GUSTAVO SILVA PAIVA

Orientador: Marco Antonio Moreira de Carvalho

UM MÉTODO PARA PLANEJAMENTO DE PRODUÇÃO EM SISTEMAS DE MANUFATURA FLEXÍVEL

Ouro Preto

Agosto de 2016

Universidade Federal de Ouro Preto
Instituto de Ciï£¡ncias Exatas
Bacharelado em Ciência da Computação

UM MÉTODO PARA PLANEJAMENTO DE PRODUÇÃO EM SISTEMAS DE MANUFATURA FLEXÍVEL

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.

GUSTAVO SILVA PAIVA

Ouro Preto
Agosto de 2016

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Um Método Para Planejamento de Produção em Sistemas de Manufatura Flexível

GUSTAVO SILVA PAIVA

Monografia defendida e aprovada pela banca examinadora constituída por:

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

Dr. Puca Huachi Vaz Penna
Universidade Federal de Ouro Preto

Dr. Joubert de Castro Lima
Universidade Federal de Ouro Preto

Ouro Preto, Agosto de 2016

Resumo

Em sistemas de manufatura flexível, uma mesma máquina pode ser configurada com diferentes ferramentas para processar diferentes tarefas, cada uma exigindo um conjunto específico de ferramentas. Há um limite para o número máximo de ferramentas instaladas simultaneamente na máquina, e entre o processamento de duas tarefas diferentes pode ser necessária a troca destas ferramentas, implicando na interrupção da produção para este fim. O problema de Minimização de Trocas de Ferramentas visa sequenciar o processamento das tarefas no intuito de minimizar estas trocas. Neste trabalho são propostos uma nova representação em grafos para o problema, uma nova heurística e um novo método de busca local. Estes métodos foram combinados em uma busca local iterada e comparados ao estado da arte em relação ao problema tratado. Experimentos computacionais extensos demonstram que o método proposto é competitivo e obteve novas melhores soluções para instâncias da literatura, superando o estado da arte atual.

Abstract

In flexible manufacturing systems, a single machine can be configured with different tools for processing different tasks, each requiring a specific set of tools. There is a limit to the maximum number of tools installed simultaneously on the machine and between the processing of two different tasks it may be necessary to switch these tools, causing the interruption of the production line. The Minimization of Tool Switches problem aims to sequence the processing of tasks in order to minimize these switches. This paper proposes a new representation in graphs, a new heuristic and a new local search method. These methods were combined in an iterated local search and compared to state of the art related to the problem. Extensive computational experiments show that the proposed method is competitive and obtained new best solutions for literature instances, surpassing the current state of the art.

Aos meus pais, Hélio e Eunice, pelo amor incondicional e por alimentar o meu eu acadêmico.

Agradecimentos

Agradeço a minha amiga de vida e irmã, Isabela, pelo carinho, risadas e amparo em todos os momentos.

A minha companheira, Carolina, que tive a sorte de conhecer durante minha graduação. Agradeço pela afeição e pelos momentos mais felizes durante essa etapa da minha vida e, também, pela compreensão nos feriados e fins de semanas de trabalho e, principalmente, pelo apoio e incentivo desde o início.

Agradeço à Universidade Federal de Ouro Preto, lugar onde tive a honra de estudar, aprender e amadurecer intelectualmente e pessoalmente.

Agradeço ao meu orientador, Marco Antonio, por todo aprendizado passado e por ser esse professor excepcional que admiro.

Agradeço à todos os professores, principalmente aos professores Guilherme Assis, Joubert Lima, Saul Delabrida e Tiago Senna, pelo maior bem que um homem pode obter, o conhecimento.

Agradeço a todos os meus amigos de curso que percorreram essa árdua caminhada ao meu lado pela troca de conhecimento e estudos em conjunto.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Atualmente, com a concorrência cada vez mais acirrada as empresas se vêem com a necessidade de flexibilizar, ainda mais, a sua produção visando otimizar a utilização dos recursos. Para isto o sistema de manufatura flexível (*Flexible Manufacturing System* – FMS) está sendo, comumente, adotado em muitas empresas que possuem uma ampla matriz de produtos. Este sistema é caracterizado por permitir uma maior flexibilidade no planejamento da produção,por exemplo, ao permitir uma rápida adequação à produção de um conjunto novo de produtos utilizando o maquinário já existente na linha de produção e também ao permitir uma rápida readequação da linha de produção frente a imprevistos.

Um tipo muito comum de FMS, principalmente em empresas metalúrgicas, é aquele que utiliza máquinas flexíveis. Uma máquina flexível é definida pela sua capacidade de efetuar diferentes tipos de operações sem que haja uma brusca troca entre uma operação e outra, tornando a produção mais dinâmica. Além disto, este tipo de máquina possui um compartimento de capacidade fixa em que *ferramentas* são carregadas. Cada produto requer que um conjunto de ferramentas (e.g., lâminas de corte, brocas de perfuração, etc) específico seja carregado na máquina flexível para sua produção. O compartimento de ferramentas, geralmente é suficiente para armazenar todas as ferramentas necessárias para fabricação dos produtos isoladamente, porém, não é suficiente para armazenar todas as ferramentas existentes simultaneamente.

A utilização do FMS fornece um grande mecanismo para diversificação da produção, entretanto, a operacionalização de um FMS é também influenciada por outros fatores, principalmente, o planejamento da produção e o escalonamento das ferramentas em máquinas flexíveis. Com a diversificação concedida pelo FMS é possível a manufatura de diferentes tipos de produtos, que por sua vez podem requerer diferentes tipos de ferramentas. Como o compartimento de ferramentas é limitado e os diversos produtos podem requerer diferentes conjuntos de ferramentas, ao se produzir diferentes produtos em sequência, eventualmente serão necessárias trocas de ferramentas para dar continuidade à produção. Estas trocas de ferramentas implicam na interrupção da linha de produção, pois a máquina deverá ser desligada para que receba a nova configuração de ferramentas. A interrupção da linha de produção

aumenta o custo da produção e é desejável, portanto, que seja realizada o menor número de vezes possível.

A partir de uma demanda por produtos, predeterminada, é necessário a criação de um plano de produção para que uma máquina cumpra esta demanda. Este plano é dividido em *tarefas* e tem como objetivo a minimização do tempo ocioso da máquina de produção, de forma a maximizar a produtividade e diminuir os custos relacionados.

O plano de produção consiste em:

1. Determinar a ordem em que as tarefas serão executadas; e
2. Decidir quando realizar cada troca de ferramentas e quais ferramentas serão trocadas, de maneira a viabilizar a produção.

O *Problema de Minimização de Trocas de Ferramentas* é definido como o problema de determinar o melhor plano de produção possível, gerando a sequência em que as tarefas devem ser executadas de forma a minimizar o número de trocas de ferramentas necessário durante o processo de produção.

Existem diferentes versões do MTSP apresentadas na literatura. O caso geral do MTSP considera que os tamanhos das ferramentas são uniformes (logo a localização das mesmas no compartimento é irrelevante) e que os custos de troca de ferramentas também são uniformes (isto é, o custo para trocas de ferramentas é o mesmo). Já na versão do MTSP que considera ferramentas de tamanho não uniformes, a localização destas ferramentas é relevante, pois uma ferramenta pode ocupar um espaço proporcional a duas ou mais ferramentas no compartimento, restringindo a utilização do mesmo. Versões mais recentes do problema consideram um ambiente dinâmico (*online*) para MTSP, no qual não se sabe *a priori* todas as tarefas que deverão ser realizadas: as demandas por produtos são conhecidas somente após o término da produção.

Como demonstrado por (Tang e Denardo, 1988), o MTSP pode ser classificado em dois casos:

1. O problema de carregamento das ferramentas, no qual deve ser determinado o número mínimo de trocas de ferramentas, a partir de uma sequência fixa de tarefas;
2. O problema de sequenciamento de tarefas, no qual deve ser determinado a sequência que possui o menor número de trocas de ferramentas.

Trataremos o problema de sequenciamento de tarefas como sendo o problema principal do MTSP, pois o problema de carregamento é simples e pode ser resolvido em tempo determinístico polinomial pelo algoritmo KTNS (*Keep Tool Needed Soonest*), vide (Tang e Denardo, 1988).

