Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM

Novos Algoritmos Heurísticos para o Problema de Minimização de Troca de Ferramentas

Bolsista: Gustavo Silva Paiva
Orientador: Marco Antonio Moreira de Carvalho – DECOM/UFOP

Nota: Relatório referente ao período de 01/03/2015 a 30/02/2016, apresentado à Universidade Federal de Ouro Preto, como parte das exigências do programa de iniciação científica -PIP

Local: Ouro Preto - Minas Gerais - Brasil
Data: 28 de fevereiro de 2016

Título do Resumo

Assinatura do orientador(a): _____

Nome Completo do orientador(a)

Assinatura do bolsista: _____

Nome Completo do bolsista

Sumário

Capítulo 1

Introdução

1.1 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.

1.2 Descrição do Problema

Em um sistema de manufatura flexível, existem tarefas que devem ser realizadas por uma máquina que possui um compartimento limitado de ferramentas. Um plano de produção determina como estas tarefas devem ser sequenciadas na linha de produção. Cada tarefa requer um conjunto de diferentes ferramentas para que seja realizada.

Para executar diferentes tarefas em sequência em uma mesma linha de produção, tornam-se obrigatórias as trocas de ferramentas quando o número total de ferramentas necessárias para realizar as tarefas for maior do que a capacidade do respectivo compartimento – o que é o caso comum. A quantidade destas trocas deve ser minimizada de forma a aumentar a produtividade pela diminuição das interrupções na linha de produção e consequentemente o tempo ocioso da máquina de produção.

O Problema de Minimização de Trocas de Ferramentas (*Minimization of Tool Switch Problem* – MTSP) consiste em determinar uma sequência de tarefas, que serão executadas minimizando o número de trocas de ferramentas.

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 and 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 and Denardo (1988).

Uma instância do MTSP pode ser descrita pelos seguintes dados: o 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.

Uma solução do MTSP é representada pela permutação T_ϕ do conjunto T e também de um plano de trocas de ferramentas. O plano e o número de trocas de ferramentas de T_ϕ podem ser determinados pelo algoritmo KTNS, como observado anteriormente.

A Tabela 1 apresenta um exemplo numérico de uma instância do MTSP. A primeira linha representa as tarefas (enumeradas de 1 a 5) e sua ordem de execução. Em seguida, 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 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	7	5	6
	7			6	
Capacidade do compartimento = 4					

Uma solução da instância apresentada na Tabela 1 é apresentada na Tabela 2. A sequência de tarefas realizadas nesta solução é $T_\phi = \{5,1,4,2,3\}$. As ferramentas sublinhadas não são necessárias para executar a tarefa da respectiva coluna, mas foram mantidas no compartimento para que, eventualmente, minimizem o número de trocas para execução de uma tarefa subsequente. Para execução desta solução são necessárias 11 trocas de ferramentas sendo estas: quatro trocas para inserção das ferramentas iniciais na máquina (ferramentas 1, 2, 4 e 6), uma troca

na transição entre as tarefas 5 e 1 (ferramenta 6 pela ferramenta 7), três trocas na transição entre as tarefas 1 e 4 (ferramentas 1, 4 e 7 pelas ferramentas 3, 5 e 6), uma troca na transição entre as tarefas 4 e 2 (ferramenta 2 pela ferramenta 1) e outras duas trocas na transição entre as tarefas 2 e 3 (ferramentas 5 e 6 pelas ferramentas 4 e 7).

Tabela 2: *Exemplo de uma solução do MTSP.*

Tarefas	5	1	4	2	3
Ferramentas	1	1	2	1	$\underline{1}$
	$\underline{2}$	2	3	3	3
	4	4	5	5	4
	6	7	6	$\underline{6}$	7
Capacidade da Compartimento = 4					

A Tabela 3 apresenta a solução ótima para a instância apresentada na Tabela 1. A sequência de tarefas nesta solução é $T^*_\phi = \{1,3,5,2,4\}$, exigindo 8 trocas de ferramentas para que seja executada, sendo estas: quatro trocas para inserção das ferramentas inicias na máquina (ferramentas 1, 2, 4 e 7), uma troca na transição entre as tarefas 1 e 3 (ferramenta 2 pela ferramenta 3), uma troca na transição entre as tarefas 3 e 5 (ferramenta 7 pela ferramenta 6), uma troca na transição entre as tarefas 5 e 2 (ferramenta 4 pela ferramenta 5) e uma última troca na transição entre as tarefas 2 e 4 (ferramenta 1 pela ferramenta 2).

Tabela 3: *A solução ótima da instância da Tabela 1 MTSP.*

Tarefas	1	3	5	2	4
Ferramentas	1	$\underline{1}$	1	1	2
	2	3	3	3	3
	4	4	$\underline{4}$	5	5
	7	7	6	$\underline{6}$	6
Capacidade da Compartimento = 4					

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.

Capítulo 2

Objetivos

2.1 Objetivos Gerais

1. Elaborar heurísticas consistentes e robustas que possam ser utilizadas no contexto de problemas de minimização de trocas de ferramentas que permitam a obtenção rápida de soluções próximas da solução ótima sem que se perca a vantagem da busca sistemática – inicialmente considerando problemas específicos, mas com uma possibilidade de generalização;
2. Avaliar o comportamento heurístico de diferentes formulações para problemas de sequenciamento;
3. Pesquisar técnicas para melhoria de soluções obtidas por heurísticas usando programação linear inteira;
4. Ampliar o conjunto de problemas testes com soluções ótimas já conhecidas através da execução de testes sistemáticos usando formulações já desenvolvidas;
5. Buscar a aplicação prática dos métodos desenvolvidos em contextos reais, a fim de que também seja constituído um avanço para as indústrias nacionais;
6. Além dos objetivos principais, outros produtos deste projeto de pesquisa serão trabalhos publicados em periódicos e eventos nacionais e internacionais, os quais contribuem para a promoção dos centros de pesquisas nacionais e também da tecnologia.

2.2 Objetivos Específicos

1. Propor um modelo de otimização que contemple apropriadamente as especificidades de diferentes aplicações do problema com o uso dos modelos descritos na literatura;
2. Implementação computacional de um *software* para determinação do número mínimo de troca de ferramentas usando ferramentas de Engenharia de Produção e Inteligência Computacional, o que inclui a utilização de métodos heurísticos e meta-heurísticos;
3. Avaliação do *software* implementado considerando dados reais e também com problemas teste publicamente disponíveis;

Capítulo 3

Fundamentos Teóricos e Revisão

Serão tratados, nesta seção, os trabalhos mais relevantes referentes ao problema de Minimização de Trocas de Ferramentas.

Tang and Denardo (1988) apresentaram uma política ótima que minimiza o número total de trocas de ferramentas dada uma sequência de tarefas fixa, denominada KTNS (*Keep Tools 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 lagrangeana.

Crama et al. (1994) mostrou uma prova formal de que o problema é NP-Difícil quando $C \geq 2$. Foi proposta a heurística gulosa MSG (*Multiple Start Greedy*) que verifica, a cada iteração, a melhor tarefa a ser adicionada em uma sequência inicialmente vazia. Também foi proposta, uma formulação do MTSP baseada na formulação do problema do caixeiro-viajante, de forma a ser possível a utilização das heurísticas desse clássico problema, já bastante estudado. Com essa mesma idéia, Hertz et al. (1998) propuseram diferentes métricas para calcular os pesos das arestas na mesma formulação de Crama et al. (1994). Também foi adaptada uma função objetivo, para que as heurísticas desenvolvidas originalmente para o problema do caixeiro-viajante se adequassem ao MTSP.

