	0,01	18,30	0,00	19,30	0,01
15	10	16,83	0,08	17,83	0,00	17,57	0,00	17,63	0,00	17,50	0,00	17,87	0,00
20	5	21,80	0,06	22,60	0,01	22,50	0,00	22,50	0,00	22,40	0,00	25,70	0,00
20	10	23,07	0,15	24,50	0,00	23,93	0,00	24,00	0,01	23,63	0,00	24,87	0,01
20	15	22,08	0,12	23,03	0,00	22,83	0,00	23,28	0,01	22,60	0,00	22,93	0,00
25	5	25,10	0,03	25,40	0,00	25,20	0,00	25,50	0,00	25,40	0,00	27,80	0,00
25	10	28,20	0,16	29,57	0,00	29,20	0,00	29,20	0,00	29,17	0,00	30,73	0,00
25	15	27,95	0,21	29,53	0,00	28,93	0,00	29,52	0,01	28,83	0,00	29,82	0,00
25	20	26,61	0,15	27,69	0,00	27,58	0,00	27,78	0,01	27,24	0,00	27,78	0,00

Tabela 6.2: Resultados obtidos para o Grupo B (Yanasse et al., 2009).

m	C	ILS		d_1		d_2		d_3		d_4		d_5	
		S^*	T	S	T	S	T	S	T	S	T	S	T
15	5	17,20	0,08	18,30	0,01	18,20	0,01	18,20	0,01	18,20	0,01	20,10	0,01
15	10	17,37	0,12	18,53	0,01	18,23	0,00	18,50	0,01	17,93	0,01	18,83	0,01
20	5	22,40	0,09	23,60	0,01	23,00	0,01	23,30	0,01	23,30	0,01	26,30	0,01
20	10	24,17	0,22	25,63	0,01	25,50	0,00	25,83	0,01	25,10	0,01	26,63	0,01
20	15	22,60	0,20	23,87	0,01	23,52	0,01	24,10	0,01	23,45	0,01	23,95	0,01
25	5	25,40	0,05	26,20	0,01	26,10	0,01	26,30	0,01	26,40	0,01	29,60	0,01
25	10	28,77	0,24	30,57	0,01	29,93	0,01	30,17	0,01	29,87	0,01	32,13	0,01
25	15	29,74	0,39	31,68	0,01	31,04	0,01	31,48	0,01	30,82	0,01	31,94	0,01
25	20	27,19	0,24	28,31	0,01	28,10	0,01	28,47	0,02	28,37	0,01	27,19	0,01

Tabela 6.3: Resultados obtidos para o Grupo C (Yanasse et al., 2009).

m	C	ILS		d_1		d_2		d_3		d_4		d_5	
		S^*	T	S	T	S	T	S	T	S	T	S	T
15	5	21,60	0,60	25,50	0,02	24,30	0,01	25,80	0,02	24,70	0,02	25,50	0,01
15	10	19,80	1,02	22,37	0,01	22,20	0,01	23,60	0,04	21,87	0,02	22,13	0,01
20	5	25,60	0,75	28,40	0,01	27,90	0,01	28,40	0,02	28,40	0,02	30,30	0,01
20	10	28,33	1,82	32,17	0,02	31,03	0,01	33,80	0,03	30,97	0,02	32,50	0,01
20	15	25,52	2,00	28,17	0,01	27,90	0,01	29,78	0,04	27,72	0,02	28,12	0,01
25	5	32,50	0,71	35,10	0,02	35,00	0,01	35,00	0,02	34,60	0,02	40,30	0,02
25	10	35,07	2,19	38,33	0,01	37,97	0,01	40,47	0,03	37,67	0,02	40,57	0,02
25	15	34,07	3,49	37,83	0,01	37,22	0,01	39,90	0,04	37,13	0,02	38,68	0,01
25	20	29,66	2,41	32,16	0,01	31,94	0,01	33,98	0,05	31,74	0,02	29,66	0,01

Tabela 6.4: Resultados obtidos para o Grupo D (Yanasse et al., 2009).