1.1 Motivação

Através de estudos, foi constatado que empresas que utilizam FMS possuem planos de produção que podem ser melhorados, logo, trabalhos realizados sobre o MTSP possuem grandes aplicabilidades práticas, desde metalúrgicas até fabricantes de aeronaves. O MTSP foi caracterizado NP-Difícil por (Crama et al., 1994).

Na eletrônica é possível observar que a utilização de métodos de montagem de placas de circuitos impressos (*Printed Circuit Board* – PCB) pode caracterizar um caso do MTSP. Um dos métodos mais utilizados é o método de Tecnologia de Montagem Superficial (*Surface Mount Technology* – SMT). O SMT possibilita a utilização de sistemas de manufatura mais robustos para a montagem de PCBs, como por exemplo o FMS. Neste cenário, deseja-se montar PCBs inserindo componentes eletrônicos nos mesmos. É possível fazer uma analogia da montagem dos PCBs como sendo as tarefas que devem ser realizadas e os componentes eletrônicos como sendo as ferramentas necessárias para execução das tarefas.

Uma máquina flexível seria capaz de executar métodos de montagem de PCBs da seguinte forma: ao invés de ferramentas esta máquina comportaria alimentadores de diferentes componentes eletrônicos, assim sendo possível a fixação destes em um PCB. Logo, o MTSP pode ser aplicado também para otimizar a montagem de PCBs, demostrando uma das aplicações industriais deste problema.

1.2 Objetivos

Este trabalho tem como objetivo geral propor uma heurística robusta que possa ser utilizada para resolução do problema de minimização de trocas de ferramentas, permitindo que seja alcançada soluções próximas ao ótimo com um baixo custo computacional. Pretende-se, também, comparar esta heurística com outras presentes na literatura de forma a qualificar os resultados obtidos e contribuir para a literatura do problema.

Os objetivos específicos deste trabalho são apresentados a seguir:

1. Propor modificações na representação do problema utilizando conceitos da Teoria dos Grafos;

2. Propor uma heurística para geração de soluções inicias e um método de busca local;

3. Implementar um ferramenta de resolução do MTSP composto pelos métodos propostos;

4. Avaliar os resultados obtidos pela ferramenta implmentada considerando dados disponibilizados publicamente na literatura e comparando-o com o estado da arte.

1.3 Organização do Trabalho

O restante deste trabalho está organizado da seguinte maneira: O Capítulo 2 relata os trabalhos apresentados na literatura. Os fundamentos teóricos do MTSP é apresentado no Capítulo 3. O Capítulo 4 detalha as contribuições deste trabalho. O conjunto dos métodos propostos é comparado com os melhores resultados da literatura e também com o estado da arte utilizando instâncias da literatura no Capítulo 5. Por fim, as conclusões e propostas de trabalhos futuros são apresentadas no Capítulo 6.

Capítulo 2

Revisão da Literatura

Tang e Denardo (1988) apresentaram uma política ótima que minimiza o número total de trocas de ferramentas dada uma sequência de tarefas fixa, denominada KTNS (*Keep Tool Needed Soonest*). Caso sejam necessárias trocas de ferramentas, essa política será responsável por manter na máquina, as ferramentas que serão utilizadas mais cedo dentre as próximas tarefas da sequência. Mais tarde no mesmo ano, Bard (1988) propôs uma formulação não linear e inteira e também uma heurística baseada em relaxação lagrangiana.

Crama et al. (1994) apresentaram uma prova formal de que o problema é NP-Difícil quando $C \geq 2$. No mesmo trabalho, foi proposta uma heurística gulosa denominada(*Multiple Start Greedy*) e uma formulação do MTSP baseada na formulação do Problema do Caixeiro Viajante (PCV). Usando a mesma formulação do PCV, Hertz et al. (1998) propuseram diferentes métricas para calcular os pesos das arestas e adaptaram heurísticas desenvolvidas originalmente para o problema do caixeiro viajante para o MTSP.

Shirazi e Frizelle (2001) testaram heurísticas da literatura em instâncias reais de indústrias. Foi possível perceber que as instâncias reais são mais fáceis do que as instâncias geradas pela literatura, pela existência de ferramentas que são utilizadas por grande parte da linha de produção e as máquinas, geralmente, possuem um compartimento de ferramenta com capacidade superior do que as necessárias.

Al-Fawzan e Al-Sultan (2003) desenvolveram uma busca tabu com memórias de curto e de longo prazo e com a utilização de mecanismos estratégicos e probabilísticos para a análise do espaço de soluções.

Laporte et al. (2004) propuseram um modelo de programação linear inteira. Para resolver este modelo, foi implementado um algoritmo *branch-and-cut*. Foi também apresentado um esquema *branch-and-bound* que consegue resolver instâncias pequenas, com até 25 tarefas e 25 ferramentas.

Yanasse et al. (2009) propôs um algoritmo *branch-and-bound*, e também introduziu um conjunto de instâncias novo. Apesar de mostrar resultados mais satisfatórios do que o método de Laporte et al. (2004) para instâncias onde o mesmo falhava, esse novo algoritmo não foi

capaz de resolver outros casos e não foi possível solucionar instâncias com 25 tarefas e 15, 20 e 25 ferramentas.

Recentemente, Catanzaro et al. (2015) apresentaram uma revisão dos modelos de programação inteira para o MTSP destacando características de cada modelo no intuito de melhorá-los. Foram então desenvolvidos dois novos modelos que possuem um relaxamento linear melhor do que todos os anteriores.

Chaves et al. (2016) abordaram o MTSP utilizando a metaheurística *Clustering Search*, que tem como ideia encontrar *clusters* promissores, regiões do espaço de soluções com chances maiores de obter boas soluções, para que estes sejam intensivamente examinados pela busca local *Variable Neighborhood Descent*. Para a fase de busca de *clusters* foi implementado o recente algoritmo genetico *Biased Random Key Genetic Algorithm*. Este método foi denominado *CS+BRKGA* e seus resultados são promissores, melhorando os resultados obtidos anteriormente pelo ILS proposto por Chaves et al. (2012), considerando o conjunto de instâncias proposto em Yanasse et al. (2009) e Crama et al. (1994).

Algumas extensões do MTSP tratam a existência de ferramentas de tamanho não uniforme (Matzliach e Tzur, 1998; Crama e Talloen, 2007; Marvizadeh e Choobineh, 2013), máquinas paralelas (Fathi e Barnette, 2002), módulos de ferramentas para troca de múltiplas ferramentas (Raduly-Baka e Nevalainen, 2015), além de novas funções objetivos como a minimização de paradas da máquina flexível e a diminuição da reinserção de ferramentas em diferentes espaços do compartimento da máquina flexível (Adjiashvili et al., 2015).

Capítulo 3

Fundamentação Teórica

Neste capítulo é apresentada a fundamentação teórica sobre o Minimização de Troca de Ferramentas e as técnicas utilizadas durante este trabalho.

3.1 Problema de Minimização de Troca de Ferramentas

Uma instância do MTSP é composta pelo conjunto de tarefas que devem ser realizadas $T = \{1, \ldots, n\}$, o conjunto de ferramentas disponíveis $F = \{1, \ldots, m\}$, o conjunto de ferramentas T_j necessárias para executar a tarefa $j \in T$ e a capacidade C do compartimento de ferramentas da máquina. Considera-se que toda tarefa $j \in T$ tem, no máximo, C ferramentas em seu subconjunto T_j, isto é, $|T_j| \leq C$. Uma solução do MTSP é representada pela permutação ϕ do conjunto T e também de um plano de trocas de ferramentas.

