UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE
DEPARTAMENTO DE COMPUTAÇÃO

DIÊGO DOS SANTOS GUIMARÃES

HEURÍSTICA HÍBRIDA GRASP E *SIMULATED ANNEALING* PARA O PROBLEMA DE PROGRAMAÇÃO DE TAREFAS EM MÁQUINAS PARALELAS COM TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQUÊNCIA E DE RECURSOS

ALEGRE - ES
2018

DIÊGO DOS SANTOS GUIMARÃES

HEURÍSTICA HÍBRIDA GRASP E *SIMULATED ANNEALING* PARA O PROBLEMA DE PROGRAMAÇÃO DE TAREFAS EM MÁQUINAS PARALELAS COM TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQUÊNCIA E DE RECURSOS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro De Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. M.Sc. Edmar Hell Kampke.

ALEGRE - ES

2018

DIÊGO DOS SANTOS GUIMARÃES

HEURÍSTICA HÍBRIDA GRASP E *SIMULATED ANNEALING* PARA O PROBLEMA DE PROGRAMAÇÃO DE TAREFAS EM MÁQUINAS PARALELAS COM TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQUÊNCIA E DE RECURSOS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro De Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em 22 de junho de 2018.

COMISSÃO EXAMINADORA

Prof. M.Sc. Edmar Hell Kampke
Universidade Federal do Espírito Santo
Orientador

Prof. M.Sc. Jacson Rodrigues Correia da Silva
Universidade Federal do Espírito Santo

Prof. M.Sc. Valéria Alves da Silva
Universidade Federal do Espírito Santo

“O fracasso é somente a oportunidade de começar de novo, de forma mais inteligente”

Henry Ford

AGRADECIMENTO

Primeiramente a Deus que permitiu que tudo isso acontecesse, ao longo da minha vida, e não somente nestes anos como universitário, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

Agradeço aos meu pais Ângela Maria dos Santos Guimarães e Sebastião Carlos Morisco Guimarães que sempre me apoiaram e me incentivaram, sempre me auxiliando nas horas mais difíceis.

A minha irmã Dislaini dos Santos Guimarães, que sempre esteve presente em todos os momentos, sendo além de irmã uma grande amiga.

A minha namora Estefânia Pecegueiro Prudêncio, que sempre esteve ao meu lado antes, durante e depois da faculdade.

Aos meus amigos que conheci na UFES, aos amigos do Social das Cachaças, principalmente alguns que me ajudaram bastante nessa caminhada com várias noites viradas estudando, Joabe, Paula, Marilia, Robson, Melissa, Leandro, Josimar e em especial um amigo que não se encontra mais entre nós mas que me acolheu em vitória para que eu pudesse iniciar minha vida profissional na área, obrigado Yuri de onde você estiver.

Ao professor Edmar Hell Kampke pela orientação, paciência, apoio e confiança.

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

RESUMO

O sequenciamento de tarefas em máquinas paralelas é um problema enfrentado em todos os tipos e níveis de indústrias. Este trabalho aborda o problema de programação de tarefas em máquinas paralelas com tempos de preparação dependentes da sequência e de recursos. O problema tem como característica o tempo de preparação, que é determinado pelo sequenciamento das tarefas nas máquinas, e também pela quantidade de recursos associados, que varia entre um valor mínimo e um máximo, e tem como objetivo a redução do tempo total de conclusão das tarefas (*makespan)*. Devido à complexidade NP-Difícil do problema, inicialmente propõe-se implementar um algoritmo baseado na meta-heurística híbrida GRASP e *Simulated Annealing*. O GRASP utiliza como método construtivo uma variação do algoritmo DJASA, e como busca local a meta-heurística *Simulated Annealing*. Apresentaram resultados satisfatórios, pois em alguns casos foram superiores aos resultados encontrados na literatura, comprovando a possibilidade do uso destes algoritmos na resolução de problemas existentes em indústrias.

Palavras-chave: Máquinas Paralelas, *Makespan,* DJASA, GRASP, *Simulated Annealing*

ABSTRACT

The sequencing of tasks in parallel machines is a problem faced in all types and levels of industries. This work addresses the problem of scheduling tasks on parallel machines with sequence and resource-dependent preparation times. The problem is characterized by the preparation time, which is determined by the sequencing of the tasks in the machines, and also by the amount of associated resources, which varies from a minimum to a maximum, and aims to reduce the total time of completion of the tasks (makespan). Due to the NP-Difficult complexity of the problem, initially it is proposed to implement an algorithm based on the hybrid meta-heuristic GRASP and Simulated Annealing. The GRASP uses as a constructive method a variation of the DJASA algorithm, and as local search the meta-heuristic Simulated Annealing. They presented satisfactory results, since in some cases they were superior to the results found in the literature, proving the possibility of the use of these algorithms in the resolution of existing problems in industries.

Keywords: Parallel Machines, *Makespan*, DJASA, GRASP, *Simulated Annealing*

LISTA DE FIGURAS

Figura 1 - Exemplo de aplicação do PTMP-SR 11
Figura 2 - Relação entre *Setup Time* S_{ijk} e os Recursos R_{ijk}. 12

SUMÁRIO

1 INTRODUÇÃO 9

1.1 O PROBLEMA E SUA IMPORTÂNCIA 10

1.2 OBJETIVOS 13

1.2.1 Objetivo Geral 13

1.2.2 Objetivos Específicos 13

1.3 HIPÓTESE 13

1.4 ESTRUTURA DO TRABALHO 13

2 REVISÃO DE LITERATURA 14

3 METODOLOGIA 16

3.1 MODELAGEM MATEMÁTICA 16

3.2 *DYNAMIC JOB ASSIGNMENT WITH SETUPS RESOURCE ASSIGNMENT* (DJASA) 17

3.3 GRASP 18

3.3.1 Construção da Solução 19

3.3.2 Busca Local 19

4 RESULTADOS E DISCUSSÃO 21

5 CONCLUSÕES 28

6 REFERÊNCIAS 29

1 INTRODUÇÃO

A Pesquisa Operacional (PO) são métodos científicos utilizados para resolver diferentes problemas de tomada de decisão, com o objetivo de aproveitar melhor recursos escassos. A PO tem o comportamento de auxiliar nos processos de tomada de decisões em quaisquer setores e níveis da economia, visando à satisfação do usuário. Dessa forma, a PO é uma área da Ciência da Computação que tem a como interseção outras áreas do conhecimento humano.