m	C	ILS		d_1		d_2		d_3		d_4		d_5	
		S^*	T	S	T	S	T	S	T	S	T	S	T
15	5	25,90	1,61	30,80	0,03	30,00	0,02	32,60	0,04	29,80	0,03	31,60	0,02
15	10	18,20	2,70	21,30	0,03	21,10	0,01	25,85	0,08	21,20	0,02	21,70	0,02
20	5	29,20	2,14	34,00	0,02	33,10	0,02	35,80	0,03	33,60	0,02	36,40	0,03
20	10	20,60	3,22	23,30	0,02	23,80	0,02	27,90	0,09	23,60	0,03	24,30	0,02
20	15	21,67	4,16	24,00	0,03	24,17	0,01	27,07	0,09	24,00	0,03	23,97	0,02
25	5	35,10	2,35	38,70	0,02	38,60	0,02	40,40	0,03	38,20	0,03	44,30	0,03
25	10	25,40	4,10	27,90	0,03	28,30	0,02	34,50	0,10	28,20	0,03	30,80	0,03
25	15	36,25	9,35	40,18	0,02	39,70	0,01	44,53	0,08	40,18	0,06	40,40	0,02
25	20	26,15	4,17	28,70	0,03	28,40	0,02	31,13	0,10	28,43	0,03	28,75	0,02
15	10	15,90	4,56	18,80	0,03	18,70	0,02	24,80	0,14	18,70	0,04	18,80	0,04
20	10	21,60	11,38	26,70	0,03	26,50	0,03	35,00	0,15	26,10	0,04	27,00	0,04
20	15	22,60	12,50	26,70	0,03	27,50	0,03	31,10	0,16	26,30	0,05	26,70	0,04
25	10	26,60	13,46	32,00	0,04	31,20	0,04	40,10	0,14	31,20	0,04	32,80	0,06
25	15	25,00	7,41	26,80	0,04	27,40	0,03	32,20	0,14	27,50	0,05	27,30	0,06
25	20	25,50	7,67	27,60	0,03	27,53	0,02	30,67	0,16	27,87	0,04	27,83	0,03

Tabela 6.5: Resultados obtidos para o Grupo E (Yanasse et al., 2009).

m	C	*ILS*		d_1		d_2		d_3		d_4		d_5	
		$S*$	T	S	T	S	T	S	T	S	T	S	T
10	4	13,50	0,10	15,50	0,01	15,30	0,01	15,20	0,01	15,20	0,01	15,80	0,01
10	5	11,20	0,10	12,70	0,01	12,00	0,01	12,70	0,01	11,90	0,01	12,40	0,01
10	6	10,30	0,06	10,80	0,01	11,00	0,01	12,50	0,02	10,90	0,01	11,60	0,03
10	7	10,00	0,03	10,30	0,01	10,20	0,01	11,40	0,02	10,30	0,01	10,60	0,01
20	6	27,40	1,01	30,80	0,01	31,30	0,01	32,40	0,02	30,60	0,02	34,20	0,01
20	8	22,20	1,19	24,80	0,02	24,30	0,01	29,20	0,05	24,40	0,08	27,20	0,01
20	10	20,30	0,82	21,70	0,01	21,80	0,01	25,10	0,06	21,60	0,02	23,30	0,01
20	12	20,20	0,73	21,30	0,01	21,50	0,01	24,50	0,06	21,30	0,02	22,00	0,01

Tabela 6.6: Resultados obtidos para o Grupo C_1 (Crama et al., 1994).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	S	T	S	T	S	T	S	T	S	T
4	13,10	0,10	14,70	0,01	14,60	0,01	15,20	0,01	14,30	0,01	15,20	0,01
5	11,20	0,10	12,40	0,01	11,90	0,01	13,20	0,01	12,40	0,01	13,00	0,01
6	10,30	0,06	11,10	0,01	10,90	0,01	12,30	0,02	10,90	0,01	11,40	0,01
7	10,10	0,04	10,40	0,01	10,40	0,01	11,10	0,02	10,50	0,01	10,50	0,01