Matzliach and Tzur (1998) estudaram o MTSP com ferramentas de tamanho não uniforme em um ambiente dinâmico. Foram desenvolvidas três heurísticas para este problema, utilizando o conceito da política LRU (*Least Recently Used*) do problema de paginação de memória em computadores. Assim, essas heurísticas tentam manter na máquina as ferramentas que foram recentemente utilizadas. Uma dessas heurísticas considera também, a mudança de hábitos na demanda dos produtos, isto é, uma mudança da frequência em que um produto é demandado, adaptando-se a um ambiente mais realístico.

Shirazi and Frizelle (2001) fizeram uma comparação entre as heurísticas de Crama et al. (1994) e outras heurísticas da literatura em instâncias reais de indústrias. Percebeu-se que os métodos utilizados na prática, poderiam ser melhorados pelas heurísticas descritas anteriormente. Também foi observado que as instâncias reais são mais fáceis do que as instâncias geradas pela literatura, pois existem 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.

Para o MTSP com máquinas idênticas que operam paralelamente, Fathi and Barnette (2002) utilizaram buscas locais e a heurística do problema de escalonamento de tarefas em máquinas paralelas *List-Processing heuristic*, que consiste na criação de uma lista de tarefas que são executadas assim que uma máquina se torna disponível.

Al-Fawzan and 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.

Zhou et al. (2004) apresentaram uma aplicação da metaheurística *beam-search*, baseada em um novo esquema *branch-and-bound*, limitando o número de nós que serão explorados em cada nível da árvore de busca. (Resultados)

Crama and Talloen (2007) revisitaram o MTSP, mostrando que quando se considera ferramentas de tamanho não uniforme, especificamente com uma sequência fixa de tarefas, o problema se mantém NP-Difícil pela redução deste ao problema de *3-Partition*. No entanto, com um valor fixo de C, é possível a resolução desse problema em tempo determinístico polinomial, porém, a um alto custo exponencial.

Amaya et al. (2008), pela primeira vez, utilizaram algoritmos meméticos para abordar o MTSP. Foram apresentados uma busca local, um algoritmo genético e um algoritmo memético, sendo o último obtido pelo apri-

moramento de um algoritmo genético com uma busca local. Essas metaheurísticas, principalmente o algoritmo memético, atingiram resultados convincentes obtendo um número de trocas menor comparado com o *beam-search* de Zhou et al. (2004).

Yanasse et al. (2009) propôs um algoritmo *branch-and-bound*, este também introduziu um conjunto de instâncias novo. Apesar de mostrar resultados mais satisfatórios do que o *branch-and-bound* 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. Senne et al. (2009) apresentaram metaheurísticas *beam-search* baseadas no algoritmo de *branch-and-bound*. O resultado dessas podem ser utilizados como limitante superior para acelerar métodos exatos reduzindo o espaço de busca.

Amaya et al. (2011) abordou o problema novamente com a ideia de algoritmos meméticos cooperativos formando uma cadeia de agentes meméticos que se comunicam entre si por meio de topologias famosas de redes: Anel, *Broadcast* e Aleatório. Estes novos algoritmos meméticos obtiveram resultados superiores quando comparados com os algoritmos propostos por Amaya et al. (2008) e, consequentemente, por Zhou et al. (2004).

Chaves et al. (2012) propôs uma heurística de duas fases: uma fase de criação, onde é gerada uma solução viável, e outra de aprimoramento, aplicando a metaheurística *Iterated Local Search* no intuito de encontrar um ótimo local a partir solução gerada anteriormente. A solução obtida por essa heurística foi utilizada como limitante superior no algoritmo *branch-and-bound* de Yanasse et al. (2009), diminuindo em alguns casos até 70% o tempo de execução e número de nós examinados nas instâncias do mesmo. O número de exemplares solucionados por esse algoritmo também aumentou de 109 para 189 dos 260 exemplares testados.

Amaya et al. (2013) aprimorou seus algoritmos meméticos com a técnica de *Cross-Entropy*, uma abordagem de Monte Carlo para problemas de otimização combinatória, melhorando ainda mais os resultados obtidos por Amaya et al. (2011), mostrando que a utilização de metaheurísticas é uma opção viável para a solucionar o MTSP.

Novamente o MTSP com ferramentas não uniformes foi estudado, desta vez por Marvizadeh and Choobineh (2013) que formularam um modelo de programação inteira. Além disso, foram apresentadas três heurísticas: duas de fusão de conjuntos, com o objetivo de unir conjuntos de tarefas que podem ser executadas sem que haja trocas de ferramentas entre elas, e também um algoritmo genético. Pode-se perceber que para pequenas instâncias a distância entre a solução gerada pela heurísticas e a solução ótima é pequena e também foi observado que o algoritmo genético proposto obteve os melhores resultados para instâncias de indústrias reais, em um tempo aceitável.

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.

Raduly-Baka and Nevalainen (2015) apresentaram o MTSP modular, no qual existem módulos que suportam mais do que uma ferramenta e podem ser trocados com custo unitário. Além disto foi provado que o MTSP modular também é NP-Difícil.

Adjiashvili et al. (2015) trataram o MTSP que considera um tempo de instalação para cada ferramenta e, também, um tempo de produção para cada tarefa, além disso, podem ocorrer trocas sem a necessidade de paradas na máquina, contanto que exista um espaço no compartimento de ferramentas que não tenha sido utilizado por tempo suficiente para instalar uma nova ferramenta. O foco deste trabalho foi para o *Machine Stop Minimization* para o caso do MTSP com uma sequência fixa de tarefas. Foi provado e demonstrado a otimalidade de um algoritmo guloso que resolve este problema em tempo polinomial. Além disso também foi demonstrado uma nova função objetivo para o caso geral do MTSP, chamado de *Split Minimization* objetivando diminuir o somatório do chamado *Split Degrees* de cada ferramenta, que é a quantidade de diferentes espaços em que uma mesma ferramenta foi inserida. Já este segundo problema foi provado ser NP-Difícil.

Chaves et al. (2016) apresentaram uma nova abordagem ao 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*. Os resultados apresentados pelo *CS+BRKGA* foram superiores aos resultados obtidos pelo ILS proposto anteriormente em Chaves et al. (2012), considerando o conjunto de instâncias proposto em Yanasse et al. (2009) e Crama et al. (1994).

Capítulo 4

Materiais e Métodos

Nesta primeira etapa do projeto, foi realizada uma revisão sistemática da literatura a respeito do problema tratado e também dos problemas relacionados, conforme descrito no capítulo anterior. Todos os trabalhos foram classificados de acordo com o problema tratado, abordagem e data. Desta forma, foi traçado um panorama acurado do estado da arte da pesquisa relacionada ao problema tratado.

Ainda, foram coletadas instâncias e geradores de instâncias adotados pela comunidade acadêmica para o problema tratado, recursos estes que serão utilizados em fases subsequentes do projeto.