Dentre os problemas com resolução pela PO destacam-se a Otimização Combinatória. A Otimização Combinatória a tomada de decisões no caso de problemas discretos encontrados em diversas áreas, entre elas estão, problemas de programação e planejamento de produção, roteamento de veículos, problemas de localização, redes de telecomunicação, entre outras. Em muitos destes problemas surgem frequentemente vários critérios de desempenho que conflitam entre si (funções objetivos). Por exemplo, a execução do plano de produção do controle de chão de fábrica em é necessário programar a produção em cada centro de trabalho de forma a satisfazer objetivos globais da empresa (KAMPKE, 2010).

Como apontado por VOLMANN et al. (1988), encontra-se três objetivos básicos em uma programação de tarefas na produção. O primeiro objetivo está associado com datas de entrega: em que basicamente, não é desejado o atraso dessas datas, e quando a relevância do custo de estoque, esforça-se para evitar que as tarefas sejam finalizadas antes das datas. O segundo objetivo está associado com o tempo de fluidez das tarefas no chão de fábrica: deseja-se que esse tempo seja pequeno, ou proporcional, e o estoque em processamento seja mínimo. O terceiro objetivo envolve utilizar de centros de trabalho: almeja-se maximizar a utilização dos equipamentos e da mão de obra.

Uma ampla porção dos problemas de otimização combinatória não detêm de um método eficiente para serem solucionados, fazem parte desta classe os problemas denominados como NP-difíceis. A característica fundamental deste tipo de problema é que embora nunca ter sido evidenciado a inexistência, nenhum algoritmo eficiente foi encontrado para estes problemas. Um método interessante de resolução destes tipos de problema é utilizar de algoritmos não exatos ou heurísticos, que buscam soluções próximas utilizando de baixo tempo de processamento.

O problema consiste em alocar *n* tarefas em *m* máquinas, de forma a otimizar um critério definido, como exemplo o tempo de conclusão máximo de todas as tarefas, também conhecido como *makespan.*

Deste modo, dado o valor do problema e a dificuldade de obter soluções com tempo computacional aceitável e de qualidade, fomenta-se a motivação para a pesquisa com base na utilização de métodos heurísticos.

1.1 O PROBLEMA E SUA IMPORTÂNCIA

O Problema de Sequenciamento de Tarefas em Máquinas Paralelas (PSTMP) está presente em indústrias de várias áreas de atuações, este problema tem grande importância, sobretudo para as indústrias de manufatura, pois ocorre, por exemplo, na fabricação de cerâmicas, onde existem diferentes máquinas de polimento. Cada máquina inicia o polimento com tipos diferentes de cerâmica, e ao finalizar a máquina precisa de limpeza e preparação para o polimento de outros tipos de cerâmica. O tempo para limpar e preparar varia de acordo com a máquina, com o tipo da cerâmica e com a quantidade de recursos disponíveis para utilização, no caso, pode ser quantidade de pessoas que serão utilizadas para realizar a limpeza e ajustar as máquinas. Ocorrendo também em outros ramos de atuação como na indústrias têxtil, de alvenaria, automobilística e entre outras

A solução do PSTMP proporciona a redução de custos de operação e manutenção de equipamentos e prolongamento do tempo da vida útil das máquinas paralelas, acarretando em um menor desgaste dos equipamentos.

O PSTMP tem sido abordado com frequência nos últimos anos (KAMPKE, 2010). Ruiz e Andrés (2007) comprovaram que o PSTMP com tempos de preparação (*setup times*) é um problema do tipo NP-Difícil.

O Problema de Sequenciamento de Tarefas em Máquinas Paralelas com Tempos de Preparação Dependentes da Sequência e de Recursos (PSTMP-SR), pode ser reduzido a um simples caso do PSTMP, e dessa forma também ser considerado NP-Difícil. Nesse caso, a busca pelas soluções ótimas com emprego de métodos matemáticos exatos pode ser inviável computacionalmente (COTA *et. al.*, 2014). Devido a sua complexidade, uma alternativa para resolver o problema é a

utilização de heurísticas, com o objetivo de obter soluções de qualidade com baixo custo computacional (KAMPKE, 2010).

O Problema de Sequenciamento de Tarefas em Máquinas Paralelas pode ser divididos em três categorias de acordo com o tipo de máquina: Uniformes, idênticas e não relacionadas. Máquinas uniformes são quando a variação do tempo de processamento de uma tarefa é proporcional entre as máquinas, Máquinas idênticas são quando o tempo de processamento de uma tarefa é o mesmo para todas, Máquinas não relacionadas o tempo de processamento de uma tarefa é diferente em cada máquina (HADDAD, 2012).

Portanto o PSTMP com máquinas não relacionadas é o que possui maior complexidade e assim uma forma eficiente de resolução é utilizar meta-heurísticas para solucionar o problema, reduzindo o *makespan*. Pois encontrar soluções ótimas ou aceitáveis, manualmente, para o problema, é praticamente impossível, devido a sua complexidade.

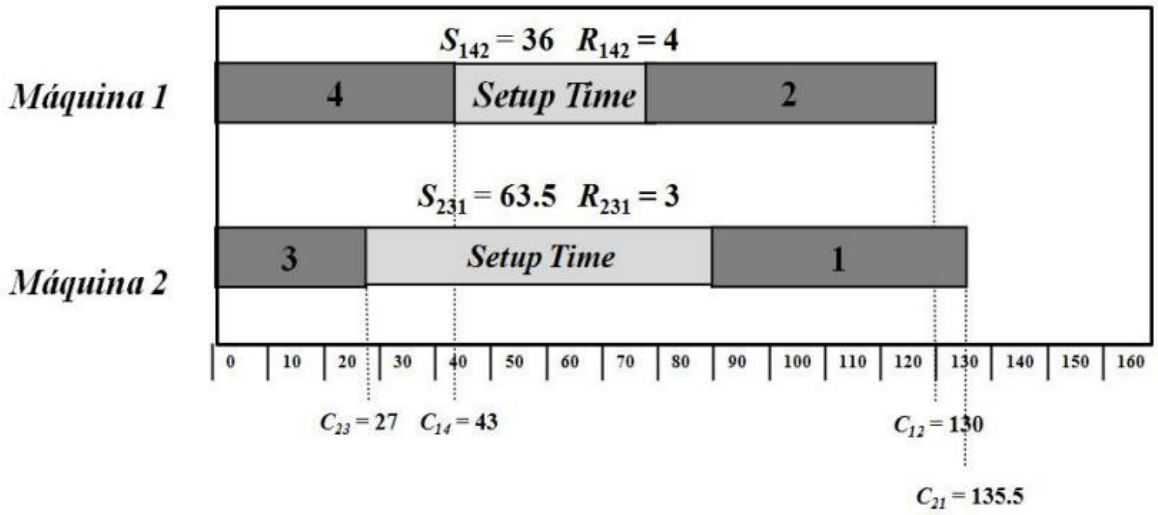

Figura 1 - Exemplo de aplicação do PTMP-SR
Fonte: Kampke (2010).