Tabela 6.7: Resultados obtidos para o Grupo C_2 (Crama et al., 1994).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	S	T	S	T	S	T	S	T	S	T
6	26,60	1,03	29,40	0,01	29,30	0,01	32,70	0,02	29,90	0,02	33,00	0,01
8	21,70	1,45	24,50	0,01	25,30	0,01	28,00	0,05	25,30	0,02	26,80	0,01
10	20,10	1,04	21,80	0,01	22,60	0,01	26,00	0,06	22,30	0,02	23,40	0,02
12	19,60	0,61	20,50	0,01	20,70	0,01	24,30	0,06	20,70	0,02	20,70	0,02

Tabela 6.8: Resultados obtidos para o Grupo C_3 (Crama et al., 1994).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	S	T	S	T	S	T	S	T	S	T
15	106,40	104,03	121,90	0,06	119,50	0,04	128,00	0,15	118,50	0,12	124,40	0,08
17	88,50	160,88	103,60	0,07	102,80	0,04	115,30	0,08	100,50	0,09	106,60	0,08
20	70,50	228,83	86,10	0,05	83,90	0,05	102,30	0,20	84,30	0,10	87,10	0,08
25	52,90	206,65	64,20	0,07	65,90	0,04	82,10	0,25	64,10	0,10	65,40	0,08

Tabela 6.9: Resultados obtidos para o Grupo C_4 (Crama et al., 1994).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	S	T	S	T	S	T	S	T	S	T
20	198,70	533,78	222,60	0,13	219,30	0,08	231,80	0,15	221,10	0,27	230,90	0,14
22	174,00	801,76	198,90	0,12	196,20	0,09	222,50	0,12	198,40	0,20	207,00	0,14
25	146,50	1144,95	172,10	0,08	172,40	0,07	206,20	0,26	171,00	0,18	177,50	0,14
30	114,00	1908,61	138,50	0,09	136,20	0,07	172,40	0,35	138,00	0,20	143,60	0,14

Tabela 6.10: Resultados obtidos para o Grupo *datA* (Catanzaro et al., 2015).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T
4	12,50	0,09	13,70	0,02	13,60	0,01	14,00	0,01	13,60	0,01	15,10	0,01
5	10,80	0,09	11,70	0,01	11,90	0,01	13,40	0,02	11,80	0,01	12,90	0,01
6	10,10	0,05	10,80	0,01	10,90	0,01	12,60	0,03	11,00	0,01	11,40	0,01
7	10,00	0,04	10,20	0,01	10,20	0,01	11,50	0,03	10,20	0,01	10,50	0,01

Tabela 6.11: Resultados obtidos para o Grupo *datB* (Catanzaro et al., 2015).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T
6	26,50	1,02	29,90	0,02	28,70	0,01	31,70	0,02	30,10	0,02	31,70	0,01
8	21,70	1,38	24,40	0,01	23,50	0,01	28,90	0,05	25,10	0,02	26,20	0,01
10	19,70	1,07	22,30	0,01	21,60	0,01	26,80	0,06	22,00	0,02	22,60	0,02
12	19,20	0,55	20,30	0,01	20,40	0,01	23,10	0,06	20,20	0,02	20,20	0,02

Tabela 6.12: Resultados obtidos para o Grupo *datC* (Catanzaro et al., 2015).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T
15	98,80	111,27	114,60	0,05	112,20	0,04	120,20	0,06	112,60	0,09	118,90	0,07
17	82,60	164,64	97,90	0,09	98,20	0,04	113,60	0,09	96,30	0,12	101,40	0,09
20	66,70	229,75	82,30	0,06	82,30	0,04	100,60	0,23	82,00	0,08	84,20	0,07
25	51,30	191,48	63,10	0,05	63,10	0,04	76,20	0,25	62,30	0,10	63,90	0,06

Tabela 6.13: Resultados obtidos para o Grupo *datD* (Catanzaro et al., 2015).