O material gerado durante o período aqui relatado será utilizado para desenvolver uma ferramenta para a solução aproximada do Problema de Minimização de Trocas de Ferramentas. Esse *software* irá incorporar as tecnologias estado-da-arte em otimização para a resolução de um modelo construído com base em experiências encontradas na literatura e eventualmente em estudos em empresas que estabeleçam um convênio com a universidade. Os métodos desenvolvidos permitirão que a produção com máquinas flexíveis seja mais eficiente, gerando melhor aproveitamento dos recursos disponíveis e menor custo associado.

Após a validação dos resultados obtidos, será realizada uma avaliação crítica a respeito do trabalho desenvolvido, abrangendo todas as estruturas e estratégias de solução empregadas, a fim de analisar o desempenho individual de cada uma.

Os resultados obtidos pelos métodos propostos serão comparados com resultados obtidos por métodos que compõem o estado da arte em relação a cada problema específico. Na comparação, serão analisados e discutidos a relação entre o desempenho dos métodos propostos e as propriedades de cada solução obtida.

Os algoritmos e programas desenvolvidos serão documentados de forma apropriada e as propostas e resultados experimentais serão divulgados para a comunidade científica.

Capítulo 5

Desenvolvimento

Neste capítulo serão descritos os métodos implementados durante este trabalho. Para facilitar a compreensão das descrições dos métodos, considere N sendo o conjunto de tarefas, M o conjunto de Ferramentas e C a capacidade do compartimento da máquina e F_i o conjunto das ferramentas necessárias para execução da tarefa i. Considere também $Custo(\phi, j)$ como sendo o custo de inserção da tarefa j ao fim da sequência ϕ e $NumeroTrocas(\phi)$ como sendo o menor número de trocas para execução da sequência ϕ.

5.1 *Keep Tools Needed Soonest* – KTNS

Desenvolvido por Tang and Denardo (1988), o KTNS é um algoritmo para calcular o menor número de trocas de ferramentas de uma sequência de tarefas, ou seja, é utilizado para avaliar uma sequência. Este algoritmo mantém no compartimento da máquina as ferramentas que serão usadas mais recentemente, para isto a cada execução de uma tarefa - cujas ferramentas não ocupam toda a capacidade do compartimento - é adicionado uma nova ferramenta da mais recente tarefa caso seja necessário. Sua complexidade é $O(|M||N|)$. Considere $S_{i,k}$ como sendo o conjunto das k ferramentas que serão utilizadas mais recentemente nas tarefas após a tarefa i.

Algoritmo 1: KTNS - Keep Tools Needed Soonest

```
Dados: Uma sequência de tarefas φ
Resultado: Sequência ótima de troca de ferramentas
para cada i ∈ φ faça
    se |F_i| < C então
        k ← C − |F_i|;
        F_i ← F_i ∪ S_{i,k};
    fim
fim
```

5.2 *Multiple Start Greedy*

Multiple Start Greedy foi apresentada por Crama et al. (1994), apesar de ser uma heurísticas simples ela é muito eficaz. A partir de uma tarefa inicial são testadas, a cada iteração, todas as tarefas que ainda não estão na sequência atual para ser adicionada ao fim da sequência com o intuito de achar qual a melhor tarefa para ser executada no fim desta sequência. Para analisar qual tarefa é a melhor é utilizado o algoritmo *KTNS* para descobrir o número de trocas de ferramentas. Esta heurística é executada considerando cada uma das tarefas como sendo a tarefa inicial.

Sua complexidade é $O(|M||N|^4)$.

Algoritmo 2: Multiple Start Greedy

Dados: Uma instância do MTSP
$MelhorSequencia \leftarrow \emptyset$
para cada $i \in N$ **faça**
  $Q \leftarrow \{i\}$;
  $S \leftarrow N \setminus \{i\}$;
  enquanto $S \neq \emptyset$ **faça**
    $k \leftarrow min(Custo(Q, j)), j \in S$;
    $Q \leftarrow Q \cup \{k\}$;
    $S \leftarrow S \setminus \{k\}$;
  fim
  se $NumeroTrocas(Q) < NumeroTrocas(MelhorSequencia)$ **então**
    $MelhorSequencia \leftarrow Q$
  fim
fim
retorna $MelhorSequencia$

5.3 Heurísticas do Problema do Caixeiro-Viajante – PCV

As duas próximas heurísticas também foram apresentadas por Crama et al. (1994) e são heurísticas do PCV adaptadas para o MTSP, para que estas tenham sido adaptadas foi criado um grafo completo $G = (V, E)$, tal que V é o conjunto de vértices representando as tarefas do MTSP e o conjunto de arestas (i, j) representando um limitante inferior para o número de trocas ao executar a tarefa j após a tarefa i, este limitante inferior será utilizado como sendo a distância entre dois vértices. Considere $Distancia(x, y)$ como sendo o peso da aresta que liga o vértice x ao vértice y.

5.3.1 *Nearest Neighbor*

A heurística *Nearest Neighbor* é uma clássica heurística do PCV. É escolhida uma tarefa inicial para fazer parte de uma sequência inicialmente vazia e então a cada iteração é adicionado ao fim desta sequência a tarefa ainda não executada que possui a menor distância em relação à última tarefa da sequência atual. Esta heurística é executada considerado cada uma das tarefas como sendo a tarefa inicial. Sua complexidade é $O(|N|^3)$.

Algoritmo 3: Nearest Neighbor

Dados: Uma instância do MTSP
$MelhorSequencia \leftarrow \emptyset$
para cada $j \in N$ **faça**
  $Q \leftarrow \{j\}$;
  $S \leftarrow N \setminus \{j\}$;
  $UltimoVertice \leftarrow j$;
  enquanto $S \neq \emptyset$ **faça**
    $k \leftarrow min(Distancia(UltimoVertice, i)), i \in S$;
    $Q \leftarrow Q \cup \{j\}$;
    $S \leftarrow S \setminus \{j\}$;
    $UltimoVertice \leftarrow k$;
  fim
  se $NumeroTrocas(Q) < NumeroTrocas(MelhorSequencia)$ **então**
    $MelhorSequencia \leftarrow Q$
  fim
fim
retorna $MelhorSequencia$

5.3.2 *Farthest Insertion*

Farthest Insertion é uma heurística mais robusta para o PCV, nela tenta-se eliminar as piores escolhas possíveis que podem ocorrer na heurística *Nearest Neighbor*. É escolhida uma tarefa inicial para fazer parte de uma sequência inicialmente vazia e então a cada iteração encontra-se o vértice ainda não sequenciado com a maior distância para ser inserido após uma tarefa que já foi sequenciada, e então é colocado este vértice na melhor posição da sequência atual, desta forma estas más decisões não irão ocorrer. Sua complexidade é $O(|N|^4)$.