A Figura 1 apresenta um exemplo de solução do PSTMP-SR em que se observa que cada tarefa pode ser processada que qualquer máquina. O PSTMP-SR considera que *n* tarefas que serão processadas em *m* máquinas paralelas, não relacionadas. Cada máquina pode processar apenas uma tarefa por vez e cada tarefa pode ser processada em apenas uma máquina. Cada tarefa *i* possui, um tempo determinado de processamento P_{ij} na máquina *i*. Assim pode se observar que tempo

de conclusão do processamento da tarefa 4 na máquina 1, denotado por $C_{14} = 43$.

O *setup time*, ou tempo de preparação, é o tempo necessário entre a execução de duas tarefas, em uma máquina, para limpeza, realocação, posicionamento, entre outras coisas. Como pode ser visto na Figura 1, o *setup time* entre as tarefas 4 e 2 na máquina 1, é denotado por $S_{142} = 36$. Vale ressaltar que o *setup time* é definido de acordo com a quantidade de recurso utilizada, que nesse caso $R_{142} = 4$, ou seja, entre o processamento das tarefas 4 e 2 na máquina 1, são utilizados 4 recursos.

O *setup time* S_{ijk} é definido a partir de uma relação inversamente proporcional com os recursos R_{ijk}, de forma que quanto maior for o número de recursos usados, como por exemplo a mão-de-obra, menor será o *setup time*, ou vice-versa.

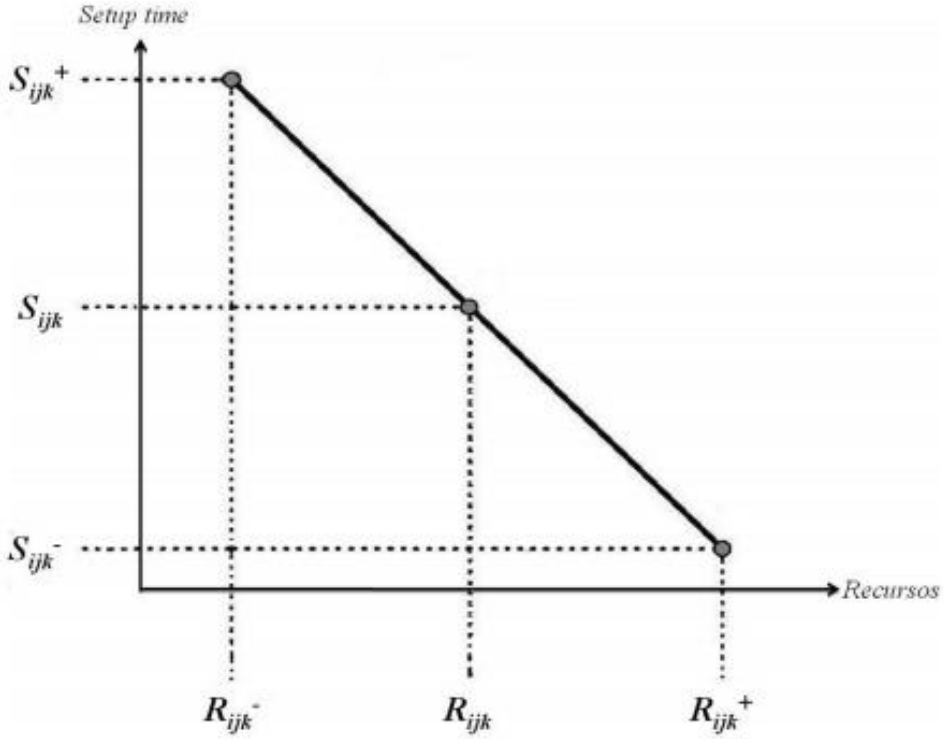

Figura 2 - Relação entre *Setup Time* S_{ijk} e os Recursos R_{ijk}.
Fonte: Kampke (2010).

A Figura 2 mostra a relação entre *setup time* e os recursos. Sendo que $R_{ijk}+$, $R_{ijk}-$, $S_{ijk}+$, $S_{ijk}-$ são valores conhecidos para cada combinação de valores de *i*, *j* e *k*.

O objetivo do PSTMP-SR é alocar todas as tarefas nas máquinas de tal forma que o tempo de conclusão das tarefas e a quantidade de recursos usados seja a melhor possível.

Assim sendo, pode-se afirmar que PSTMP-SR possui grande relevância prática e alta complexidade, por isso será abordado nesse trabalho.

1.2 OBJETIVOS

A seguir será abordada uma breve descrição dos objetivos gerais e específicos relacionados a este trabalho.

1.2.1 Objetivo Geral

O objetivo deste trabalho foi aplicar um método híbrido *Greedy Randomized Adaptive Search Procedure* (GRASP) com uma variação do método construtivo *Dynamic Job Assignment With Setups Resource Assignment* (DJASA) e que usa na etapa de busca local a meta-heurística *Simulated Annealing* (SA). A partir disso apresentar uma forma alternativa para resolver o PSTMP-SR.

1.2.2 Objetivos Específicos

a) Desenvolver a meta-heurística GRASP com SA, adaptado para resolver o PSTMP-SR.

b) Aplicar o método desenvolvido as instâncias disponibilizas por Ruiz e Andrés (2011).

c) Avaliar o desempenho do método hibrido GRASP e SA (GRASP-SA), por meio de experimentos computacionais, comparando os resultados com os obtidos por Kampke(2010).

1.3 HIPÓTESE

A Hipótese desse trabalho é que o método hibrido GRASP juntamente com SA, seja capaz de resolver o PSTMP-SR fornecendo um modelo alternativo para resolver o problema.

1.4 ESTRUTURA DO TRABALHO

O restante deste trabalho está dividido como a seguir: no Capítulo 2 é constituído uma revisão bibliográfica com os principais e mais recentes trabalhos da literatura. O Capítulo 3 descreve a metodologia utilizada. Os resultados obtidos são apresentados no Capítulo 4, e por fim as conclusões são apresentadas no Capítulo 5.