C	*ILS*		d_1		d_2		d_3		d_4		d_5	
	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T	$S*$	T
20	198,00	489,23	223,40	0,08	219,20	0,09	230,20	0,15	220,50	0,17	230,90	0,16
22	173,80	721,80	199,50	0,08	196,90	0,12	217,90	0,17	198,00	0,20	206,30	0,19
25	146,50	1144,53	170,90	0,08	170,40	0,11	206,60	0,28	173,00	0,22	178,10	0,15
30	115,00	1853,48	139,30	0,08	139,20	0,13	175,70	0,35	138,80	0,23	144,00	0,16

Apêndice B

Guia de instalação e uso do resolvedor Concorde

Guia de instalação e uso do resolvedor Concorde

Túlio Neme de Azevedo - tulioneme10@gmail.com

Marco Antonio Moreira de Carvalho - marco.opt@gmail.com

Departamento de Computação

Universidade Federal de Ouro Preto - MG

19 de outubro de 2017

Introdução

O *Concorde*[1] é um resolvedor para o clássico Problema do Caixeiro Viajante, do inglês *Traveling Salesman Problem* (TSP), e alguns problemas de otimização relacionados. Desenvolvido em ANSI C, o resolvedor possui os melhores resultados reportados até o momento, resolvendo todas as instâncias disponíveis na *TSPLIB*[2].

Este guia apresenta como instalar e utilizar o *Concorde* em um ambiente Linux, utilizando os resolvedores de programação linear QSOpt ou CPLEX. Todos os passos abaixo foram realizados em um computador com sistema operacional Ubuntu, porém, funcionam também para outras distribuições, incluindo Mac OS.

[1] http://www.math.uwaterloo.ca/tsp/concorde

[2] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

Instalação e compilação

Antes de tudo, verifique se sua máquina possui os resolvedores de programação linear QSopt ou CPLEX.

1. Qsopt: 64 bits[3] - 32 bits[4]
2. CPLEX[5] (é necessário fazer um cadastro no site da IBM e você escolhe a arquitetura)

A instalação requer passos diferentes dependendo do resolvedor de programação linear empregado, portanto foram dividas em duas subseções, uma para o QSopt e outra para o CPLEX.

[3] http://www.math.uwaterloo.ca/~bico/qsopt/beta/index.html

[4] http://www.math.uwaterloo.ca/~bico/qsopt/downloads/downloads.htm

[5] http://www-01.ibm.com/support/docview.wss?uid=swg24036489

QSopt

Após o *download* do QSopt, crie um link simbólico para o arquivo qsopt.a. No mesmo diretório dos arquivos do QSopt:

$ sudo ln -s qsopt.a libqsopt.a .

Faça o *download* do código-fonte do Concorde, disponível na seção *Download* do site oficial[6], (última versão disponibilizada em 19 de Dezembro de 2003).

O arquivo está compactado, e para descompactar utilize no terminal:

[6] http://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.htm

$ gunzip co031219.tgz

$ tar xvf co031219.tar

Em seguida selecione o diretório "concorde" criado e então configure:

$ cd concorde

$ ./configure –with-qsopt=DIR (diretório contendo "qsopt.h" e "qsopt.a")

Em seguida, execute o makefile do concorde:

$ make

Crie um link simbólico para a biblioteca do Concorde. No mesmo diretório do Concorde:

$ sudo ln -s concorde.a libconcorde.a .

Pronto, agora você pode resolver os problemas do TSP disponíveis na TSPLIB.

Se você quer modificar os parâmetros do resolvedor para resolver problemas *TSP-like*, então terá que corrigir alguns *bugs* eventuais gerados pelo Concorde.

Os desenvolvedores do Concorde adicionaram em algumas funções os parâmetros com identificadores "new" e "class", que em C++ são palavras reservadas, portanto, isso pode gerar erros de compilação.

Para resolver esse erro, basta editar o arquivo "concorde.h", na pasta de instalação do Concorde, e modificar (ou mesmo excluir) os identificadores "new" e "class".

Note que somente apagar as palavras resolve o problema, já que em C++ os *headers* não precisam ter o identificador dos parâmetros, apenas o tipo.

Obs: Tome cuidado para não fazer uma troca automática em todo o código e apagar mais do que deveria.