Algoritmo 4: Farthest Insertion

```
Dados: Uma instância do MTSP
MelhorSequencia ← ∅
para cada i ∈ N faça
    Q ← {i};
    S ← N \ {i};
    UltimoVertice ← i;
    enquanto S ≠ ∅ faça
        k ← min(Distancia(UltimoVertice, j)), j ∈ S;
        Q ← Q ∪ {k};
        S ← S \ {j};
        UltimoVertice ← k;
    fim
    se NumeroTrocas(Q) < NumeroTrocas(MelhorSequencia) então
        MelhorSequencia ← Q
    fim
fim
retorna MelhorSequencia
```

5.4 *Best Position Insertion*

É uma heurística simples e gulosa, inicialmente o conjunto de tarefas é ordenado pela quantidade de ferramentas necessárias para execução das tarefa. A tarefa com maior número de ferramentas, empates são desfeitos aleatoriamente, para fazer parte de uma sequência inicialmente vazia e então a cada iteração é adicionado a próxima tarefa com o maior número de ferramentas na melhor posição da sequência atual, o método KTNS é utilizado para descobrir a melhor posição. Sua complexidade é $O(|M||N|^3)$. Considere a função OrdenaDecrescente como sendo a função que ordena o conjunto de tarefas de acordo com número de ferramentas necessárias para execução de cada uma e a função InsereMelhorPosicao(Q, E) como sendo a função que insere a tarefa E na melhor posição da sequência Q considerando todas possibilidades, desde a primeira até a última posição de Q.

Algoritmo 5: Best Position Insertion

```
Dados: Uma instância do MTSP
T ← OrdenaDecrescente(N);
Q ← Q ∪ {T1};
T ← T \ T1;
enquanto T ≠ ∅ faça
    E ← {T1};
    T ← T \ E;
    InsereMelhorPosicao(Q, E);
    S ← S \ {E};
fim
retorna Q
```

5.5 K-OPT

A heurística K-OPT é uma simples busca local. A partir de uma sequência inicial são feitas trocas k de elementos, para isto é calculado todas os k-combinações de elementos e em seguida todas as possíveis permutações, de cada

k-combinação, são realizadas e classificadas com o algoritmo KTNS. Após verificar todas essas possibilidades a melhor permutação gerada é comparada com a sequência inicial, caso seja melhor ela se torna a nova sequência inicial e o processo reinicia caso contrário retorna a sequência inicial como sendo a melhor sequência gerada. Considere a função GeraKCombinações(T, k) como sendo a função que gera todas as k-combinações de um sequência T e a função GeraPermutações(T, C, k) como sendo a função que gera todas as permutações dos k elementos da combinação C na sequência T.

Algoritmo 6: K-OPT

```
Dados: Uma instância do MTSP, um valor de k
T ← {1,2···|N|};
TiverMelhoria ← verdadeiro;
melhorSequencia ← {∅};
enquanto TiverMelhoria == verdadeiro faça
    MelhorSequencia ← T;
    para cada C ∈GeraKCombinações(T,k) faça
        para cada ϕ ∈GeraPermutações(T,C,k) faça
            se NumeroTrocas(ϕ) < NumeroTrocas(MelhorSequencia) então
                MelhorSequencia ← Q;
            fim
        fim
    fim
    se NumeroTrocas(MelhorSequencia) < NumeroTrocas(T) então
        T ← MelhorSequencia;
    senão
        TiverMelhoria ← falso;
    fim
fim
retorna T
```

5.6 Heurística Construtiva

A Heurística Construtiva baseia-se no fato de tentar reduzir o número de ferramentas restantes a ser utilizada, por exemplo, tentar executar inicialmente as tarefas que possuem ferramentas que são utilizadas poucas vezes. Para isso é criado um Grafo G = (V, A), tal que as ferramentas representam os vertices, e as arestas ligam as ferramentas de uma mesma tarefa, uma tarefa que requer as ferramentas 1, 2 e 3 criariam um conjunto de arestas {1, 2}, {1, 3} e {2, 3}. Considere a função CriaGrafo() como a função que cria o grafo determinado no texto acima. Considere também a função VerificaCompartimento() como sendo a função que verifica se existe alguma tarefa que pode ser executa com a configuração atual do compartimento de ferramentas e a função AtualizaGrafo() como sendo a função que remove as arestas de tarefas executadas.

Algoritmo 7: Heurística Construtiva

```
Dados: Uma instância do MTSP
G ← CriaGrafo();
S ← {∅}
v ←Vertice com menor grau diferente de 0;
enquanto grau(v) > 0 faça
    t ← aresta de v (tarefa) com o menor incremento no número de trocas;
    S ← S∪t;
    VerificaCompartimento();
    AtualizaGrafo();
    v ←Vertice com menor grau diferente de 0;
fim
retorna S
```

5.7 Pré-processamento

Nesta seção será tratado o tema de pré-processamento de instâncias do MTSP. Este tipo de técnica tem o objetivo de reduzir a dificuldade do problema e consequentemente o seu tamanho. O pré-processamento é, geralmente, eficaz em instâncias reais, aonde existe grandes probabilidades de existência de características passíveis de redução. Em instâncias artificiais, geradas pela literatura, este tipo de característica não existe pois estas são feitas exatamente para que não seja possível a utilização desta técnica.

5.7.1 Dominância de Tarefas

Dominância de tarefas é a característica mais evidente de pré-processamento do MTSP. Esta ocorre quando o conjunto de ferramentas de uma tarefa é subconjunto do conjunto de ferramentas de uma outra tarefa, quando isto ocorre podemos retirar a tarefa dominada do conjunto de tarefas a serem sequenciadas pois após a execução da tarefa dominante é possível a execução da tarefa dominada sem que ocorra nenhuma troca de ferramenta. Em termos de algoritmo, é feito uma verificação de cada tarefa $x \in N$ procurando a existência de uma outra tarefa $y \in N \setminus x$ tal que $F_x \subset F_y$.

Algoritmo 8: Dominância de Tarefas

```
Dados: Uma instância do MTSP
para cada x ∈ N faça
    Q ← N \ x;
    para cada y ∈ Q faça
        se F_x ⊂ F_y então
            X é dominada por Y;
        fim
    fim
fim
```

5.8 Heurística Proposta

Nesta seção será apresentada a heurística proposta neste trabalho. Esta foi dividida em três etapas, etapa de Sequenciamento das Ferramentas, de Sequenciamento das Tarefas e de Refinamento.

Para apresentar a nova heurística é necessário fazer algumas definições. A primeira definição é o Grafo de Ferramentas, definido como $G = (V, A)$, no qual V é o conjunto de vertices que representam as Ferramentas e A é o conjunto das arestas (i, j) que representam se uma ferramenta i é utilizada ao mesmo tempo da ferramenta j, cada aresta possui um peso associado que é a quantidade de vezes que a ferramenta i e a ferramenta j aparecem juntas em uma tarefa.

5.8.1 Sequenciamento das Ferramentas

A primeira etapa desta heurística sequencia as ferramentas de forma a unir ferramentas que tendem a ser utilizadas em conjunto. A partir de um ferramenta inicial é executada uma *BFS* com a adição de um critério para definir qual será a ordem de inserção dos vizinhos de um vértice a Fila da *BFS*. Este critério é baseado no peso das arestas, pesos maiores indicam ferramentas que são utilizadas mais vezes em pares na máquina.

Considere o tipo $Fila$ uma implementação com os métodos básicos de uma estrutura adaptada como fila, considere também a função $OrdenaDecrescente()$ e a função $Vizinhos(v)$ que ordena um conjunto de arestas a

partir de seus pesos e a função que retorna os vizinhos de um vértice v, respectivamente.