O plano de trocas de ferramentas é representado por uma matriz binária $A^{\phi}_{m,n}$ na qual as colunas obedecem a ordem das colunas em ϕ e cada entrada $a^{\phi}_{i,j} = 1$ caso a ferramenta i esteja na máquina durante a execução da tarefa j, ou $a^{\phi}_{i,j} = 0$ caso contrário. Adicionalmente, para toda tarefa $j \in T$ temos que $a^{\phi}_{i,j} = 1$ para toda ferramenta $i \in T_j$. Logo, o número de trocas de ferramentas corresponde ao número de inversões em que $a^{\phi}_{i,j-1} = 0$ e $a^{\phi}_{i,j} = 1$ ($\forall i \in F,\ j \in \{2, \ldots n\}$), indicando que a ferramenta i não estava carregada na máquina durante a execução de uma tarefa $j-1$, porém, foi carregada antes da execução da tarefa j. O carregamento das ferramentas iniciais também são contabilizadas como trocas de ferramentas. Considere que a matriz A^{ϕ} possui uma coluna adicional 0 indicando o estado inicial da máquina no qual nenhuma ferramenta foi carregada, i.e., $a^{\phi}_{i,0} = 0$ para toda ferramenta i. A Equação 3.1 apresenta a função objetivo do MTSP que será minimizada:

$$Z_{MTSP}(A^{\phi}) = \sum_{i=1}^{m} \sum_{j=1}^{n} a^{\phi}_{i,j}(1 - a^{\phi}_{i,j-1}) \tag{3.1}$$

Crama et al. (1994) definiram um padrão existente em matrizes binárias do MTSP, o *1-block*: um conjunto de entradas consecutivas em uma mesma linha da matriz com o valor igual

a 1. Nota-se que cada *1-block* caracteriza uma troca de ferramentas, visto que este indicam o intervalo na qual cada ferramenta (uma linha da matriz) permanece carregada na máquina. Assim, pode-se utilizar desta definição para calcular a função objetivo do MTSP, i.e., o número de trocas de ferramentas de uma sequência ϕ é equivalente ao número de *1-blocks* na matriz A^{ϕ}.

A Tabela 3.1 apresenta um exemplo numérico de uma instância do MTSP. A primeira linha representa as tarefas (enumeradas de 1 a 5) e as quatro próximas linhas apresentam as ferramentas necessárias para realizar a tarefa da respectiva coluna. Por fim a última linha apresenta a capacidade do compartimento de ferramentas da máquina.

Tabela 3.1: Exemplo de uma instância do MTSP.

Tarefas	1	2	3	4	5
Ferramentas	1	1	3	2	1
	2	3	4	3	4
	4		5	5	6
Capacidade do compartimento = 3					

A Tabela 3.2 apresenta duas possíveis soluções para a instância da Tabela 3.1. Em (a) a matriz A^{ϕ} se refere à solução $\phi = [1, 2, 3, 4, 5]$. A ferramenta 4, sublinhada, não é necessária para execução da tarefa 2, mas foi mantida no compartimento para evitar futuras trocas desnecessárias. A solução exige 9 trocas de ferramentas, sendo três para inserção das ferramentas iniciais (ferramentas 1, 2 e 4); uma troca para execução da tarefa 2 (ferramenta 2 pela ferramenta 3); uma troca para execução da tarefa 3 (ferramenta 1 pela ferramenta 5); uma troca na para execução da tarefa 4 (ferramenta 4 pela ferramenta 2) e três trocas para execução da tarefa 5 (ferramentas 2, 3 e 5 pelas ferramenta 1, 4 e 6, respectivamente). A Tabela 3.2(b), apresenta uma segunda solução, referente a $\phi = [3, 4, 1, 5, 2]$, que resulta em 8 trocas de ferramentas.

Tabela 3.2: Exemplo de representação em matriz binária e possível solução.

ϕ	1	2	3	4	5
	1	1	0	0	1
	1	0	0	1	0
	0	1	1	1	0
	1	<u>1</u>	1	0	1
	0	0	1	1	0
	0	0	0	0	1

(a)

ϕ	3	4	1	5	2
	0	0	1	1	1
	0	1	1	0	0
	1	1	0	0	1
	1	0	1	1	<u>1</u>
	1	1	0	0	0
	0	0	0	1	0

(b)

3.2 Busca Local Iterada

Cada problema de otimização combinatória possui um conjunto S de possíveis soluções, também chamado de *espaço de busca*. Dada uma solução específica, esta pode ser alterada pela aplicação de operações denominadas *movimentos*. As soluções em S que podem ser obtidas a partir de uma solução em particular pela aplicação de movimentos define uma *vizinhança*. Cada solução obtida é chamada de *vizinha* da solução inicial. Diferentes movimentos definem diferentes vizinhanças que representam diferentes maneiras de explorar o espaço de busca com a finalidade de determinar a melhor solução possível para o problema tratado.

Métodos de *busca local* são heurísticas que buscam, a partir de uma solução inicial, soluções melhores até que um *ótimo local* seja alcançado. Ótimo local é uma solução que dada uma estrutura específica de vizinhança, não ha solução vizinha de melhor valor. Com base nesta estrutura de vizinhança, é realizada uma busca entre os vizinhos da solução atual e, com base em um critério definido pela busca local (e.g. primeiro vizinho de melhora, vizinho com maior melhora, etc.), um vizinho é escolhido para se tornar a novo solução atual. Este procedimento é executado até que se atinja a melhor solução possível.

Baseando-se no fato de que as buscas locais, geralmente, produzem melhores resultados quando aplicadas sucessivamente, Lourenço et al. (2003) propôs a metaheurística *Busca Local Iterada* (ou *Iterated Local Search* – ILS). Este método consiste em aplicar iteradamente métodos de busca local em uma série de soluções com sutis, porém relevantes, *perturbações* (i.e., alterações aleatórias na composição da solução) das mesmas, de modo a eventualmente alcançar um novo vizinho que consiga levar à soluções que gerem melhoras na solução global. Esta idéia simples vem se mostrando efetiva na resolução de difíceis problemas de otimização combinatória.

A qualidade das soluções obtidas pela utilização de iterações de busca local é limitada pelos ótimos locais da região do espaço de busca explorado. Para contornar esta característica, é necessário uma maneira de permitir que a busca local explore uma parte significativa do espaço de busca S. Para isso são utilizados procedimentos de perturbação, que servem como uma maneira deb deslocar a exploração para outra região do espaço de busca. Estes procedimentos tentam manter uma parte significativa das características da solução obtida pela iteração anterior, com a adição de um fator de variação para que o espaço de busca S seja melhor examinado.

A substituição, ou não, de uma solução atual por uma solução vizinha é determinada de acordo com um *critério de aceitação*. Geralmente, esses se relacionam com alterações da qualidade da solução atual: melhoria, não piora, ou mesmo, piora. Diferentes *critérios de parada* são utilizados para determinar quando uma busca local iterada deve ser suspensa. Estes variam entre número de iterações, número de iterações sem melhoria da solução, tempo computacional gasto entre outros.

No Algoritmo 3.1 é apresentado a arquitetura básica da ILS.

Algoritmo 3.1: Busca Local Iterada

```
s_0 = GeraSolucaoInicial;
s = BuscaLocal(s_0);
enquanto critério de parada não atendido faça
    s' = Perturbe(s);
    s'' = BuscaLocal(s');
    se s'' atender o critério de aceitação então
        s ← s'';
retorna s;
```

3.3 Método de Descida

Comumente, procedimentos de busca local para problemas de minimização consistem em *métodos de descida*. Estes métodos, a partir de uma solução inicial, definem uma vizinhança com base em um movimento específico e selecionam o melhor vizinho existente. Esta operação é repetida sucessivamente enquanto houver melhoria do valor da solução, ou seja, até que um ótimo local seja atingido.

Neste trabalho, consideramos o movimento de *troca* (ou *swap*) embutido em um método de descida. Desta forma, a partir de uma solução atual, realizam-se trocas de posição entre duas tarefas, desde que haja uma melhoria no valor da solução. A cada troca realizada, o processo é reiniciado. Ao atingir um ótimo local, o processo é interrompido. No Algoritmo 3.2 é apresentado um pseudo-código do método de descida.

Algoritmo 3.2: Método de Descida

```
φ = GeraSolucaoInicial;
enquanto critério de parada não atendido faça
    para cada troca entre duas tarefas de φ, gerando φ' faça
        se φ' atender o critério de aceitação então
            φ ← φ';
retorna φ;
```

Esta estratégia é amplamente utilizada como método de busca local, principalmente em problemas cujas soluções são representadas por meio de permutações, assim como o MTSP.

Adicionalmente, também utiliza-se o movimento de troca como um mecanismo de perturbação, em que é definido um número fixo de trocas a serem realizadas, escolhidas aleatoriamente. A maior diferença desta utilização reside em realizar as trocas independente da melhoria da solução.