Para compilar um código que utilize o Concorde, é necessário ajustar alguns parâmetros. Sugiro um arquivo *makefile*, que irá facilitar a sua vida (DIRCONCORDE é o diretório da instalação do Concorde e DIRQSOPT é o diretório que contém os arquivos do QSOPT):

```
CXX= g++

CONCORDE=DIRCONCORDE

QSOPT=DIRQSOPT

CPPFLAGS= -w -m64

LIBFLAGS= -I$(QSOPT) -I$(CONCORDE) -L$(CONCORDE) -L$(QSOPT)
    -lconcorde -lqsopt -lm -lpthread

all:
```

```
    $(CXX) main.cpp -o teste $(LIBFLAGS) $(CPPFLAGS)

    @echo ''-- DONE --''


run:
    ./bin/main
```

Execute o *Makefile* (via comando *make*, no diretório em que se encontrar o *Makefile*) e o seu executável será gerado.

Pronto, tudo ok. SHOW TIME

CPLEX

Após o *download* do CPLEX, faça o *download* do código-fonte do Concorde, disponível na seção *Download* so site oficial do Concorde[7], (última versão disponibilizada em 19 de Dezembro de 2003).

[7] http://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.htm

O arquivo está compactado, e para descompactar utilize no terminal:

$ gunzip co031219.tgz

$ tar xvf co031219.tar

No diretório do Concorde, edite o arquivo "Makefile.in". Adicione -lpthread ao final da linha que define as "LIBFLAGS".

No mesmo diretório, selecione o diretório "TSP" e novamente edite o arquivo "Makefile.in" da mesma maneira que o anterior.

Usando o terminal, crie links simbólicos para os *headers* do CPLEX. No diretório de instalação do Concorde:

$ sudo ln -s /opt/ibm/ILOG/CPLEX_Studio1263/cplex/include/ilcplex/*.h .

$sudo ln -s /opt/ibm/ILOG/CPLEX_Studio1263/cplex/lib/x86-64_linux/static_pic/libcplex.a .

Após isto, no diretório "LP", edite o arquivo "lpcplex8.c", incluindo a linha #define CPX_PARAM_FASTMIP 1017. Esta constante era utilizada em versões anteriores do CPLEX, porém, foi removida das versões mais recentes.

Agora, configure o Concorde com o CPLEX, no diretório do Concorde:

$ ./configure –prefix=DIR (diretório do Concorde) –with-cplex=DIR (diretório do Concorde, que agora contém os links para os *headers* do CPLEX)

Execute o Makefile:

$ ./make

Crie um link simbólico para a biblioteca do Concorde. No mesmo diretório do Concorde:

$ sudo ln -s concorde.a libconcorde.a .

Obs: em meus testes esse passo não foi concluido com sucesso, tendo sido reportado um erro no arquivo criado. Caso ocorra, exclua o arquivo "libconcorde.a" criado, faça uma cópia do arquivo "concorde.a" e o renomeie para "libconcorde.a".

Os desenvolvedores do Concorde adicionaram em algumas funções os parâmetros com identificadores "new" e "class", que em C++ são palavras reservadas, portanto, isso pode gerar erros de compilação.

Para resolver esse erro, basta editar o arquivo "concorde.h", na pasta de instalação do Concorde, e modificar (ou mesmo excluir) os identificadores "new" e "class".

Note que somente apagar as palavras resolve o problema, já que em C++ os *headers* não precisam ter o identificador dos parâmetros, apenas o tipo.

Obs: Tome cuidado para não fazer uma troca automática em todo o código e apagar mais do que deveria.

Para compilar um código que utilize o Concorde, é necessário ajustar alguns parâmetros. Sugiro um arquivo *makefile*, que irá facilitar a sua vida (DIRCONCORDE é o diretório da instalação do Concorde, com os links simbólicos para os *headers* do CPLEX):

```
CXX= g++

CONCORDE=DIRCONCORDE

CPPFLAGS= -w -m64

LIBFLAGS= -I$(CONCORDE) -L$(CONCORDE) -lconcorde -lcplex -lm
    -lpthread

all:
    $(CXX) main.cpp -o teste $(LIBFLAGS) $(CPPFLAGS)

    @echo ''-- DONE --''


run:
    ./bin/main
```

Execute o *Makefile* (via comando *make*, no diretório em que se encontrar o *Makefile*) e o seu executável será gerado.