Algoritmo 9: Sequenciamento de Ferramentas via *BFS*

Dados: Uma instância do MTSP, um vértice inicial x
$F_\phi \leftarrow \emptyset$;
$NaoVisitados \leftarrow N$;
$Acabou \leftarrow False$;
$Q \leftarrow Fila()$;
$Q.push(x)$;
enquanto $Acabou == False$ **faça**
  enquanto $Q \neq \emptyset$ **faça**
    $front \leftarrow Q.pop()$;
    $NaoVisitados \leftarrow NaoVisitados \setminus \{front\}$;
    $VizinhosOrdenados \leftarrow OrdenaDecrescente(Vizinhos(front))$;
    para cada $v \in VizinhosOrdenados$ **faça**
      se $v \notin F_\phi$ **então**
        $F_\phi \leftarrow F_\phi \cup \{v\}$;
      fim
    fim
  fim
  se $NaoVisitados \neq \emptyset$ **então**
    $Q.push(NaoVisitados[1])$;
  senão
    $Acabou \leftarrow True$;
  fim
fim
retorna F_ϕ

Exemplo Gráfico

A Tabela 4 mostra a matriz de Ferramentas e Tarefas para uma instância do MTSP com 5 tarefas, 6 ferramentas e uma máquina flexível com capacidade para 3 ferramentas.

Tabela 4: *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	6
Capacidade do compartimento = 3					

Na figura 5.1 é ilustrado o passo a passo da execução do método 9 : Sequenciamento de Ferramentas com a *BFS* utilizando o peso da aresta para definir a ordem de visita a cada nó. Considerando o nó 1 como o nó inicial, temos que inicialmente a fila Q contém somente o nó 1 então este será visitado. O nó 1 possui 4 vizinhos (2, 3, 4, 6) a ordem da inserção destes na fila é definido pelo peso da aresta, então a ordem de inserção será 4, 2, 3 e 6. Em seguida o nó 4 será visitado porém todos seus vizinhos já estão em Q, logo ele não adiciona ninguem em Q. Quando o nó 2 é visitado somente o vizinho 5 ainda não foi adicionado na fila então ele é adicionado em Q. Os proximos nós a serem visitados 3, 6 e 5 não adicionam ninguem pois todos já foram adicionados na fila. Obtemos assim o seguinte sequenciamento de Ferramentas $F_\phi = (1,4,2,3,6,5)$.

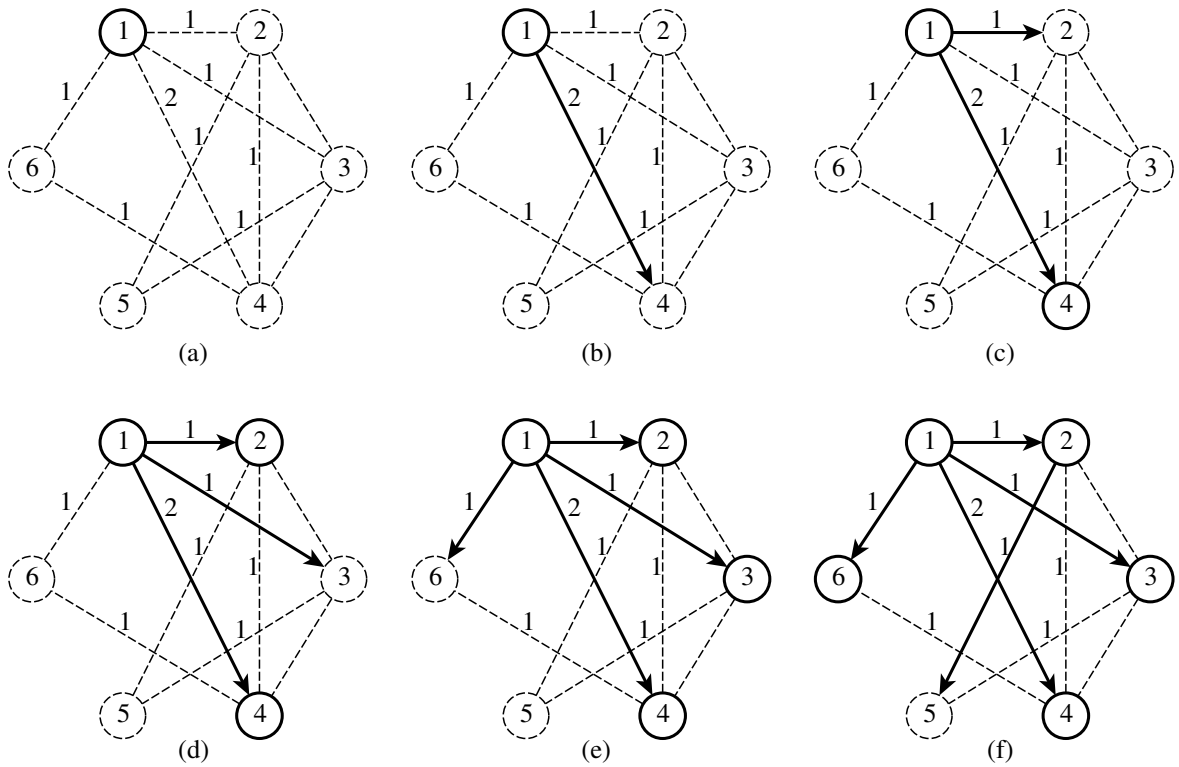

Figura 5.1: Execução do algoritmo 9 de sequenciamento de tarefas na instância apresentada na tabela 4.

5.8.2 Sequenciamento das Tarefas

A partir da sequência de ferramentas geradas na fase anterior obtemos *insights* em como deve ser feito o sequenciamento das tarefas. Como as ferramentas semelhantes tendem a ficar próximas, será feito uma varredura nesta sequência e as ferramentas já visitadas se tornam disponíveis para utilização.

De acordo com que as ferramentas se tornem disponíveis, as tarefas se tornam aptas a serem executadas. Foram definidos dois critérios de desempate quando mais de uma tarefa está apta a ser executada, estas são: Quando nenhuma tarefa tiver sido executada ainda, escolha a tarefa com o maior número de ferramentas e quando alguma tarefa já tiver sido executada, escolha a tarefa cuja execução adiciona o menor número de trocas, após estes critérios serem aplicados ainda persistindo um empate escolha arbitrariamente.

Algoritmo 10: Sequenciamento da Tarefas

Dados: Uma instância do MTSP, Sequencia F_ϕ de Ferramentas
$FerramentasDisp \leftarrow \emptyset$;
$T_\phi \leftarrow ()$;
para cada $x \in F_\phi$ **faça**
  $FerramentasDisp \leftarrow FerramentasDisp \cup x$;
  $TarefasAptas \leftarrow \{i \in J \mid T_i \subset FerramentasDisp \text{ e } i \notin T_\phi\}$;
  enquanto $TarefasAptas \neq \emptyset$ **faça**
    se $T_\phi \leftarrow \emptyset$ **então**
      $k \leftarrow max(|F_j|), j \in TarefasApta$;
    senão
      $k \leftarrow min(Custo(T_\phi, j)), j \in TarefasApta$;
    fim
    $T_\phi \leftarrow T_\phi \cup k$;
  fim
fim
retorna T_ϕ;

5.8.3 Fase de Refinamento

Após a fase de Sequenciamento das tarefas obtem-se uma solução inicial. Agora vamos refiná-la em busca de uma solução melhor. Para isso utilizamos a metaheurística *Iterated Local Search* (ILS), utilizando a solução gerada anteriormente como sendo a solução inicial. A cada iteração do ILS é realizada uma perturbação da solução atual, em seguida, é utilizado um mecanismo de busca local para o encontrar o ótimo local, este podendo se tornar a nova solução atual se atender a um critério de aceitação.