Neste trabalho, o método de descida foi utilizado como método de busca local e a aplicação de movimentos de troca aleatórios como um método de perturbação.

Capítulo 4

Desenvolvimento

Neste trabalho propõem-se uma modificação da representação do MTSP por grafos, uma nova heurística e também um novo método de busca local. Ambos os métodos são combinados em uma estratégia de busca local iterada para solução do MTSP. As seções seguintes apresentam os detalhes. Na figura 4.1 é apresentada um exemplo deste Grafo.

4.1 Uma Modificação da Representação do MTSP por Grafos

Um *Grafo de Ferramentas* é definido como $G = (V, E)$, no qual V é o conjunto de vértices que representam as ferramentas e E é o conjunto das arestas $\{i, j\}$ que representam se uma ferramenta i é utilizada ao mesmo tempo da ferramenta j. Neste trabalho propõe-se que o peso de cada aresta $\{i, j\}$ represente a quantidade de vezes que a ferramenta i e a ferramenta j aparecem juntas em uma mesma tarefa. Salvo melhor juízo, não há registro de utilização desta estratégia para solução do MTSP.

4.2 Uma Heurística Baseada em Busca em Grafos

No intuito de determinar a sequência das tarefas ϕ, inicialmente identifica-se quais ferramentas tendem a ser utilizadas em conjunto com maior frequência. A partir de um vértice inicial executa-se uma Busca em Largura (ou *Breadth-First Search* – BFS) com a adição de um critério baseado no peso das arestas para definição da ordem de exploração dos demais vértices. Arestas de maior peso indicam ferramentas que são utilizadas mais vezes em pares na máquina, logo devem estar próximas no sequenciamento das ferramentas. Desta forma, o critério de maior grau guia a execução da BFS. Ao final da execução da BFS a ordem de exploração dos vértices F_ϕ será a sequência de ferramentas.

O Algoritmo 4.1 apresenta um pseudo-código da BFS com a modificação citada anteriormente.

Algoritmo 4.1: Sequenciamento de Ferramentas via BFS

Dados: Instância MTSP, grafo de ferramentas $G = (V, E)$, vértice inicial x
Crie uma sequência F_ϕ vazia;
Crie uma fila Q vazia;
Insira x em F_ϕ;
Insira x em Q;
enquanto *existir vértice em* V *não visitado* **faça**
  enquanto $Q \neq \emptyset$ **faça**
    $f \leftarrow$ remove elemento de Q;
    Marque f como visitado;
    $vizinhos \leftarrow$ ordene decrescentemente os vizinhos de f de acordo com o peso da aresta;
    para cada $v \in vizinhos$ **faça**
      se $v \notin F_\phi$ **então**
        Insira v em Q;
        Insira v ao final de F_ϕ;
  se *existir um vertice* $k \in V$ *ainda não visitado* **então**
    Insira k em Q;
retorna F_ϕ

Na Figura 4.1 é ilustrado o passo a passo da execução da BFS conforme descrita, usando o exemplo apresentado na Tabela 3.1.

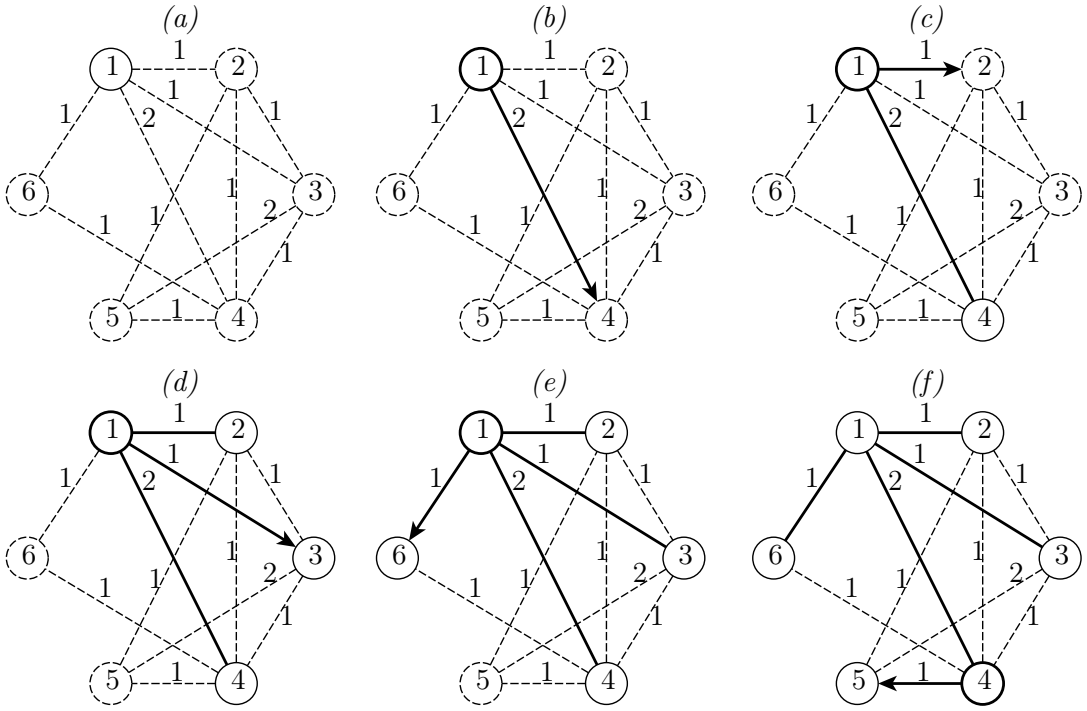

Figura 4.1: Execução da BFS para o sequenciamento de ferramentas na instância apresentada na Tabela 3.1.

Considerando o vértice 1 como o inicial, sua vizinhança possui 4 vértices {2, 3, 4 e 6}, explorados na ordem [4, 2, 3, 6], conforme ilustra a sequência de (a) a (e). Na sequência (f) o vértice 4 tem sua vizinhança explorada e somente o vértice 5 ainda não o havia sido. Neste ponto, todos os vértices já foram explorados e obtemos o seguinte sequenciamento de ferramentas $F_\phi = [1, 4, 2, 3, 6, 5]$.

O sequenciamento das tarefas ϕ é obtido a partir de F_ϕ da seguinte maneira: simula-se a disponibilização sucessiva das ferramentas em F_ϕ, uma a cada instante, verificando-se a composição das tarefas do problema, de modo que, caso a composição de uma tarefa seja o conjunto ou subconjunto das ferramentas disponíveis em determinado instante, esta tarefa é imediatamente inserida ao final da solução ϕ.

O Algoritmo 4.2 apresenta o método do sequenciamento de tarefas. No referido algoritmo, o conjunto F_{disp} é o conjunto de todas ferramentas disponíveis para utilização e o conjunto T_{aptas} é o conjunto de todas tarefas aptas a serem executadas (i.e., suas ferramentas já estão disponíveis).

Considere o exemplo da Tabela 3.1 e a sequência de ferramentas, $F_\phi = [1, 4, 2, 3, 6, 5]$. Na primeira iteração a ferramenta 1 se torna disponível, porém, nenhuma tarefa se torna apta. Então o algoritmo continua para sua segunda iteração, na qual a ferramenta 4 se torna disponível. Ainda assim, não existe nenhuma tarefa tal que todo o conjunto de ferramentas requeridas esteja disponível. Já na terceira iteração, a ferramenta 2 se torna disponível e,

Algoritmo 4.2: Sequenciamento da Tarefas

Dados: F_ϕ
$F_{disp} \leftarrow \emptyset$;
$\phi \leftarrow \emptyset$;
para cada $x \in F_\phi$ **faça**
  $F_{disp} \leftarrow F_{disp} \cup x$;
  $T_{aptas} \leftarrow \{i \in T \mid F_i \subseteq F_{disp}$ e $i \notin \phi\}$;
  enquanto $T_{aptas} \neq \emptyset$ **faça**
    se $\phi = \emptyset$ **então**
      $k \leftarrow$ tarefa $j \in T_{aptas}$ com o maior $|F_j|$;
    senão
      $k \leftarrow$ tarefa $j \in T_{aptas}$ que minimize $Z_{MTSP}(A^{\phi \cup j})$;
    Remova k de T_{aptas};
    Insira k ao final de ϕ;
retorna ϕ;

logo, a tarefa 1 se torna apta para ser executada, como a sequência ϕ ainda é vazia, utiliza-se o primeiro critério de desempate (tarefa com maior número de ferramentas), mas como só existe uma tarefa apta ela é adicionada a ϕ. Em seguida, a ferramenta 3 é disponibilizada e a tarefa 2 se torna apta, sendo adicionada a ϕ. Na quinta iteração, a ferramenta 6 torna-se disponível e então pode-se executar a tarefa 5. Finalmente, na última iteração, a ferramenta 5 se torna disponível e as tarefas 3 e 4 se t ornam aptas. Dado que adicionar a tarefa 3 adiciona menos trocas de ferramentas à solução parcial, esta é adicionada a ϕ, e por fim a tarefa 4 é adicionada. Ao final, tem-se $\phi = [1, 2, 5, 3, 4]$.