Pronto, tudo ok. SHOW TIME

Utilização

Para esse guia consideraremos a linguagem C++, utilizando as estruturas presentes na STL.

Como a abordagem do resolvedor é para problemas *TSP-like*, é necessário realizar a modelagem correta do seu problema, considerando um grafo com n vértices.

Para tal se faz necessário a criação de uma matriz de distâncias entre dois vértices do seu problema, desta forma temos uma matriz de tamanho $n \times n$. Além disso, um vetor, de tamanho n, é necessário. Desta forma o resolvedor preencherá o vetor criado com a melhor rota entre os vértices que componham o seu problema.

Crie uma função que contenha como parâmetro a matriz de distâncias e o vetor, sendo o segundo o resultado que o Concorde retorna. Nesta função realizaremos a chamada da função do Concorde que transforma a matriz de distância na estrutura de dados aceita pelo resolvedor, denominada *CCutil_graph2dat_matrix* e em seguida faremos a chamada da função que resolve o problema, chamada *CCtsp_solve_dat*, retornando um inteiro, que possui alguns parâmetros que precisam ser ajustados:

Obs: Os parâmetros estão nomeados de acordo com as descrições[8] [9] relatadas no site do Concorde.

Descrição da função *CCutil_graph2dat_matrix*:

int ncont número total de vértices (n);

int ecount número total de arestas;

*int *elist* *array* indicando o inicio e o fim dos pesos das arestas (em pares);

*int *elen* *array* indicando o peso das arestas;

int defaultelen parametro *default* para o peso das arestas;

*CCdatagroup *dat* estrutura de dados especifica do concorde;

Descrição da função *CCtsp_solve_dat*:

int ncont número total de vértices (n);

*CCdatagroup *dat* estrutura de dados especifica do concorde (preenchida por uma função);

*int *in_tour* fornece uma solução inicial (pode ser NULL);

*int *out_tour* vetor contendo a sequência ótima dos vértices (pode ser NULL);

*double *in_val* define um limite superior inicial para o valor da solução (pode ser NULL);

*double *optval* valor ótimo para a solução do Caixeiro Viajante;

[8] http://www.math.uwaterloo.ca/tsp/concorde/DOC/util.html#CCutil_graph2dat_matrix

[9] http://www.math.uwaterloo.ca/tsp/concorde/DOC/tsp.html#CCtsp_solve_dat

*int *success* retorna 1 caso a função tenha executada corretamente, ou 0 se a execução foi interrompida precocemente (de acordo com alguns limites pre estabelecidos do Concorde);

*int *foundtour* retorna 1 se uma solução foi encontrada (caso o parâmetro *success* for 0, então o caminho pode não ser o ótimo);

*char *name* nomeação dos arquivos que são escritos durante a execução do algoritmo (caso seja NULL, por padrão, "noname" será utilizado, porém, em programas *multithreaded* isto gera problemas, portanto prefira especificar algo);

*double *timebound* limite superior para o tempo de execução;

*int *hit_timebound* retorna 1 se o *timebound* for atingido, ou 0 caso contrário (pode ser nulo);

int silent caso diferente de 0, boa parte da saída será suprimida;

*CCrandstate *rstate* é utilizado pelo gerador de números aleatórios do Concorde;

Obs: Para realizar a chamada do resolvedor é necessária a inclusão da biblioteca "concorde.h".