2 REVISÃO DE LITERATURA

O problema PTMP abordado neste trabalho foi apresentado pela primeira vez por RUIZ e ANDRÉS (2007) e por relacionar os tempos para preparação e os recursos é denotado por PPTMPSR (Problema de Programação de Tarefas em Máquinas Paralelas com Setup Times e Recursos). Problema este provado ser NP-Difícil (RUIZ e ANDRÉS, 2007). A prova consiste em reduzir o problema a um problema clássico de PTMP com setup times dependentes apenas da sequência, que é NP-Difícil (WEBSTER, 1997).

Paula (2008) para resolver o Problema PSTMP-S, utilizou a meta-heurística GRASP, além da meta-heurística *Variable Neighborhood Search* (VNS) e de um híbrido de GRASP que utiliza VNS como busca local, e obteve soluções melhores para as instâncias de 6 máquinas e 300 tarefas em 95% dos resultados já encontrados na literatura.

Já Kampke (2010), utilizou o GRASP com o mecanismo reativo, juntamente com o *Path Relinking*, além do *Iterated Local Search* (ILS) para resolver PSTMP-SR em instâncias de médio e de grande porte. Assim, ele obteve os melhores resultados para as instâncias de grande porte.

Ruiz e Andrés (2011), utilizou o modelo *mixed integer program* (MIP) e algumas heurísticas *fast dispatching* (de envio rápido), como o *Dynamic Job Assignment with Setups Resource Assignment* (DJASA), que demonstrou os melhores resultados para o PSTMP-SR.

Uma heurística híbrida de GRASP e *Variable Neighborhood Descent* (VND) foi proposta por Diana *et. al.* (2012) para resolver o PSTMP-S. O objetivo foi aprimorar os resultados obtidos com o Algoritmo Genético de Vallada e Ruiz (2011).

Haddad (2012), desenvolveu o GIVMP, que é que a mistura de várias heurísticas como GRASP, ILS, VND e *Path Relinking*. Entretanto, o método não foi capaz de superar todos os resultados da literatura.

Já Haddad (2014) propôs um algoritmo denominado AIV baseado no ILS e VND. Este algoritmo implementa um *Adaptive Shortest Processing Time* (ASPT) a fim de criar uma solução inicial. O procedimento *Random Variable Neighborhood Descent* (RVND) foi utilizado para realizar a busca local, explorando aleatoriamente o espaço de solução com várias inserções e movimentos de troca. Uma perturbação na AIV, é uma aplicação de um movimento de inserção. AIV foi utilizado em problemas de teste

disponiveis na literatura e os resultados foram comparados com dois algoritmos genéticos, GA1 e GA2, ambos desenvolvidos por Vallada e Ruiz (2011). A análise estatística dos resultados computacionais mostraram que AIV é capaz de produzir, em média, melhores soluções do que ambos os algoritmos genéticos.

O método heurístico denominado AIA que é uma compilação de ASPT, ILS e *Adaptive Local Search* (ALS), criado por Cota *et. al.* (2014), obteve bons resultados ao ser comparado com os métodos GA2 (VALLADA E RUIZ, 2011) e AIV (HADDAD, 2014) quando aplicados ao PSTMP-S.

Entre os trabalhos mais recentes temos o de Pereira, Coura e Celeste (2014) que obtiveram ótimos resultados para reconfiguração de micro redes utilizando o GRASP. Assim, concluíram que o GRASP é considerado uma das melhores heurísticas para otimização combinatória.

Já Duarte *et.al.* (2015), desenvolveu um hibrido do GRASP, *path relinking* e VNS, para resolver o problema de mínima dispersão diferencial, o método demonstrou grande competitividade em comparação com outros resultados da literatura.

3 METODOLOGIA

A metodologia proposta nesse trabalho consiste na implementação da meta-heurística GRASP, para resolver o PSTMP-SR. Também será utilizado no GRASP uma adaptação do método construtivo DJASA de Ruiz e Andrés (2007), e como busca local, aplicada com os métodos propostos, será implementada a meta-heurística *Simulated Annealing*.

3.1 MODELAGEM MATEMÁTICA

O modelo matemático pretende descrever o problema de forma realista com o interesse de minimizar o tempo de conclusão total da tarefa *j* denotado por C_j. O tempo total de produção é definido por:

$$T_p = \sum_{j=1}^{n} C_j$$

Para minimizar o *makespan*, os *setup times* deverão ser os menores possíveis, acarretando na maximização dos recursos associados a eles. Porém o número de recursos também deve ser minimizado. Considerando R_{ijk} a quantidade de recursos associada ao *setup time* S_{ijk}, caso a tarefa *j* anteceda a tarefa *k* na máquina *i* e R_{ijk} = 0, caso contrário. Sendo T_{rec}, o total de recursos utilizados, pode-se definir T_{rec} por:

$$T_{rec} = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1, k \neq j}^{n} R_{ijk}$$

Tendo λ e δ como valores que definem o custo da unidade de recurso e tempos de conclusão, respectivamente, e C_{ij} o tempo de conclusão da tarefa *j* na máquina *i*, considerando que C_{ij} = 0 se a tarefa *j* não for processada na máquina *i*. Pode-se definir a função objetivo como:

$$Z = \lambda \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} R_{ijk} + \delta \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij}$$

A função objetivo para o problema, pretende minimizar a combinação linear do total de recursos (T_{rec}) e o tempo total de conclusão das tarefas.

3.2 *DYNAMIC JOB ASSIGNMENT WITH SETUPS RESOURCE ASSIGNMENT* (DJASA)

O método construtivo DJASA foi proposto por Ruiz e Andrés (2007). Este método é utilizado para encontrar qual a melhor ordem de execução de tarefas realizadas por uma máquina *i* de maneira dinâmica, considerando em cada iteração as tarefas que ainda não foram sequenciadas e todas as outras máquinas. Assim, visa reduzir o tempo total de operação.