4.3 Um Novo Método de Busca Local

Com base na definição de *1-block* apresentada por (Crama et al., 1994), propõe-se uma nova busca local que tem como objetivo diminuir o número de *1-blocks* em cada linha da matriz A^ϕ. Esta busca local consiste em examinar iterativamente a matriz A^ϕ em procura por dois ou mais *1-blocks* em uma mesma linha. Ao encontrá-los, tenta-se agrupá-los contiguamente dois a dois pela movimentação das colunas relacionadas ao primeiro *1-block*, uma a uma, para antes ou depois das colunas relacionadas ao segundo *1-block*, o que for melhor em relação ao número de trocas de ferramentas correspondente. Caso ambos os movimentos de uma coluna piorem a solução, esta coluna não é movimentada. O Algoritmo 4.3 apresenta o pseudo-código da busca local proposta.

Para ilustrar o funcionamento do agrupamento de *1-blocks*, considere a instância do MTSP apresentada na Tabela 3.1 e a solução inicial $\phi = [1, 2, 3, 4, 5]$, mostrada na tabela 3.2(a), cujo número de trocas de ferramentas é 9. Inicialmente, examina-se a primeira linha da matriz e encontra-se o *1-block* j, formado pelas colunas 1 e 2. Em seguida encontra-se o

Algoritmo 4.3: Agrupamento de *1-blocks*

Dados: Matriz A^{ϕ}
para cada *linha* l *da matriz* A^{ϕ} **faça**
  Examine a linha l até encontrar um primeiro *1-block* j;
  enquanto *existir um próximo 1-block* k **faça**
    Para cada coluna de j decida com base com o número de trocas se esta deve ser inserida antes ou depois de k ou ser mantida em j;
    $j \leftarrow k$;
retorna A^{ϕ};

segundo *1-block* k, composto apenas pela coluna 5. Avalia-se então as possíveis realocações das colunas de j. Primeiro considera-se a coluna 1: as possíveis movimentações geram as sequências $[2, 3, 4, 1, 5]$ e $[2, 3, 4, 5, 1]$, resultando em 8 e 9 trocas respectivamente. Como há um movimento de melhora, ele é executado e a solução parcial é atualizada para $[2, 3, 4, 1, 5]$. Em seguida, examina-se a segunda coluna do primeiro *1-block*. As sequências decorrentes dos dois movimentos são $[3, 4, 1, 2, 5]$ e $[3, 4, 1, 5, 2]$, resultando em 8 trocas ambos. Não havendo piora do valor da solução, o segundo movimento é realizado, resultando na solução apresentada na Tabela 3.2(b). Na sequência, o método segue examinando as demais linhas da matriz, até ter examinado todas.

4.4 Busca Local Iterada

Após a geração de uma solução inicial pela aplicação da heurística proposta, optou-se por realizar o aprimoramento da mesma usando a estratégia de *Busca Local Iterada* (Lourenço et al., 2003), descrita anteriormente. Para compor este método, utilizou-se a solução inicial gerada pela heurística proposta na Seção 4.2, a busca local por agrupamento de *1-blocks* proposta na Seção 4.3 e o método de descida, descrito anteriormente, como mecanismo de busca local e, como mecanismo de perturbação, são realizados movimentos de troca. Ao utilizar como base o método de Lourenço et al. (2003) e adicionar os mecanismos de busca e perturbação específicos que foram propostos para o MTSP, obtém-se o pseudo-código apresentado no Algoritmo 4.4.

A perturbação de uma solução é realizada através de sucessivos movimentos de troca de posições entre duas tarefas selecionadas aleatoriamente. O número de vezes em que estes movimentos ocorrerão é definido como uma porcentagem do número de tarefas do problema indicado pelo parâmetro α. A aplicação de movimentos de trocas como mecanismo de perturbação visa evitar que a busca local fique estagnada em ótimos locais. A aplicação do método de descida como mecanismo de busca local possui a característica de somente serem realizadas alterações na solução que resultem em diminuição do valor da mesma. No intuito de redução do tempo de execução deste método, optou-se por aplicá-lo em uma proporção δ do espaço de

Algoritmo 4.4: Busca Local Iterada

Dados: ϕ, α, δ
Aplique a busca local de agrupamento de *1-blocks* em A^{ϕ};
enquanto *critério de parada não atendido* **faça**
    $\phi' \leftarrow \phi$;
    Perturbe a solução ϕ' em uma proporção α aplicando movimentos de troca;
    Aplique o método de descida em ϕ' em uma proporção δ;
    Aplique a busca local de agrupamento de *1-blocks* em ϕ';
    se $Z_{MTSP}(A^{\phi'}) < Z_{MTSP}(A^{\phi})$ **então**
        $\phi \leftarrow \phi'$;
retorna ϕ;

busca, selecionando aleatoriamente os pares de tarefas para troca de posição.

Empiricamente, definiu-se os valores dos parâmetros $\alpha = \lfloor 0,2 \times |T| \rfloor$ e $\delta = \left\lfloor 0,25 \times \binom{|T|}{2} \right\rfloor$. O critério de aceitação utilizado é a diminuição no valor da solução, e o critério de parada é definido como 200 iterações.

Capítulo 5

Experimentos Computacionais

O método proposto foi implementado utilizando a linguagem C++, compilado utilizando g++ 4.4.1 e a opção de otimização -O3. Os experimentos foram realizados em um computador com processador *Intel i5 Quad Core* de 3.2GHz, 8 GB de RAM, e sistema operacional Ubuntu 15.10.

Os resultados obtidos foram comparados com uma implementação combinada de *Clustering Search*, *Descida em Vizinhança Variável* e *Algoritmo Genético de Chaves Aleatórias Viciadas*, denominada *CS+BRKGA* Chaves et al. (2016), cujos resultados são os melhores para as instâncias da literatura e representam o estado da arte atualmente. Os resultados deste método foram obtidos em um computador com processador *Intel Core i7* de 3.4 GHz e 16GB de RAM. É importante notar que, apesar de diferentes, as arquiteturas originais de ambos os métodos utilizados neste experimentos possuem poder de processamento comparáveis.

Nas tabelas a seguir são apresentados o número de tarefas (n), o número de ferramentas (m), a capacidade da máquina (C), número de instâncias (e), a solução encontrada pela heurística construtiva proposta (S_0), a melhor solução encontrada por cada método (S^*), a solução média encontrada por cada método (S), o tempo médio de execução de cada método em segundos (T) e também o *gap* (ou distância percentual) em relação ao melhor resultado conhecido, calculado como $100 \times (S^* - \text{valor\_referência})/\text{valor\_referência}$. Valores em negrito indicam que as melhores soluções foram atingidas.

Devido à utilização de componentes de aleatoriedade pelos métodos, foram realizadas 20 execuções independentes por instância, avaliando-se também o desvio padrão médio das soluções (denotado por σ) obtidas para cada instância. Atenta-se ao fato de que o desvio padrão do método *CS+BRKGA* só é apresentado para os grupos de instâncias C_3 e C_4 de Crama et al. (1994). O trabalho original do método não reporta os valores de todas as execuções deste método, impossibilitando o cálculo para os demais grupos de instâncias.