Faremos um exemplo mínimo funcional:

```
extern "C" { #include <concorde.h> }

//criando funcao para a chamada da funcao presente no Concorde
void solving_tsp_concorde(int[][] distancia, int[] tour){

   //criando uma sequencia qualquer no vetor
   for(int i = 0; i < tour->size(); i++){
      tour->at(i) = i;
   }
   if(dist->size() > 4 ){//TSP somente para mais de 4 elementos
      int rval = 0;
      int semente = rand();
      double szeit, optval, *in_val, *timebound;
      int ncount, success, foundtour, hit_timebound = 0;
      int *in_tour = (int *) NULL;
      int *out_tour = (int *) NULL;
      CCrandstate rstate;
      char *name = (char *) NULL;
      static int silent = 1;
      CCutil_sprand(semente, &rstate);
      in_val = (double *) NULL;
      timebound = (double *) NULL;
      ncount = dist->size();
      int ecount = (ncount * (ncount - 1)) / 2;
      int *elist = new int[ecount * 2];
```

```
      int *elen = new int[ecount];
      int edge = 0;
      int edge_peso = 0;
      for (int i = 0; i < ncount; i++) {
         for (int j = i + 1; j < ncount; j++) {
            if (i != j) {
               elist[edge] = i;
               elist[edge + 1] = j;
               elen[edge_peso] = dist->at(i)[j];
               edge_peso++;
               edge = edge + 2;
            }
         }
      }
      out_tour = CC_SAFE_MALLOC (ncount, int);
      name = CCtsp_problabel(" ");

      CCdatagroup dat;
    CCutil_init_datagroup (&dat);

    rval = CCutil_graph2dat_matrix (ncount, ecount, elist, elen,
        1, &dat);

    rval = CCtsp_solve_dat (ncount, &dat, in_tour, out_tour, NULL,
        &optval, &success, &foundtour, name, timebound,
        &hit_timebound, silent, &rstate);

      for (int i = 0; i < ncount; i++) {
         tour->at(i) = out_tour[i];
      }
      szeit = CCutil_zeit();
      CC_IFFREE (elist, int);
      CC_IFFREE (elen, int);
      CC_IFFREE (out_tour, int);
      CC_IFFREE (probname, char);
   }

}

int main(){

   //inicializacao da matriz e vetor
   solving_tsp_concorde(distancia,tour);

   //verificando resultado dado pelo resolvedor
   for(int i = 0; i < tour->size(); i++){
      cout<<tour->at(i)<<" ";
   }

   //faca aqui a construcao da solucao do seu problema
```

```
   return 0;
}
```

Modele seu problema de acordo com passos deste *HowTo* e resolva seu problema *TSP-like* de forma exata.

Referências

Algumas referências foram utilizadas para criarmos este *HowTo*:

http://rodrigobrito.net/2016/08/14/instalando-concorde-tsp-solver/
http://www.leandro-coelho.com/installing-concorde-tsp-with-cplex-linux/#more-146
http://www.math.uwaterloo.ca/tsp/concorde

Referências Bibliográficas

Adjiashvili, D.; Bosio, S. e Zemmer, K. (2015). Minimizing the number of switch instances on a flexible machine in polynomial time. *Operations Research Letters*, 43(3):317–322.

Al-Fawzan, M. A. e Al-Sultan, K. S. (2003). A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. *Computers & industrial engineering*, 44(1):35–47.

Amaya, J. E.; Cotta, C. e Fernández, A. J. (2008). A memetic algorithm for the tool switching problem. In *Hybrid metaheuristics*, pp. 190–202. Springer.

Amaya, J. E.; Cotta, C. e Fernández-Leiva, A. J. (2012). Solving the tool switching problem with memetic algorithms. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 26(02):221–235.

Amaya, J. E.; Cotta, C. e Fernández-Leiva, A. J. (2013). Cross entropy-based memetic algorithms: An application study over the tool switching problem. *International Journal of Computational Intelligence Systems*, 6(3):559–584.

Applegate, D.; Cook, W. e Rohe, A. (2003). Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92.

Bard, J. F. (1988). A heuristic for minimizing the number of tool switches on a flexible machine. *IIE transactions*, 20(4):382–391.

Boruvka, O. (1926). O jistem problemu minimaalnim. moravske prirodovedecke spolecnosti 3, 37-58.

Catanzaro, D.; Gouveia, L. e Labbé, M. (2015). Improved integer linear programming formulations for the job sequencing and tool switching problem. *European Journal of Operational Research*, 244(3):766–777.