O método DJASA precisa decidir a quantidade de recursos que será atribuído ao *setup time* entre as tarefas já sequenciadas, e para cada tarefa não sequenciada é feita uma simulação de inserção dessas tarefas nas *m* máquinas, depois disso as tarefas são ordenadas de acordo com o valor da função objetivo obtido por essas simulações. Dessa forma o método possui três variações em função do número de recursos (máximo, mínimo ou valor médio), sendo denominados: DJASA^{+}, DJASA^{-} e DJASA^{av}, respectivamente. Veja a seguir o algoritmo do método DJASA:

1. Adicione todas as tarefas numa lista de tarefas não sequenciadas ρ.
2. Enquanto $\rho \neq \emptyset$ faça:
    a) Para cada tarefa *k* em ρ e para cada máquina $i \in M$ faça:
        i. Temporariamente atribua recursos ao *setup time* entre a tarefa k e a tarefa anteriormente associada à máquina *i*.
        ii. Sequencie temporariamente a tarefa *k* na máquina *i*.
        iii. Calcule o valor da função objetivo *Z* após a associação dos recursos e da tarefa.
    b) Seja *j* e *l* a tarefa e a máquina, respectivamente, que resultaram no menor

incremento na função objetivo.

c) Atribua os recursos ao *setup time* entre a tarefa *j* e a tarefa anteriormente sequenciada na máquina *l*.
d) Sequencie a tarefa j na máquina l.
e) Remova a tarefa j da lista *ρ.*

3.3 GRASP

Feo e Resende (1995) propuseram a meta-heurística GRASP, na qual é um método iterativo, sendo que em cada iteração o método reinicia com a criação de uma solução. No Algoritmo 1 é apresentado o pseudocódigo padrão da meta-heurística GRASP.

```
GRASP (α)
    Zmin ← +∞;
    enquanto não CritérioDeParada faça
        S1 ← Construção_Solução (α);
        S2 ← Busca_Local (S1);
        se Z(S2) < Zmin então
            S ← S2;
            Zmin ← Z(S2);
        fim-se
    fim-enquanto;
    Retorne S;
fim;
```

Algoritmo 1 - Pseudocódigo do GRASP
Fonte: Feo e Resende (1995).

Dentre todas as soluções é retornada a de menor valor, calculada pela função objetivo *Z*. Os parâmetros definidos são o critério de parada (por tempo ou número de iterações) e a taxa de aleatoriedade *α* (definida por valores $0 \leq \alpha \leq 1$) em que os maiores valores de *α*, serão de soluções mais aleatórias.

Enquanto o critério de parada não é satisfeito, iterativamente, uma solução s_1 é construída e melhorada pelo procedimento de busca local. O algoritmo armazena a melhor solução encontrada, retornando a melhor dentre todas as iterações.

3.3.1 Construção da Solução

De acordo com Kampke(2010), a cada iteração do método GRASP, uma solução é construída, também de forma iterativa. Nessa etapa, em cada iteração é adicionada um novo elemento à solução. Os elementos ainda não adicionados formam uma Lista de Candidatos (LC).

A lista de melhores candidatos é denominada Lista Restrita de Candidatos (LRC). O tamanho da LRC é determinado pelo valor α, sendo que quanto maior o valor de α, maior será o tamanho de LRC, ou seja, mais elementos terá a lista, fazendo com que a escolha do elemento seja feita de forma mais aleatória. Por outro lado, quanto menor o valor de α, menor será o tamanho de LRC, assim a escolha será mais gulosa, ou seja, buscando sempre as melhores soluções, pois considerará apenas os melhores candidatos.

Na fase de construção do GRASP para resolver o PSTMP-SR, cada tarefa de LC não sequenciada, ou seja, que ainda não faz parte da solução, será inserida na solução de acordo com o método DJASA. Vale ressaltar que no método DJASA original sempre é escolhida a tarefa e a máquina que apresentaram o menor incremento na função objetivo *Z*. Neste trabalho, o método construtivo implementado escolhe a tarefa aleatoriamente dentre as melhores que parte de LRC. Portanto, pode-se supor que as soluções construídas pelo método proposto são de qualidade.

3.3.2 Busca Local

O *Simulated Annealing* é um método de busca local que aceita soluções piores, de forma a escapar de ótimos locais (KIRKPATRICK et al., 1983). Como outros métodos de busca local, este método requer uma estrutura de vizinhança definida e uma função de avaliação *Z* que associa a cada solução um valor que representa o custo da solução.

O algoritmo é iniciado com uma solução inicial construída pelo método de construção, e seleciona aleatoriamente uma solução *s'* vizinha de *s*. A cada geração de *s'*, testa-se a variação do valor na função objetivo, isto é, $\Delta Z = Z(S') - Z(S)$. Se $\Delta Z < 0$, então $Z(S') < Z(S)$, logo a solução *S'* apresenta melhor valor comparado à solução

s e o método aceita *S'*, que passa a ser a nova solução *S*.

Se $\Delta Z > 0$, é calculada a probabilidade de aceitação $e^{(-\Delta Z/T)}$, na qual *T* é um parâmetro do método, denominado por temperatura, que regula a probabilidade de aceitar ou não as soluções de acordo com o nível de piora da função objetivo. O procedimento só se encerra quando a temperatura chegar próximo de zero e nenhuma solução seja pior que a melhor solução, isto é, a solução evidencia um ótimo local. O Algoritmo 2 demonstra o pseudocódigo do *Simulated Annealing*.

```
Simulated Annealing
S ← S*;
T ← T0;
enquanto temperatura elevada faça
    para interações para equilíbrio faça
        Gerar uma solução S' de N(S);
        Avaliar a variação de energia – ΔZ = Z(S') – Z(S);
        se ΔZ < 0 então
            S ← S';
            se Z(S') < Z(S*) então
                S* ← S';
            fim-se
        senão
            Gerar u ∈ random[0,1];
            se u < exp(−ΔZ/T) então
                S ← S';
            fim-se
        fim-se
    fim-para
    reduzir a temperatura T;
fim-enquanto
fim.
```

Algoritmo 2 - Pseudocódigo do *Simulated Annealing*.
Fonte: KIRKPATRICK et al., (1983)

.

4 RESULTADOS E DISCUSSÃO

Para a realização dos experimentos, foi utilizada uma plataforma com processador Intel Core I5 3210M com 2 núcleos de 2.50GHz, 6GB de memória RAM. O método proposto foi implementado e modelado na linguagem de programação C++.

Foram executados testes com a finalidade de avaliar o desempenho do método proposto, em comparação dos resultados disponíveis na literatura. Foram realizados testes com as 720 instâncias do problema, que estão divididas em dois grupos: *small* e *large*, com 360 instâncias cada um. Além disso, cada grupo é subdividido em 36 combinações de valores de *n, m*, valor mínimo e máximo de *setup times* e quantidade mínima e máxima de recursos, para cada combinação, em cada grupo, foram geradas 10 instâncias de teste por Ruiz e Andrés (2011).