5.1 Instâncias de Yanasse et al. (2009)

As 1350 instâncias propostas por (Yanasse et al., 2009) são divididas em 5 grupos (A, B, C, D e E). Os grupos A, B, C e D se diferem entre si pelo número de tarefas e pelo número de instâncias. O grupo E se difere dos demais pelos valores da capacidade da máquina.

A Tabela 5.1 apresenta os resultados obtidos no Grupo A. Por se tratar de instâncias triviais, neste conjunto ambas heurísticas encontraram todos os valores ótimos. Entretanto pode-se observar que o tempo médio de execução do método ($T = 0,11$s) proposto é menor do que o tempo médio de execução do *CS+BRKGA* ($T = 3,71$s). Para este primeiro Grupo é notável a robustez do ILS proposto ($\sigma = 0$, ou seja, os resultados médios se igualam às melhores soluções conhecidas).

Tabela 5.1: Resultados do Grupo A.

n	m	C	e	*CS+BRKGA*				Método Proposto					
				S^*	S	*gap*	T	S_0	S^*	S	*gap*	T	σ
8	15	5	10	**17,00**	**17,00**	0,00%	2,39	**17,00**	**17,00**	**17,00**	0,00%	0,06	0,000
8	15	10	30	**16,83**	**16,83**	0,00%	2,65	**16,83**	**16,83**	**16,83**	0,00%	0,08	0,000
8	20	5	10	**21,80**	**21,80**	0,00%	2,97	22,10	**21,80**	**21,80**	0,00%	0,06	0,000
8	20	10	30	**23,07**	**23,07**	0,00%	3,54	23,23	**23,07**	**23,07**	0,00%	0,15	0,000
8	20	15	60	**22,08**	**22,08**	0,00%	3,57	22,15	**22,08**	**22,08**	0,00%	0,12	0,000
8	25	5	10	**25,10**	**25,10**	0,00%	4,54	**25,10**	**25,10**	**25,10**	0,00%	0,03	0,000
8	25	10	30	**28,20**	**28,20**	0,00%	4,37	28,37	**28,20**	**28,20**	0,00%	0,16	0,000
8	25	15	60	**27,95**	**27,95**	0,00%	4,61	28,05	**27,95**	**27,95**	0,00%	0,21	0,000
8	25	20	100	**26,61**	**26,61**	0,00%	4,72	26,71	**26,61**	**26,61**	0,00%	0,15	0,000

Como no Grupo A, ambos os métodos alcançaram os valores ótimos no Grupo B, vide Tabela 5.2. Podemos perceber, mais uma vez, que o método proposto possui um tempo médio de execução menor ($T = 0,18$s, contra $T = 4,05$s do *CS+BRKGA*), além disso, o ILS possui um desvio padrão médio extremamente baixo ($\sigma = 0,0026$). Na grande maioria dos casos, as soluções médias do método proposto se igualaram às melhores soluções conhecidas.

Tabela 5.2: Resultados do Grupo B.

n	m	C	e	*CS+BRKGA*				Método Proposto					
				S^*	S	*gap*	T	S_0	S^*	S	*gap*	T	σ
9	15	5	10	**17,20**	**17,20**	0,00%	2,72	**17,20**	**17,20**	**17,20**	0,00%	0,08	0,000
9	15	10	30	**17,37**	**17,37**	0,00%	3,15	17,40	**17,37**	**17,37**	0,00%	0,12	0,000
9	20	5	10	**22,40**	**22,40**	0,00%	3,13	22,70	**22,40**	**22,40**	0,00%	0,09	0,000
9	20	10	30	**24,17**	**24,17**	0,00%	3,92	24,33	**24,17**	**24,17**	0,00%	0,22	0,000
9	20	15	60	**22,60**	**22,60**	0,00%	3,99	22,70	**22,60**	**22,60**	0,00%	0,20	0,000
9	25	5	10	**25,40**	**25,40**	0,00%	4,08	25,50	**25,40**	**25,40**	0,00%	0,05	0,000
9	25	10	30	**28,77**	**28,77**	0,00%	5,08	28,97	**28,77**	28,78	0,00%	0,24	0,012
9	25	15	50	**29,74**	29,75	0,00%	5,10	30,08	**29,74**	**29,74**	0,00%	0,39	0,000
9	25	20	100	**27,19**	**27,19**	0,00%	5,26	27,27	**27,19**	**27,19**	0,00%	0,24	0,005

Tal como é apresentado na Tabela 5.3, ambos métodos atingem o resultado ótimo da mesma maneira que os dois grupos anteriores, porém em geral o tempo do ILS proposto ($T = 1,67$s) foi

significantemente menor, em relação com o tempo do CS+BRKGA ($T = 9,83$s) e o resultado do ILS possui um desvio padrão médio $\sigma = 0,0034$. Novamente, em apenas uma ocasião as soluções médias do método proposto se igualaram às melhores soluções conhecidas.

Tabela 5.3: Resultados do Grupo C.

n	m	C	e	*CS+BRKGA*				Método Proposto					
				S^*	S	*gap*	T	S_0	S^*	S	*gap*	T	σ
15	15	5	10	**21,60**	21,69	0,00%	5,31	22,20	**21,60**	**21,60**	0,00%	0,60	0,000
15	15	10	30	**19,80**	19,88	0,00%	7,10	20,13	**19,80**	**19,80**	0,00%	1,02	0,000
15	20	5	10	**25,60**	25,77	0,00%	7,26	26,60	**25,60**	**25,60**	0,00%	0,75	0,000
15	20	10	30	**28,33**	28,52	0,00%	8,93	28,93	**28,33**	28,34	0,00%	1,82	0,018
15	20	15	60	**25,52**	25,65	0,00%	9,61	25,97	**25,52**	**25,52**	0,00%	2,00	0,002
15	25	5	10	**32,50**	32,70	0,00%	9,30	33,90	**32,50**	**32,50**	0,00%	0,71	0,000
15	25	10	30	**35,07**	35,30	0,00%	13,52	35,87	**35,07**	**35,07**	0,00%	2,19	0,007
15	25	15	60	**34,07**	34,27	0,00%	13,63	34,85	**34,07**	**34,07**	0,00%	3,49	0,000
15	25	20	100	**29,66**	29,79	0,00%	13,82	30,07	**29,66**	**29,66**	0,00%	2,41	0,003

Das 260 instâncias do grupo D, apenas para 189 se conhece as soluções comprovadamente ótimas. O *ILS* foi capaz de igualar todas as melhores soluções conhecidas e também estabelecer novas melhores soluções, conforme apresentado na Tabela 5.4.

Tabela 5.4: Resultados do Grupo D.

n	m	C	e	*CS+BRKGA*				Método Proposto					
				S^*	S	*gap*	T	S_0	S^*	S	*gap*	T	σ
20	15	5	10	26,10	26,58	0,77%	10,78	27,00	**25,90**	**25,90**	0,00%	1,61	0,000
20	15	10	20	**18,20**	18,44	0,00%	12,34	18,70	**18,20**	18,21	0,00%	2,70	0,021
20	20	5	10	29,30	29,93	0,17%	14,84	30,30	**29,20**	29,25	0,00%	2,14	0,069
20	20	10	10	**20,60**	20,76	0,00%	16,08	21,10	**20,60**	**20,60**	0,00%	3,22	0,000
20	20	15	30	**21,67**	21,79	0,00%	24,66	21,93	**21,67**	**21,67**	0,00%	4,16	0,000
20	25	5	10	**35,10**	35,74	0,00%	19,16	36,00	**35,10**	35,14	0,00%	2,35	0,050
20	25	10	10	**25,40**	25,47	0,00%	21,49	25,80	**25,40**	**25,40**	0,00%	4,10	0,000
20	25	15	40	**36,25**	36,75	0,00%	28,11	37,15	**36,25**	36,26	0,00%	9,35	0,016
20	25	20	40	**26,15**	26,28	0,00%	35,53	26,48	**26,15**	**26,15**	0,00%	4,17	0,000
25	15	10	10	**15,90**	16,00	0,00%	21,14	16,10	**15,90**	**15,90**	0,00%	4,56	0,000
25	20	10	10	**21,60**	22,05	0,00%	27,48	q22,20	**21,60**	**21,60**	0,00%	11,38	0,000
25	20	15	10	**22,60**	22,82	0,00%	25,66	22,90	**22,60**	**22,60**	0,00%	12,50	0,000
25	25	10	10	**26,60**	27,06	0,00%	36,88	27,70	**26,60**	26,70	0,00%	13,46	0,069
25	25	15	10	**25,00**	**25,00**	0,00%	54,70	**25,00**	**25,00**	**25,00**	0,00%	7,41	0,000
25	25	20	30	**25,50**	25,59	0,00%	66,10	25,67	**25,50**	**25,50**	0,00%	7,67	0,000

No conjunto D as diferenças se tornam mais claras entre *ILS* ($T = 6,05$s e $\sigma = 0,0113$) e *CS+BRKGA* ($T = 27,66$s). Não é possível determinar em quantas instâncias o resultado foi melhorado, dado que apenas os resultados médios são reportados na literatura. Mais uma vez, na maioria dos casos, as soluções médias se igualaram às melhores soluções conhecidas.