Antes de descrever o ILS, serão descritos alguns componentes críticos do mesmo, sendo eles a perturbação e a busca local.

A perturbação de uma solução é feita através de sucessivas trocas de posições entre duas tarefas aleatórias. O número de vezes que estas trocas ocorrerão é definido como uma porcentagem do número de tarefas do problema. Esta porcentagem é definida pelo valor α. Esta perturbação é uma forma de escapar de ótimos locais, pois se aplicarmos a busca local repetidas vezes a solução ficara presa em um lugar sem conseguir melhorá-la. Então após aplicar a busca local, a solução é modificada aleatóriamente para que seja gerado uma novo ponto no espaço de busca intuito de aumentar o espaço buscado.

Algoritmo 11: Perturbação de uma Solução T_ϕ

```
Dados: Sequencia T_φ de tarefas, α
numPertubacoes ← |T_φ| * α;
iter ← 0;
enquanto iter ≤ numPertubacoes faça
    i ← random(1, |T_φ|) ;
    j ← random(1, |T_φ|) ;
    enquanto i = j faça
        j ← random(1, |T_φ|) ;
    fim
    trocaPosicao(i, j);
    iter ← iter + 1;
fim
retorna T_φ;
```

Considere uma solução $T_\phi = (1,2,3,4,5)$ que será utilizada no processo de perturbação descrito anteriormente, considere também um valor de $\alpha = 0.2$, como o número de tarefas de T_ϕ é igual a 5 então o loop do algoritmo de perturbação é executado uma única vez, gerando apenas um par de índices aleatórios i e j para que as tarefas que estão na i-ésima e j-ésima posição sejam invertidas. Por exemplo, se i e j forem, respectivamente, dois e quatro a nova sequência será $T_\phi = (1,4,3,2,5)$, essa troca independe da função objetivo, pois serve para diversificar as soluções geradas durante o processo de refinamento.

Crama et al. (1994) apresentou uma definição de *1-Blocks*, um conjunto de entradas consecutivas de uma linha de uma matriz com o valor igual a 1. Uma troca de ferramenta é caracterizada pela existência de dois *1-Blocks* diferentes em uma mesma linha. Com esta mesma definição foi proposta uma busca local que tem como objetivo diminuir o número de *1-Blocks*.

A solução do MTSP pode ser apresentanda como uma matriz binária na qual as linhas representam se uma ferramenta i esta presente na máquina no momento em que a tarefa j é executada. Essa matriz de solução é iterativamente examinada em procura por dois *1-Blocks* em uma mesma linha e então, tenta-se juntar os dois movendo as colunas relacionadas ao primeiro *1-Block* para antes ou depois das colunas relacionadas ao segundo

1-Block ou mantê-los como estão, o que for melhor em relação ao número de trocas correspondente.

Algoritmo 12: Busca local de uma Solução T_ϕ

Dados: Matriz M de Tarefas Por Ferramentas
para cada *linha l da matriz M* **faça**
  Examine a linha l da esquerda para a direita até achar um primeiro *1-Block j*;
  enquanto *Existir um proximo 1-Block k* **faça**
    De acordo com o número de trocas, escolha entre as seguintes permutações das colunas de M:
    $(j...k)$, $(...jk)$ ou $(...kj)$;
    $j \leftarrow k$;
  fim
  Examine a linha l da direita para a esquerda até achar um primeiro *1-Block j*;
  enquanto *Existir um proximo 1-Block k* **faça**
    De acordo com o número de trocas, escolha entre as seguintes permutações das colunas de M:
    $(j...k)$, $(kj...)$ ou $(jk...)$;
    $j \leftarrow k$;
  fim
fim
retorna M;

Apresentaremos agora um exemplo da execução da busca local, considere o exemplo de uma instância do MTSP apresentando na tabela 4 e uma solução inicial $T_\phi = (1,2,3,4,5)$ temos que a tabela binária de solução do MTSP é mostrada na tabela 5.

Tabela 5: *Tabela de binária de solução da instância da tabela 4.*

Tarefas	1	2	3	4	5
	1	1	0	0	1
	1	0	0	1	0
	0	1	1	1	0
	1	0	1	0	1
	0	0	0	1	0
	0	0	0	0	1

Temos que após a aplicação do algoritmo 1: KTNS, obtemos que o número de trocas de $T_\phi = 9$. A partir da primeira linha iremos demonstrar a execução do algoritmo 12 de busca local. Em um primeiro instante o algoritmo examina cada linha da matriz da esquerda para direita e o primeiro passo do algoritmo é encontrar o primeiro *1-block j*, este é formado pelas colunas 1 e 2, em seguida devemos achar um segundo *1-Block k*, já este é composto apenas pela coluna 5. Em seguida é avaliado as possivéis mudanças de posição do j. Sabemos que a sequência inicial $(1,2,3,4,5)$ possui um número de trocas igual a nove, as duas outras possíveis soluções são $(3,4,1,2,5)$ e $(3,4,5,1,2)$ a função objetivo destas duas soluções é oito e nove trocas respectivamente, como a solução $(3,4,1,2,5)$ melhora a solução esta será a nova solução T_ϕ e o algoritmo continua examinando cada uma das linhas e depois o processo é repetido examinando as linhas da matriz da direita para a esquerda.

É apresentado na Tabela 6 a matriz de solução após a aplicação do algoritmo 1: KTNS.

Tabela 6: *Matriz solução da nova sequência T_ϕ com 8 trocas.*

Tarefas	3	4	1	2	5
	0	0	1	1	1
	1	1	1	0	0
	1	1	0	1	0
	1	0	1	1	1
	0	1	0	0	0
	0	0	0	0	1

No intuito de aumentar a área de busca, foi incorporado um segundo método de busca local. Para isso foi utilizado o método 2-opt parcial, uma variação do 2-opt em que o espaço de busca é aleatoriamente reduzido em uma proporção δ, a fim de se obter uma redução no tempo de execução.

Este segundo algoritmo é parecido com a perturbação, porém só serão aceita trocas que melhoram a função objetivo e a partir dos vizinhos, soluções que possuem uma troca de distância da solução atual, o que possuir a menor função objetivo será escolhida para recomeçar o processo a partir do mesmo.

Juntando estes componentes obtemos o ILS, uma meta-heurística simples porém muito poderosa. Por ser composta de buscas locais e um mecanismo para escapar de ótimos locais que podem estagnar a busca, esta meta-heurística consegue encontrar boas soluções quando seus componentes possuem uma boa sinergia entre eles. A seguir será apresentado o algoritmo ILS.

O critério de aceitação utilizado é verificar se a solução obtida tem um número de trocas menor do que a melhor solução, o número de trocas é definido pelo algoritmo KTNS, como visto anteriormente.