A função objetivo Z conforme descrita anteriormente, busca minimizar o tempo de conclusão das tarefas e o número de recursos para preparar as máquinas. Para modularizar o valor da função objetivo são utilizados os parâmetros λ e δ, que representam os custos de recurso e de tempo, respectivamente. Para critério de comparação foram utilizados mesmos valores de Kampke(2010), em que $\lambda = 50$ e $\delta = 1$. Esses valores significam que o custo de uma unidade de recurso é 50 vezes mais caro que o custo de uma unidade de tempo.

Também para critérios de comparação foi utilizado o parâmetro α no método de construção do método GRASP, foi definido $\alpha = 0{,}5$ para as instâncias *small* e $\alpha = 0{,}1$ para instâncias *large,* como definidas por Kampke(2010).

Para o SA foi utilizado como critério de parada o tempo de 1 segundo e com temperatura inicial $T_0 = 1000$, para que o GRASP executasse o máximo de iterações possíveis dentro do tempo de execução de cada instância, assim, gerando muitos vizinhos em cada interação.

O tempo de parada do GRASP-SA, foi definido através de testes preliminares, sendo que para instâncias *small* o tempo de execução do GRASP foi $Tc = (n - m)/2$ e para instâncias *large* $Tc=(n-m)/10$, pois em instâncias muito grandes ficaria executando muito tempo desnecessariamente, já que o resultados eram encontrados nos primeiros minutos.

As Tabelas a seguir, apresentam os resultados obtidos em cada um dos métodos, expondo somente a média dos valores de *Z* para cada conjunto de 10

instâncias que pertencem a mesma combinação dos valores de *n, m*, valor mínimo e máximo de *setup times* e quantidade mínima e máxima de recursos.

Nesta seção será demonstrado os resultados da comparação com os melhores resultados obtidos por Kampke(2010) e a meta-heurística GRASP em combinação com os métodos DJASA como método construtivo e *Simulated Annealing* como busca local. Será utilizado a sigla (GRASP-SA) para simplificar a nomenclatura da união do métodos.

As Tabelas 1, 2, 3, 4, apresentam os resultados obtidos por cada método em cada grupo de instâncias, *small* e *large*, respectivamente. Destacando os valores em negrito para representar os melhores resultados médios listados na respectiva Tabela.

Instâncias				***Médias Z***	
n	***m***	***$S^-;S^+$***	***$R^-;R^+$***	***GRASP-SA***	**Kampke(2010)**
6	3	[1,50];[50,100]	[1,3];[3;5]	6.945	**6.318**
			[1,5];[5;10]	7.773	**7.169**
		[50,100];[100,150]	[1,3];[3;5]	8.152	**7.986**
			[1,5];[5;10]	8.660	**8.057**
	4	[1,50];[50,100]	[1,3];[3;5]	4.751	**4.360**
			[1,5];[5;10]	4.985	**4.200**
		[50,100];[100,150]	[1,3];[3;5]	5.582	**5.168**
			[1,5];[5;10]	5.995	**5.238**
	5	[1,50];[50,100]	[1,3];[3;5]	2.946	**2.743**
			[1,5];[5;10]	2.697	**2.473**
		[50,100];[100,150]	[1,3];[3;5]	3.489	**3.181**
			[1,5];[5;10]	3.721	**3.131**
8	3	[1,50];[50,100]	[1,3];[3;5]	12.057	**11.029**
			[1,5];[5;10]	13.492	**12.236**
		[50,100];[100,150]	[1,3];[3;5]	15.387	**14.811**
			[1,5];[5;10]	17.113	**16.047**
	4	[1,50];[50,100]	[1,3];[3;5]	8.822	**8.287**
			[1,5];[5;10]	10.205	**8.772**
		[50,100];[100,150]	[1,3];[3;5]	10.546	**9.965**
			[1,5];[5;10]	11.780	**10.709**
	5	[1,50];[50,100]	[1,3];[3;5]	6.419	**5.748**
			[1,5];[5;10]	7.247	**6.459**
		[50,100];[100,150]	[1,3];[3;5]	8.108	**7.193**
			[1,5];[5;10]	8.652	**7.916**
10	3	[1,50];[50,100]	[1,3];[3;5]	17.282	**16.003**
			[1,5];[5;10]	20.076	**18.301**
		[50,100];[100,150]	[1,3];[3;5]	23.824	**22.866**
			[1,5];[5;10]	27.669	**25.870**
	4	[1,50];[50,100]	[1,3];[3;5]	13.858	**12.782**
			[1,5];[5;10]	15.432	**13.280**
		[50,100];[100,150]	[1,3];[3;5]	16.915	**16.051**
			[1,5];[5;10]	18.814	**16.950**
	5	[1,50];[50,100]	[1,3];[3;5]	9.780	**9.139**
			[1,5];[5;10]	11.556	**9.925**
		[50,100];[100,150]	[1,3];[3;5]	12.630	**11.848**
			[1,5];[5;10]	14.726	**12.144**

Tabela 1 - Comparação das médias de valor de Z nas instâncias do grupo *small* obtidas pelo método GRASP-SA