O comportamento visto anteriormente nos Grupos A, B e C se repete no Grupo E, vide Tabela 5.5. Desta vez, o ILS tem tempo de execução médio de $0,51$s enquanto o CS+BRKGA possui tempo médio de $6,54$s. Outra vez, o desvio padrão do ILS foi baixo ($\sigma = 0,0051$).

Exceto por uma ocasião, os resultados médios do método proposto se igualaram aos melhores resultados conhecidos. Em menor escala, o mesmo ocorre para a solução inicial gerada e para o método *CS+BRKGA*.

Tabela 5.5: Resultados do Grupo E.

n	m	C	e	*CS+BRKGA*				Método Proposto					
				S^*	S	*gap*	T	S_0	S^*	S	*gap*	T	σ
10	10	4	10	**13,50**	**13,50**	0,00%	1,55	13,70	**13,50**	**13,50**	0,00%	0,10	0,000
10	10	5	10	**11,20**	11,21	0,00%	1,96	11,30	**11,20**	**11,20**	0,00%	0,10	0,000
10	10	6	10	**10,30**	**10,30**	0,00%	2,88	**10,30**	**10,30**	**10,30**	0,00%	0,06	0,000
10	10	7	10	**10,00**	**10,00**	0,00%	3,45	**10,00**	**10,00**	**10,00**	0,00%	0,03	0,000
15	20	6	10	**27,40**	27,71	0,00%	7,09	28,40	**27,40**	27,42	0,00%	1,01	0,041
15	20	8	10	**22,20**	22,33	0,00%	7,68	22,80	**22,20**	**22,20**	0,00%	1,19	0,000
15	20	10	10	**20,30**	20,34	0,00%	12,71	20,40	**20,30**	**20,30**	0,00%	0,82	0,000
15	20	12	10	**20,20**	**20,20**	0,00%	14,97	**20,20**	**20,20**	**20,20**	0,00%	0,73	0,000

Os dados reportados para este conjunto de instâncias indicam com clareza o bom desempenho do método proposto sobre o método comparado bem como sua robustez.

5.2 Instâncias de Crama et al. (1994)

As 160 instâncias propostas por (Crama et al., 1994) foram divididas em 4 grupos (C_1, C_2, C_3 e C_4) que diferem entre si pelos valores de número de tarefas, número de ferramentas e capacidade da máquina.

O comportamento dos Grupos A, B, C e E se repete para os grupos C_1 e C_2 por se tratar de instâncias consideradas triviais e de pequenas dimensões (10 e 15 tarefas respectivamente). Os resultados para os grupos C_1 e C_2 são sumarizados na Tabela 5.6, os resultados de cada grupo são separados por uma linha em branco.

Tabela 5.6: Resultados dos Grupos C_1 e C_2.

n	m	C	e	*CS+BRKGA*				Método Proposto				
				S^*	S	*gap*	T	S_0	S^*	S	*gap*	T
10	10	4	10	**13,10**	13,11	0,00%	1,57	13,20	**13,10**	**13,10**	0,00%	0,10
10	10	5	10	**11,20**	**11,20**	0,00%	2,05	11,30	**11,20**	**11,20**	0,00%	0,10
10	10	6	10	**10,30**	**10,30**	0,00%	2,64	**10,30**	**10,30**	**10,30**	0,00%	0,06
10	10	7	10	**10,10**	**10,10**	0,00%	3,41	**10,10**	**10,10**	**10,10**	0,00%	0,04
15	20	6	10	**26,60**	26,87	0,00%	8,18	27,00	**26,60**	26,70	0,00%	1,03
15	20	8	10	**21,70**	21,72	0,00%	8,88	21,90	**21,70**	**21,70**	0,00%	1,45
15	20	10	10	**20,10**	**20,10**	0,00%	11,20	**20,10**	**20,10**	**20,10**	0,00%	1,04
15	20	12	10	**19,60**	**19,60**	0,00%	18,05	**19,60**	**19,60**	**19,60**	0,00%	0,61

Para os Grupos C_1 e C_2, ambos os métodos alcançaram os melhores resultados, porém, o ILS ($T = 0,075$s e $T = 1,033$, respectivamente) obteve soluções em um espaço de tempo

menor em relação ao CS+BRKGA ($T = 2,418$s e $T = 11,578$s, respectivamente). As soluções médias do método proposto se igualam às melhores conhecidas, o que implica em desvio padrão nulo para todas instâncias dos grupos C_1 e C_2. O mesmo ocorre, em menor número, para o método *CS+BRKGA*.

Os Grupos C_3 e C_4 exigiram maior esforço dos métodos comparados em relação aos anteriores. Não se conhece as soluções ótimas destas instâncias, entretanto, o método proposto foi capaz de determinar novos melhores resultados para parte das instâncias. Novamente, não é possível determinar em quantas instâncias o resultado foi melhorado, dado que se conhece apenas os resultados médios. Os resultados são detalhados na Tabela 5.7, novamente, os resultados de cada grupo são separados por uma linha em branco.

Tabela 5.7: Resultados dos Grupos C_3 e C_4.

n	m	C	e	*CS+BRKGA*				Método Proposto				
				S^*	S	*gap*	T	S_0	S^*	S	*gap*	T
30	40	15	10	106,80	108,03	0,25%	140,05	111,70	**106,40**	106,82	0,00%	104,03
30	40	17	10	88,70	89,98	0,23%	127,09	91,80	**88,50**	88,78	0,00%	160,88
30	40	20	10	70,70	71,85	0,28%	121,43	74,40	**70,50**	70,80	0,00%	228,83
30	40	25	10	53,10	53,97	0,38%	104,02	55,30	**52,90**	53,15	0,00%	206,65
40	60	20	10	199,80	202,25	0,55%	599,34	208,50	**198,70**	199,27	0,00%	533,78
40	60	22	10	175,30	177,28	0,75%	557,97	182,30	**174,00**	174,40	0,00%	801,76
40	60	25	10	147,50	149,35	0,68%	533,52	154,00	**146,50**	146,83	0,00%	1144,95
40	60	30	10	114,50	116,94	0,44%	473,56	118,70	**114,00**	114,39	0,00%	1908,61

Para o grupo C_3, o *ILS* se mostra mais robusto ($\sigma = 0,141$ e $T = 175,10$s) do que o *CS+BRKGA* ($\sigma = 0,680$ e $T = 123,15$s), porém, ao custo de maior tempo de execução. No grupo C_4 esta tendência se acentua: o *ILS* mantém baixo o erro em relação às melhores soluções ($\sigma = 0,284$), ao contrário do *CS+BRKGA* ($\sigma = 1,215$) que não foi capaz de obter as melhores soluções para nenhum dos subgrupos de instâncias. Entretanto, o tempo de execução médio do *ILS* ($T = 1097,28$s) é aproximadamente duas vezes o tempo de execução do *CS+BRKGA* ($T = 541,10$s), muito embora seja um tempo aceitável na prática. A Tabela 5.8 apresenta o desvio padrão obtido em cada grupo de instâncias por ambos os métodos comparados. Como mencionado anteriormente, a publicação original do método *CS+BRKGA* reporta o desvio padrão somente para os grupos C_3 e C_4.