Algoritmo 13: ILS

Dados: Matriz M de Tarefas Por Ferramentas, α, δ , numIteracoes
$F_\phi \leftarrow$ Sequenciamento de Ferramentas(M);
$T_\phi \leftarrow$ Sequenciamento de Tarefas(F_ϕ);
Aplique a busca local de redução de *1-Blocks* em T_ϕ;
$iter \leftarrow 0$;
enquanto $iter \leq numIteracoes$ **faça**
  $T'_\phi \leftarrow T_\phi$;
  Perturbe a solução T'_ϕ como visto anteriormente usando α como parâmetro;
  Aplique a busca local 2-opt Parcial em T'_ϕ usando δ como parâmetro;
  Aplique a busca local de redução de *1-Blocks* em T'_ϕ;
  se T'_ϕ *atender o critério de aceitação* **então**
    $T_\phi \leftarrow T'_\phi$;
  fim
fim
retorna T_ϕ;

Capítulo 6

Experimentos Computacionais

O cronograma do projeto foi cumprido rigorosamente. Conforme apresentado no Capítulo 3, uma revisão sistemática da literatura foi realizada.

Um relatório técnico foi gerado e será aproveitado em futuras publicações relacionadas ao tema ao longo da execução do projeto.

Além de fazer uma revisão da literatura também foi implementado os métodos considerados estado da arte para o MTSP, estes foram utilizados para obter um conhecimento prático do problema e também para comparar os métodos propostos durante este trabalho. Em experimentos iniciais, os métodos propostos neste trabalho atinge resultados melhores ou iguais aos melhores resultados da literatura.

Os métodos apresentados e propostos foram implementados utilizando a linguagem C++ e foi compilado com as *flags* -O3 e std=c=+11. Os testes foram realizados em uma máquina com processador *Intel i5 Quad Core* de 3.2GHz e 8 GB de RAM, utilizando o sistema operacional Ubuntu 15.10. Estes experimentos foram feitos para garantir e proporcionar comparações confiáveis com os métodos propostos.

Os experimentos conduzidos utilizaram os exemplares disponibilizados por: Crama et al. (1994), Chaves et al. (2012) e Catanzaro et al. (2015).

6.1 Instâncias de Chaves et al. (2012)

Os exemplares introduzidos por Chaves et al. (2012) foram divididos em 5 grupos.Na tabela é apresetado as caractéristicas de cada um dos grupos. As Colunsa Números de Tarefas, Número de Ferramentas e Capacidade mostram o intervalo que estes valores assumem em seu respectivo grupo.

Tabela 7: *Caractéristicas das instâncias de Chaves et al. (2012)*

Grupo	Número de Tarefas	Número de Ferramentas	Capacidade	Quantidade de Exemplares
A	[8, 8]	[15, 25]	[5, 20]	340
B	[9, 9]	[15, 25]	[5, 20]	330
C	[15, 15]	[15, 25]	[5, 20]	340
D	[20, 25]	[15, 25]	[5, 20]	260
E	[10, 15]	[10, 20]	[4, 12]	80

Nas quatro tabelas seguintes o significado de N, M e C são respectivamente o número de tarefas, o número de ferramentas e a capacidade da máquina. Para o Grupo A, B e C, o método proposto atingiu o ótimo de todas instâncias com um tempo inferior ao CS+BRKGA de Chaves et al. (2016) como apresentado nas tabela 8, 9 e 10. Já para o grupo D, o método proposto atingiu todos os ótimos, melhorando ainda os resultados obtidos pelo CS+BRKGA de Chaves et al. (2016) com um tempo menor do que o mesmo.

Tabela 8: *Resultados do Grupo A*

			CS+BRKGA		Proposto	
N	M	C	Resultado	Tempo	Resultado	Tempo
8	15	5	17.00	0.98	17.00	0.08
8	15	10	16.83	1.03	16.83	0.10
8	20	5	21.80	1.30	21.80	0.08
8	20	10	23.07	1.49	23.07	0.17
8	20	15	22.08	1.34	22.08	0.14
8	25	5	25.10	1.60	25.10	0.06
8	25	10	28.20	1.70	28.20	0.19
8	25	15	27.95	1.62	27.95	0.23
8	25	20	26.61	1.47	26.61	0.17

Tabela 9: *Resultados do Grupo B*

			CS+BRKGA		Proposto	
N	M	C	Resultado	Tempo	Resultado	Tempo
9	15	5	17.20	2.72	17.20	0.11
9	15	10	17.37	3.15	17.37	0.16
9	20	5	22.40	3.13	22.40	0.13
9	20	10	24.17	3.92	24.17	0.26
9	20	15	22.60	3.99	22.60	0.24
9	25	5	25.40	4.08	25.40	0.10
9	25	10	28.77	5.08	28.77	0.27
9	25	15	29.74	5.10	29.74	0.43
9	25	20	27.19	5.26	27.19	0.29

Tabela 10: *Resultados do Grupo C*

			CS+BRKGA		Proposto	
n	m	c	Resultado	Tempo	Resultado	Tempo
15	15	5	21.60	0.21	21.60	0.76
15	15	10	19.80	0.09	19.80	1.20
15	20	5	25.60	0.51	25.60	0.91
15	20	10	28.33	0.32	28.33	1.84
15	20	15	25.52	0.21	25.52	2.00
15	25	5	32.50	0.38	32.50	0.92
15	25	10	35.07	0.41	35.07	2.20
15	25	15	34.07	0.36	34.07	3.33
15	25	20	29.66	0.27	29.66	2.47

Tabela 11: *Resultados do Grupo D*

			CS+BRKGA		Proposto	
n	m	c	Resultado	Tempo	Resultado	Tempo
20	15	5	26.10	10.78	25.90	2.10
20	15	10	18.20	12.34	18.20	3.39
20	20	5	29.30	14.84	29.30	2.61
20	20	10	20.60	16.08	20.60	3.93
20	20	15	21.67	24.66	21.67	4.56
20	25	5	35.10	19.16	35.10	2.69
20	25	10	25.40	21.49	25.40	4.75
20	25	15	36.25	28.11	36.25	9.59
20	25	20	26.15	35.53	26.15	4.75
25	15	10	15.90	21.14	15.90	6.56
25	20	10	21.60	27.48	21.60	12.96
25	20	15	22.60	25.66	22.60	13.41
25	25	10	26.60	36.88	26.60	16.27
25	25	15	25.00	54.70	25.00	11.35
25	25	20	25.50	66.10	25.50	10.31

Os experimentos computacionais foram apresentandos e o método proposto obteve em sua maioria resultados iguais ou melhores do que o estado da arte para este problema. Além disso o tempo computacional deste é baixa comparada com outros métodos que obtém soluções semelhantes, demostrando que este novo método, além de robusto, é rápido. Os parametros do método foram definidos através do programa de *tunagem* de paramêtro automático *iRace*.

Capítulo 7

Conclusão

O Problema de Minimização de Trocas de Ferramentas é um problema NP-Díficil que possui aplicabilidade na área industrial. Neste trabalho foi proposto uma heurística para o MTSP baseada em três etapas: Sequenciamento das Ferramentas, Sequenciamento das Tarefas e Fase de Refinamento. Esta heurística se mostrou robusta e obteve resultados promissores. Em trabalhos futuros poderão ser abordados novos algoritmos heurísticos e exatos, visto que este segundo ainda não foi muito abordados pela literatura.

Os produtos futuros deste projeto de pesquisa serão também avaliados por pares (comitês científicos e corpos editoriais) na forma de artigos submetidos para periódicos nacionais e internacionais. Um artigo científico será submetido para uma revista indexada com reconhecida relevância na área e também será submetido ao prêmio de iniciação científica promovido pela Sociedade Brasileira de Pesquisa Operacional. Como resultados adicionais esperados temos a participação em congressos e conferências onde os temas pesquisados serão discutidos com a comunidade científica.