Instâncias				***Médias de Z***	
n	***m***	***$S^-;S^+$***	***$R^-;R^+$***	***GRASP-SA***	**Kampke(2010)**
50	10	[1,50];[50,100]	[1,3];[3;5]	**92.555**	92.615
			[1,5];[5;10]	118.918	**112.832**
		[50,100];[100,150]	[1,3];[3;5]	**143.253**	143.256
			[1,5];[5;10]	165.885	**164.401**
	15	[1,50];[50,100]	[1,3];[3;5]	68.455	**67.979**
			[1,5];[5;10]	83.720	**74.990**
		[50,100];[100,150]	[1,3];[3;5]	98.430	**97.742**
			[1,5];[5;10]	112.856	**105.534**
	20	[1,50];[50,100]	[1,3];[3;5]	52.554	**49.928**
			[1,5];[5;10]	61.145	**53.859**
		[50,100];[100,150]	[1,3];[3;5]	72.566	**70.002**
			[1,5];[5;10]	79.795	**74.022**
75	10	[1,50];[50,100]	[1,3];[3;5]	169.190	**166.415**
			[1,5];[5;10]	**221.951**	236.000
		[50,100];[100,150]	[1,3];[3;5]	289.186	**286.680**
			[1,5];[5;10]	**344.929**	354.873
	15	[1,50];[50,100]	[1,3];[3;5]	132.112	**130.347**
			[1,5];[5;10]	167.748	**159.490**
		[50,100];[100,150]	[1,3];[3;5]	206.321	**205.400**
			[1,5];[5;10]	242.385	**236.376**
	20	[1,50];[50,100]	[1,3];[3;5]	108.058	**107.804**
			[1,5];[5;10]	130.117	**120.296**
		[50,100];[100,150]	[1,3];[3;5]	**158.829**	160.067
			[1,5];[5;10]	182.916	**172.409**
100	10	[1,50];[50,100]	[1,3];[3;5]	257.844	**246.576**
			[1,5];[5;10]	**346.393**	359.014
		[50,100];[100,150]	[1,3];[3;5]	476.787	**469.931**
			[1,5];[5;10]	**568.132**	585.176
	15	[1,50];[50,100]	[1,3];[3;5]	203.132	**195.793**
			[1,5];[5;10]	**266.830**	270.150
		[50,100];[100,150]	[1,3];[3;5]	345.259	**336.943**
			[1,5];[5;10]	**408.227**	409.763
	20	[1,50];[50,100]	[1,3];[3;5]	172.647	**167.631**
			[1,5];[5;10]	220.038	**206.661**
		[50,100];[100,150]	[1,3];[3;5]	269.843	**267.809**
			[1,5];[5;10]	318.832	**306.226**

Tabela 2 - Comparação das médias de valor de Z nas instâncias do grupo *large* obtidas pelo método GRASP-SA

Na etapa a seguir, é apresentada a comparação entre os resultados do método proposto e os resultados de Kampke(2010). Para essas comparações utilizamos a medida da Porcentagem Média de Desvio (PMD), calculada pela seguinte fórmula:

$$PMD = \frac{1}{R} \sum_{r=1}^{R} \frac{(Z_{melhor} - Z_h)}{Z_{melhor}} \times 100\%$$

em que Z_{melhor} representa a solução que será comparada e Z_h representa a solução obtida pelo método proposto, R representa o número de instâncias avaliadas.

Deste modo, se o valor de *PMD* for um valor *x* positivo ($x > 0$), então as soluções obtidas pelo método proposto são em média *x%* melhores que a solução utilizada para comparação. Caso contrário, se *x* for negativo ($x < 0$), as soluções obtidas pelo método proposto não conseguiu encontrar, em média, resultados melhores.

As Tabelas 3 e 4, listam o valor de PMD, que são utilizado para comparar os resultados da meta-heurística GRASP-AS com os de Kampke (2010).

Ao comparar as instâncias do grupo *small*, os melhores resultados da meta-heurística GRASP-SA e os melhores resultados da literatura, não foi possível obter nenhum resultado melhor, como pode ser analisado nas Tabelas 1 e 3.

Porém, nas instâncias do grupo *large*, foi possível visualizar uma pequena melhora em algumas das médias das instâncias, demonstradas na coluna do GRASP-SA em negrito nas Tabela 2 e 4. Em apenas 9 das 36 grupos foi possível perceber essas melhorias. Sendo que em um grupo foi possível observar uma melhoria de 0,6% em relação aos resultado de Kampke (2010).

Mesmo que as médias de resultados obtidos pelo método proposto não foram maiores que as de Kampke(2010), o GRASP–SA conseguiu, em média, resultados piores de apenas 0,93% e 0,25% para as instâncias *small* e *large*, respectivamente. Os resultado podem não ter atingido valores melhores pois a escolha dos parametros para o Simulated Annealing foram escolhido tendo como prioridade o baixo tempo de execução, buscando reduzir o tempo computacional para gerar a solução.

Instâncias				***PMD (%)***
n	***m***	***$S^-;S^+$***	***$R^-;R^+$***	***GRASP-SA***
6	3	[1,50];[50,100]	[1,3];[3;5]	-0,99
			[1,5];[5;10]	-0,84
		[50,100];[100,150]	[1,3];[3;5]	-0,21
			[1,5];[5;10]	-0,75
	4	[1,50];[50,100]	[1,3];[3;5]	-0,90
			[1,5];[5;10]	-1,87
		[50,100];[100,150]	[1,3];[3;5]	-0,80
			[1,5];[5;10]	-1,45
	5	[1,50];[50,100]	[1,3];[3;5]	-0,74
			[1,5];[5;10]	-0,91
		[50,100];[100,150]	[1,3];[3;5]	-0,97
			[1,5];[5;10]	-1,88
8	3	[1,50];[50,100]	[1,3];[3;5]	-0,93
			[1,5];[5;10]	-1,03
		[50,100];[100,150]	[1,3];[3;5]	-0,39
			[1,5];[5;10]	-0,66
	4	[1,50];[50,100]	[1,3];[3;5]	-0,65
			[1,5];[5;10]	-1,63
		[50,100];[100,150]	[1,3];[3;5]	-0,58
			[1,5];[5;10]	-1,00
	5	[1,50];[50,100]	[1,3];[3;5]	-1,17
			[1,5];[5;10]	-1,22
		[50,100];[100,150]	[1,3];[3;5]	-1,27
			[1,5];[5;10]	-0,93
10	3	[1,50];[50,100]	[1,3];[3;5]	-0,80
			[1,5];[5;10]	-0,97
		[50,100];[100,150]	[1,3];[3;5]	-0,42
			[1,5];[5;10]	-0,70
	4	[1,50];[50,100]	[1,3];[3;5]	-0,84
			[1,5];[5;10]	-1,62
		[50,100];[100,150]	[1,3];[3;5]	-0,54
			[1,5];[5;10]	-1,10
	5	[1,50];[50,100]	[1,3];[3;5]	-0,70
			[1,5];[5;10]	-1,64
		[50,100];[100,150]	[1,3];[3;5]	-0,66
			[1,5];[5;10]	-2,13
Médias				-0,93%

Tabela 3 - PMD-Comparação dos melhores resultados heurísticos da literatura com os métodos baseados na meta-heurística GRASP-SA (grupo *small*)