Chaves, A.; Lorena, L.; Senne, E. e Resende, M. (2016). Hybrid method with cs and brkga applied to the minimization of tool switches problem. *Computers & Operations Research*, 67:174–183.

Chaves, A. A.; Senne, E. L. F. e Yanasse, H. H. (2012). A new heuristic for the minimization of tool switches problem. *Gestão & Produção*, 19(1):17–30.

Cook, W. (2012). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.

Crama, Y.; Kolen, A. W. J.; Oerlemans, A. G. e Spieksma, F. C. R. (1994). Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6(1):33–54.

Crama, Y.; Moonen, L. S.; Spieksma, F. C. e Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, 182(2):952–957.

Djellab, H.; Djellab, K. e Gourgand, M. (2000). A new heuristic based on a hypergraph representation for the tool switching problem. *International Journal of Production Economics*, 64(1):165–176.

Fathi, Y. e Barnette, K. (2002). Heuristic procedures for the parallel machine problem with tool switches. *International Journal of Production Research*, 40(1):151–164.

Gendreau, M.; Hertz, A. e Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

Golden, B. e Stewart, W. (1985). Empirical analysis of heuristics in the travelling salesman problem, e. lawer, j. lenstra, a. rinnooy and d. shmoys.

Hertz, A.; Laporte, G.; Mittaz, M. e Stecke, K. E. (1998). Heuristics for minimizing tool switches when scheduling part types on a flexible machine. *IIE transactions*, 30(8):689–694.

Kernighan, B. W. e Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247.

Laporte, G.; Salazar-Gonzalez, J. J. e Semet, F. (2004). Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, 36(1):37–45.

Linhares, A. e Yanasse, H. H. (2002). Connections between cutting-pattern sequencing, {VLSI} design, and flexible machines. *Computers & Operations Research*, 29(12):1759 – 1772.

Paiva, G. S. e Carvalho, M. A. M. (2017). Improved heuristic algorithms for the job sequencing and tool switching problem. *Comput. Oper. Res.*, 88(C):208–219.

Privault, C. e Finke, G. (2000). k-server problems with bulk requests: an application to tool switching in manufacturing. *Annals of Operations Research*, 96(1-4):255–269.

Raduly-Baka, C. e Nevalainen, O. S. (2015). The modular tool switching problem. *European Journal of Operational Research*, 242(1):100–106.

Senne, E. L. F. e Yanasse, H. H. (2009). Beam search algorithms for minimizing tool switches on a flexible manufacturing system. In *XI WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering, MACMESE*, volume 9, pp. 68–72.

Shirazi, R. e Frizelle, G. (2001). Minimizing the number of tool switches on a flexible machine: an empirical study. *International Journal of Production Research*, 39(15):3547–3560.

Tang, C. S. e Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part i: minimization of the number of tool switches. *Operations research*, 36(5):767–777.

Tzur, M. e Altman, A. (2004). Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes. *IIE Transactions*, 36(2):95–110.

Yanasse, H. e Lamosa, M. (2005). An application of the generalized travelling salesman problem: the minimization of tool switches problem. In *International Annual Scientific Conference of the German Operations Research Society*, p. 90.

Yanasse, H. e Lamosa, M. (2006). On solving the minimization of tool switches problem using graphs. In *Annals of the XII ICIEOM – International Conference on Industrial Engineering and Operations Management*.

Yanasse, H. H. e Pinto, M. J. (2002). The minimization of tool switches problem as a network flow problem with side constraints. *XXXIV SBPO–Simpósio Brasileiro de Pesquisa Operacional, realizado no Rio de Janeiro, RJ*, 8:86.

Yanasse, H. H.; Rodrigues, R. d. C. M. e Senne, E. L. F. (2009). Um algoritmo enumerativo baseado em ordenamento parcial para resolução do problema de minimização de trocas de ferramentas. *Gestão and Produção*, 13(3).

Zhou, B.-H.; Xi, L.-F. e Cao, Y.-S. (2005). A beam-search-based algorithm for the tool switching problem on a flexible machine. *The International Journal of Advanced Manufacturing Technology*, 25(9-10):876–882.