Os dados reportados para este conjunto de instâncias reforçam a qualidade do método proposto, demonstrando robustez e sendo capaz de melhorar resultados anteriormente estabelecidos na literatura. Também, os resultados obtidos pela heurística proposta para geração de soluções iniciais é evidenciado.

Tabela 5.8: Desvio padrão dos Grupos C_3 e C_4.

n	m	C	e	*CS+BRKGA* σ	Método Proposto σ
30	40	15	10	0,730	0,177
30	40	17	10	0,810	0,129
30	40	20	10	0,690	0,138
30	40	25	10	0,490	0,119
40	60	20	10	1,310	0,359
40	60	22	10	1,130	0,254
40	60	25	10	1,110	0,296
40	60	30	10	1,310	0,227

5.3 Instâncias de Catanzaro et al. (2015)

Recentemente, Catanzaro et al. (2015) introduziram um novo conjunto de instâncias composto por 160 instâncias, divididas em 4 grupos (*datA*, *datB*, *datC* e *datD*) que diferem entre si pelos valores de número de tarefas, número de ferramentas e capacidade da máquina.

Na tabela 5.9 são sumarizados os resultados do ILS proposto para estes grupos de instâncias. Nesta tabela, a coluna *MSC* representa a melhor solução conhecida, reportada por Catanzaro et al. (2015) em conjunto com as instâncias. Não foi informado qual algoritmo e o tempo de execução para atingir tais soluções, embora o artigo original tenha comparado modelos de programação linear inteira de Tang e Denardo (1988), Laporte et al. (2004) e também outros três modelos propostos pelos mesmos. Apesar das características das instâncias possuírem semelhanças com as instancias de Crama et al. (1994), estes dois grupos de exemplares são diferentes e é possível perceber uma semelhança dos resultados do método proposto.

Tabela 5.9: Resultados para instâncias propostas por Catanzaro et al. (2015).

Grupo	n	m	C	e	Método Proposto					MSC
					S_0	S^*	S	T	σ	
$datA1$	10	10	4	10	12,70	**12,50**	**12,50**	0,09	0,000	**12,50**
$datA2$	10	10	5	10	**10,80**	**10,80**	**10,80**	0,09	0,000	**10,80**
$datA3$	10	10	6	10	**10,10**	**10,10**	**10,10**	0,05	0,000	**10,10**
$datA4$	10	10	7	10	**10,00**	**10,00**	**10,00**	0,04	0,000	**10,00**
$datB1$	15	20	6	10	29,20	**26,50**	**26,50**	1,02	0,000	26,90
$datB2$	15	20	8	10	22,60	**21,70**	21,71	1,38	0,022	22,00
$datB3$	15	20	10	10	20,10	**19,70**	**19,70**	1,07	0,000	19,80
$datB4$	15	20	12	10	**19,20**	**19,20**	**19,20**	0,55	0,000	**19,20**
$datC1$	30	40	15	10	119,90	**98,80**	99,22	111,269	0,185	102,00
$datC2$	30	40	17	10	101,10	**82,60**	82,82	164,640	0,140	85,90
$datC3$	30	40	20	10	79,70	**66,70**	66,83	229,752	0,145	69,40
$datC4$	30	40	25	10	58,20	**51,30**	51,39	191,483	0,091	53,60
$datC1$	40	60	20	10	239,20	**198,00**	198,58	489,231	0,290	203,20
$datC2$	40	60	22	10	211,40	**173,80**	174,17	721,795	0,234	179,00
$datC3$	40	60	25	10	178,20	**146,50**	147,02	1144,53	0,258	152,50
$datC4$	40	60	30	10	140,30	**115,00**	115,47	1853,48	0,203	120,90

Apenas para os grupos $datA$ e $datB4$, os resultados do ILS foram iguais aos melhores resultados conhecidos na literatura. Para todos os demais grupos, o ILS determinou novas melhores soluções. Nota-se, também, que o método proposto se mostra, mais uma vez, robusto, com um desvio padrão médio de no máximo $0,290$ trocas – nota-se que os valores da solução média se igualam às melhores soluções conhecidas para o grupo $datA$ e maior parte do grupo $datB$. Em quatro ocasiões, a solução inicial obtida também se iguala à melhor solução conhecida até então.

Apesar do tempo de execução ser baixo para os dois primeiros grupos, fica evidente que o mesmo cresce rapidamente de acordo com o tamanho da instância. Além disso pode-se observar a qualidade da solução inicial, em algumas instâncias foi possível atingir os melhores resultados. Por ser tratar de um algoritmo guloso, a solução inicial obtida pelo mesmo se deteriora de acordo com o crescimento do tamanho das instâncias, ainda assim esta é suficiente para limitar o espaço de busca considerado pelo ILS.

Por fim, os dados reportados para este último conjunto de instâncias embasam a conclusão da superioridade do método proposto sobre o estado da arte da literatura, sendo o mesmo capaz de suplantar os melhores resultados conhecidos com robustez.

Capítulo 6

Conclusões

A Minimização de Trocas de Ferramentas é um problema combinatório NP-Difícil de interesse prático por sua aplicabilidade industrial direta em sistemas de manufatura flexível.

Neste trabalho foram propostos uma nova representação em grafos para o problema, um novo método heurístico para geração de soluções e um novo método de busca local baseada no agrupamento de *1-blocks*. Estes métodos foram embutidos em uma estratégia de busca local iterada (ou *ILS*) juntamente com movimentos de troca e com o método de descida, utilizado como método de perturbação e de busca local, respectivamente.

Experimentos computacionais extensivos consideraram 1670 instâncias da literatura (divididas em treze grupos) e compararam o método proposto com o estado da arte, representado pela recente metaheurística *CS + BRKGA* e também com os melhores resultados conhecidos para algumas instâncias.

Para os nove primeiros grupos, ambos os métodos foram capazes de obter as soluções ótimas em baixo tempo de execução para grande parte das instâncias, porém, consideradas fáceis. Para os conjuntos de instâncias mais difíceis, a diferença de performance entre os métodos tornou-se clara. O *ILS* foi capaz de estabelecer novos melhores resultados para três diferentes grupos de instâncias. Já nos últimos quatro grupos, o *ILS* obteve resultados idênticos aos melhores resultados conhecidos na literatura para um grupo e para os demais foram estabelecidos novos melhores resultados. Adicionalmente, o *ILS* demonstrou em geral maior robustez do que o estado da arte ao obter um baixo desvio padrão entre as soluções obtidas em execuções independentes. Em vários casos, o desvio padrão obtido foi igual a zero, isto é, todas execuções resultaram em soluções com o mesmo número de trocas de ferramentas.

Pôde-se notar, também, que o método heurístico proposto para geração da solução inicial, na grande maioria do casos, apresenta soluções próximas da melhor conhecida ou até mesmo iguais, com um tempo computacional extremamente baixo. De uma maneira geral, os experimentos sustentam as conclusões sobre a qualidade e robustez do método proposto. Embora o tempo de execução tenha aumentado consideravelmente para instâncias de maiores dimensões, ainda tratam-se de valores aceitáveis na prática.

Os trabalhos futuros se concentrarão na pesquisa de novas estruturas de vizinhança e aceleração das já existentes, seja por melhoria das técnicas empregadas ou pelo emprego de novas técnicas, como paralelismo.

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.

Bard, J. F. (1988). A Heuristic for Minimizing the Number of Tool Switches on a Flexible Machine. *IIE Transactions*, 20(4):382–391.

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*.

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

Chaves, A. A.; Yanasse, H. H. e Senne, E. L. F. (2012). Uma nova heurística para o problema de minimização de trocas de ferramentas. *Gestão & Produção*, 19(1):17–30.

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. e Talloen, E. (2007). The Tool Switching Problem Revisited. *European Journal of Operational Research*.

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

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.

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.

Lourenço, H. R.; Martin, O. C. e Stützle, T. (2003). *Iterated local search*. Springer.

Marvizadeh, S. Z. e Choobineh, F. F. (2013). Reducing the number of setups for CNC punch presses. *Omega*, 41(2):226–235.

Matzliach, B. e Tzur, M. (1998). The online tool switching problem with non-uniform tool size. *International Journal of Production Research*, 36(12):3407–3420.

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

Shirazi, R. e Frizelle, G. D. M. (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.

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 & Produção*, 16(3):370–381.