Instâncias				***PMD (%)***
m	***m***	***$S^-;S^+$***	***$R^-;R^+$***	***GRASP-SA***
50	10	[1,50];[50,100]	[1,3];[3;5]	**0,01**
			[1,5];[5;10]	-0,54
		[50,100];[100,150]	[1,3];[3;5]	**0,00**
			[1,5];[5;10]	-0,09
	15	[1,50];[50,100]	[1,3];[3;5]	-0,07
			[1,5];[5;10]	-1,16
		[50,100];[100,150]	[1,3];[3;5]	-0,07
			[1,5];[5;10]	-0,69
	20	[1,50];[50,100]	[1,3];[3;5]	-0,53
			[1,5];[5;10]	-1,35
		[50,100];[100,150]	[1,3];[3;5]	-0,37
			[1,5];[5;10]	-0,78
75	10	[1,50];[50,100]	[1,3];[3;5]	-0,17
			[1,5];[5;10]	**0,60**
		[50,100];[100,150]	[1,3];[3;5]	-0,09
			[1,5];[5;10]	**0,28**
	15	[1,50];[50,100]	[1,3];[3;5]	-0,14
			[1,5];[5;10]	-0,52
		[50,100];[100,150]	[1,3];[3;5]	-0,04
			[1,5];[5;10]	-0,25
	20	[1,50];[50,100]	[1,3];[3;5]	-0,02
			[1,5];[5;10]	-0,82
		[50,100];[100,150]	[1,3];[3;5]	**0,08**
			[1,5];[5;10]	-0,61
100	10	[1,50];[50,100]	[1,3];[3;5]	-0,46
			[1,5];[5;10]	**0,35**
		[50,100];[100,150]	[1,3];[3;5]	-0,15
			[1,5];[5;10]	**0,29**
	15	[1,50];[50,100]	[1,3];[3;5]	-0,37
			[1,5];[5;10]	**0,12**
		[50,100];[100,150]	[1,3];[3;5]	-0,25
			[1,5];[5;10]	**0,04**
	20	[1,50];[50,100]	[1,3];[3;5]	-0,30
			[1,5];[5;10]	-0,65
		[50,100];[100,150]	[1,3];[3;5]	-0,08
			[1,5];[5;10]	-0,41
Médias				-0.25

Tabela 4 - PMD-Comparação dos melhores resultados heurísticos da literatura com os métodos baseados na metaheurística GRASP-SA (grupo large)

5 CONCLUSÕES

O presente trabalho abordou o Problema de Sequenciamento de Tarefas em Máquinas Paralelas com Tempos de Preparação Dependentes da Sequência e de Recursos (PSTMP-SR).

Para a resolução do problema proposto foi utilizada a meta-heurística *GRASP,* com uma variação do método construtivo DJASA e na busca local a meta-heurística SA.

Os resultados obtidos pelo *GRASP-SA* para o resolução do problema PSTMP-SR foram satisfatórios, pois em alguns casos foram superiores aos resultados encontrados na literatura.

O *GRASP-SA* se mostrou como uma boa alternativa para a resolução do PSTMP-SR. Em trabalhos futuros, novos métodos poderão ser explorados, como a variação do *GRASP*, o *GRASP Reativo* com *Simulated Annealing* ou *Algoritmo Genético* com *Simulated Annealing*.

Por fim, os algoritmos implementados estão disponíveis em *https://github.com/diegomoe/tcc-diego*.

6 REFERÊNCIAS

COTA, L. P., HADDAD, M. N., SOUZA, M. J. F., MARTINS, A. X. Um algoritmo heurístico para resolver o problema de sequenciamento em máquinas paralelas não-relacionadas com tempos de preparação dependentes da sequência. **XXXV Congresso Nacional de Matemática Aplicada e Computacional**, v.3, Natal – RN, 2014.

DIANA, R. O.M.; SIQUEIRA, E C.; FILHO, M. F.F.; SOUZA, S R. Algoritmo imuno inspirado para minimização de *makespan* em máquinas paralelas não relacionadas, com tempos de preparação dependentes da sequência**. Anais do XIX Congresso Brasileiro de Automática**, 2012.

DUARTE, A.; SANCHEZ-ORO, J.; RESENDE, M. G. C.; GLOVER, F.; MARTÍ, R. Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. **Information Sciences**, v. 296, pag. 46-60, 2015.

FEO, T.A.; RESENDE, M.G. C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, pag. 109-133, 1995.

HADDAD, M. N. **Algoritmos heurísticos híbridos para o problema de sequenciamento em máquinas paralelas não-relacionadas com tempos de preparação dependentes da sequência**. 89 f. 2012. Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Ouro Preto, 2012.

HADDAD, M. N.; COTA, L. P.; SOUZA, M. J. F, MACULAN, N. AIV: A Heuristic Algorithm based on Iterated Local Search and Variable Neighborhood Descent for solving the Unrelated Parallel Machine Scheduling Problem with Setup Times, **16th International Conference on Enterprise Information Systems**, pag. 376-383, Lisboa, Portugal, 2014.

KAMPKE, E. H. **Metaheurísticas para problema de programação de tarefas em máquinas paralelas com tempos de preparação dependentes da sequência e de recursos**. 111 f. 2010. Dissertação (Mestrado) - Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Viçosa, Viçosa - MG. 2010.

KIRKPATRICK, S.; GELLAT, D. C.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, v. 220, pag. 671-680, 1983.

PAULA, M. R. **Heurísticas para a minimização dos atrasos em sequenciamento de máquinas paralelas com tempos de preparação dependentes da sequência.** 121 f. 2008. Dissertação (Mestrado) - Programa de Pós Graduação em Ciência da Computação, Universidade Federal de Minas Gerais. 2008.

PEREIRA, A. L.; COURA, D. J. C.; CELESTE, W. C. Utilização da Metaheurística GRASP como Solução para o Problema de Reconfiguração de Redes Inteligentes de Energia. **Anais do Congresso Sul Brasileiro de Computação**, v. 7, 2014.

RUIZ, R.; ANDRÉS, C. Unrelated parallel machines scheduling with resourceassignable sequence dependent setup times. Em Baptiste, P., Kendall, G., Munier-Kordon, A., Sourd, F., eds.: **Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)**, pag. 439–446, Paris, France, 2007.

RUIZ, R.; ANDRÉS, C. Scheduling unrelated parallel machines with resource-assignable sequence dependent setup times. **International Journal of Advanced Manufacturing Technology**, v.57, pag. 777-794, 2011.

VALLADA, E.; RUIZ, R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. ***European Journal of Operational Research*,** v. 211, pag. 612–622. 2011.

VOLMANN, T.E.; BERRY, W.L.; WHYBARK D.C. **Manufacturing Planning and Control Systems**, Dow Jones-Irwin, 2ª edição, 1988.

WEBSTER, S.T. The complexity of scheduling job families about a common date. **Operations Research Letters**, 20(2), 65–74, 1997.