Referências Bibliográficas

David Adjiashvili, Sandro Bosio, and Kevin Zemmer. Minimizing the number of switch instances on a flexible machine in polynomial time. *Operations Research Letters*, 43(3):317–322, May 2015. ISSN 0167-6377. doi: 10.1016/j.orl.2015.04.001. URL http://www.sciencedirect.com/science/article/pii/S0167637715000498.

M. A. Al-Fawzan and K. S. Al-Sultan. A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. *Computers & Industrial Engineering*, 44(1):35–47, January 2003. ISSN 0360-8352. doi: 10.1016/S0360-8352(02)00183-3. WOS:000179686800003.

Jhon Edgar Amaya, Carlos Cotta, and Antonio J. Fernández. A Memetic Algorithm for the Tool Switching Problem. In María J. Blesa, Christian Blum, Carlos Cotta, Antonio J. Fernández, José E. Gallardo, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, number 5296 in Lecture Notes in Computer Science, pages 190–202. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88438-5, 978-3-540-88439-2. URL http://link.springer.com/chapter/10.1007/978-3-540-88439-2_14.

Jhon Edgar Amaya, Carlos Cotta, and Antonio J. Fernández-Leiva. Memetic cooperative models for the tool switching problem. *Memetic Computing*, 3(3):199–216, June 2011. ISSN 1865-9284, 1865-9292. doi: 10.1007/s12293-011-0059-6. URL http://link.springer.com/article/10.1007/s12293-011-0059-6.

Jhon Edgar Amaya, Carlos Cotta, and Antonio J. Fernández-Leiva. Cross entropy-based memetic algorithms: An application study over the tool switching problem. *International Journal of Computational Intelligence Systems*, 6(3):559–584, May 2013. ISSN 1875-6891. doi: 10.1080/18756891.2013.792542. URL http://dx.doi.org/10.1080/18756891.2013.792542.

JONATHAN F Bard. A Heuristic for Minimizing the Number of Tool Switches on a Flexible Machine. *IIE Transactions*, 20(4):382–391, December 1988. ISSN 0740-817X. doi: 10.1080/07408178808966195. URL http://dx.doi.org/10.1080/07408178808966195.

Daniele Catanzaro, Luis Gouveia, and Martine Labbé. Improved integer linear programming formulations for the job Sequencing and tool Switching Problem. *European Journal of Operational Research*, 2015. ISSN 0377-2217. doi: 10.1016/j.ejor.2015.02.018. URL http://www.sciencedirect.com/science/article/pii/S0377221715001198.

A. A. Chaves, L. A. N. Lorena, E. L. F. Senne, and M. G. C. Resende. Hybrid method with CS and BRKGA applied to the minimization of tool switches problem. *Computers & Operations Research*, 67:174–183, March 2016. ISSN 0305-0548. doi: 10.1016/j.cor.2015.10.009. URL http://www.sciencedirect.com/science/article/pii/S0305054815002403.

Antônio Augusto Chaves, Horacio Hideki Yanasse, and Edson Luiz França Senne. Uma nova heurística para o problema de minimização de trocas de ferramentas. *Gestão & Produção*, 19(1):17–30, 2012. ISSN 0104-530X. doi: 10.1590/S0104-530X2012000100002. SCIELO:S0104-530X2012000100002.

Yves Crama and Ellen Talloen. The Tool Switching Problem Revisited. *European Journal of Operational Research*, 2007.

Yves Crama, Antoon W. J. Kolen, Alwin G. Oerlemans, and Frits C. R. Spieksma. Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6(1):33–54, January 1994. ISSN 0920-6299, 1572-9370. doi: 10.1007/BF01324874. URL http://link.springer.com/article/10.1007/BF01324874.

Y. Fathi and K. W. Barnette. Heuristic procedures for the parallel machine problem with tool switches. *International Journal of Production Research*, 40(1):151–164, January 2002. ISSN 0020-7543. doi: 10.1080/00207540110076115. URL http://dx.doi.org/10.1080/00207540110076115.

Alain Hertz, Gilbert Laporte, Michel Mittaz, and Kathryn E. Stecke. Heuristics for minimizing tool switches when scheduling part types on a flexible machine. *IIE Transactions*, 30(8):689–694, August 1998. ISSN 0740-817X. doi: 10.1080/07408179808966514. URL http://dx.doi.org/10.1080/07408179808966514.

G. Laporte, J. J. Salazar-Gonzalez, and F. Semet. Exact algorithms for the job sequencing and tool switching problem. *Iie Transactions*, 36(1):37–45, January 2004. ISSN 0740-817X. doi: 10.1080/07408170490257871. WOS:000220824200005.

S. Zamiri Marvizadeh and F. F. Choobineh. Reducing the number of setups for CNC punch presses. *Omega*, 41(2):226–235, April 2013. ISSN 0305-0483. doi: 10.1016/j.omega.2012.06.001. URL http://www.sciencedirect.com/science/article/pii/S0305048312000965.

B. Matzliach and M. Tzur. The online tool switching problem with non-uniform tool size. *International Journal of Production Research*, 36(12):3407–3420, December 1998. ISSN 0020-7543. doi: 10.1080/002075498192120. WOS:000077598600009.

Csaba Raduly-Baka and Olli S. Nevalainen. The modular tool switching problem. *European Journal of Operational Research*, 242(1):100–106, April 2015. ISSN 0377-2217. doi: 10.1016/j.ejor.2014.09.052. URL http://www.sciencedirect.com/science/article/pii/S0377221714007917.

Edson Luiz França [UNESP Senne, Horacio Hideki Yanasse, M. Jha, C. Long, N. Mastorakis, and C. A. Bulucea. Beam Search Algorithms for Minimizing Tool Switches on a Flexible Manufacturing System. *Proceedings of The 11th Wseas International Conference on Mathematical and Computational Methods In Science and Engineering (macmese '09)*, pages 68–72, January 2009. URL http://base.repositorio.unesp.br/handle/11449/9362.

R. Shirazi and G. D. M. Frizelle. Minimizing the number of tool switches on a flexible machine: an empirical study. *International Journal of Production Research*, 39(15):3547–3560, October 2001. ISSN 0020-7543. doi: 10.1080/00207540110060888. WOS:000171519700014.

Christopher S. Tang and Eric V. Denardo. Models Arising from a Flexible Manufacturing Machine, Part I: Minimization of the Number of Tool Switches. *Operations Research*, 36(5):767–777, October 1988. ISSN 0030-364X. doi: 10.1287/opre.36.5.767. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.36.5.767.

Horacio Hideki Yanasse, Rita de Cássia Meneses Rodrigues, and Edson Luiz França Senne. Um algoritmo enumerativo baseado em ordenamento parcial para resolução do problema de minimização de trocas de ferramentas. *Gestão & Produção*, 16(3):370–381, 2009. ISSN 0104-530X. doi: 10.1590/S0104-530X2009000300005. SCIELO:S0104-530X2009000300005.

Bing-Hai Zhou, Li-Feng Xi, and Yong-Shang Cao. 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, February 2004. ISSN 0268-3768, 1433-3015. doi: 10.1007/s00170-003-1925-2. URL http://link.springer.com/article/10.1007/s00170-003-1925-2.