UNIVERSIDADE FEDERAL DO PARANÁ

VINICIUS MARTINS TON

GREEDY RANDOMIZED ADAPTIVE EVOLUTIONARY PATH RELINKING APLICADO A PROBLEMAS DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM RECURSOS RENOVÁVEIS

CURITIBA

2020

VINICIUS MARTINS TON

GREEDY RANDOMIZED ADAPTIVE EVOLUTIONARY PATH RELINKING APLICADO A PROBLEMAS DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM RECURSOS RENOVÁVEIS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Métodos Numéricos em Engenharia, área de concentração em Programação Matemática, no Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Setores de Tecnologia e Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. José Eduardo Pécora Junior
Coorientador: Prof. Dr. Maurício Guilherme de Carvalho Resende

CURITIBA

2020

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

T663g Ton, Vinicius Martins
Greedy Randomized Adaptive Evolutionary Path Relinking aplicado a problemas de máquinas paralelas não relacionadas com recursos renováveis [recurso eletrônico] Vinicius Martins Ton. – Curitiba, 2020.

Dissertação - Universidade Federal do Paraná, Ciências Exatas, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, 2020.

Orientador: José Eduardo Pécora Junior.
Coorientador: Maurício Guilherme de Carvalho Resende.

1. Programação (matemática). 2. Linguagem de programação (Computadores). I. Universidade Federal do Paraná. II. Pécora Junior, José Eduardo. III. Resende, Maurício Guilherme de Carvalho. IV. Título.

CDD: 005.133

Bibliotecária: Vanusa Maciel CRB- 9/1928

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **VINICIUS MARTINS TON** intitulada: **GREEDY RANDOMIZED ADAPTIVE EVOLUTIONARY PATH RELINKING APLICADO A PROBLEMAS DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM RECURSOS RENOVÁVEIS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 30 de Julho de 2020.

Assinatura Eletrônica
06/08/2020 16:52:37.0
MAURICIO GUILHERME DE CARVALHO RESENDE
Presidente da Banca Examinadora

Assinatura Eletrônica
30/07/2020 14:23:04.0
LEONARDO SILVA DE LIMA
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
30/07/2020 14:20:11.0
LUCIANA DE SOUZA PESSÔA
Avaliador Externo (PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO)

Assinatura Eletrônica
30/07/2020 14:51:09.0
GUSTAVO VALENTIM LOCH
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Este trabalho é dedicado a todos aqueles que ainda acreditam na ciência.

AGRADECIMENTOS

Primeiramente à Deus que guiou cada passo meu neste caminho.

À minha mãe Terezinha, que neste período lutou bravamente e venceu a batalha contra o câncer, me ensinando a nunca desistir. Ela quem sempre me apoiou e apoia em todas as minhas decisões, sem ela nada seria possível.

À minha noiva Tanize, por trazer a leveza e a calma que eu precisava após um dia difícil. Por estar sempre presente e ser meu suporte neste período. Por acreditar em mim, apoiar os meus sonhos e embarcar nesta aventura que virá.

Ao professor Pécora, que além de orientador se tornou um grande amigo. Por todas as conversas que me fizeram crescer pessoal e profissionalmente. Por sua orientação, por sua dedicação, por seus conselhos, por sempre acreditar no meu potencial e me proporcionar diversas oportunidades. Também agradeço pela paciência e pelas incontáveis horas em vídeo chamadas.

Ao professor Maurício Resende, por aceitar esta coorientação e por compartilhar seus conhecimentos. Sua contribuição foi valiosa para conclusão deste trabalho.

Aos professores do GTAO, professor Cassius e professor Gustavo pelas dicas, conselhos e ensinamentos durante o período do mestrado.

Às amizades que o nosso grupo de pesquisa me proporcionou, em especial à Andressa, à Kellen, à Thayse, à Bruna, à Thalita e ao Bruno, sem vocês os dias sofridos debugando códigos não seriam prazerosos. Obrigado por toda a ajuda, por todos os momentos de descontração e desabafos e por todos os cafézinhos.

Ao professor Angel, que proporcionou a oportunidade de realizar um estágio de pesquisa na Université Laval e a experiência de trabalhar em um dos mais renomados centros de pesquisas, o CIRRELT. Agradeço também por aceitar continuar esta parceria nos próximos quatro anos.

A todos os meus amigos e meus familiares, agradeço todo o incentivo e me desculpo pela minha ausência. Agradeço em especial à Aline, que acompanhou esta jornada de perto, e inclusive me deu um lar quando mais precisei.

"Desta passagem, a aprendizagem é a única bagagem levada."
(Vitor Isensee)

RESUMO

Esta dissertação aborda o problema de máquinas paralelas não relacionadas, com restrição de recursos renováveis (UPMR), para minimizar o *makespan*. Para este problema é proposto um *Greedy Randomized Adaptive Evolutionary Path-Relinking* (GRAEPR) e uma abordagem híbrida com um modelo de programação por restrição (CP). Os resultados apresentam soluções competitivas com as presentes na literatura, estabelecendo alguns novos *Lower* e *Upper Bounds*. Além disso, é apresentada uma extensão para este problema. É introduzido o problema de máquinas paralelas não relacionadas, com *setup* dependente e restrição de recursos renováveis (UPMSR). Para este problema é apresentado um modelo de programação inteira mista (MILP), um modelo de programação por restrição e uma uma adaptação da abordagem de Fleszar e Hindi (2018). Além disso, são modificadas as abordagens do *Greedy Randomized Adaptive Evolutionary Path-Relinking* e híbrida desenvolvidas para o UPMR. Um conjunto de instâncias é gerada para UPMSR e os resultados evidenciam o potencial existente na abordagem GRAEPR.

Palavras-chaves: Máquinas paralelas não relacionadas. Restrição de recursos Renováveis. Programação linear inteira mista. Programação por restrição. *Path-relinking*.

ABSTRACT

This thesis addresses the problem of unrelated parallel machines, with restriction of renewable resources (UPMR), to minimize the makespan. For this problem, a Greedy Randomized Adaptive Evolutionary Path-Relinking (GRAEPR) and a hybrid approach with a constraint programming (CP) model is proposed. The results show competitive solutions with those found in the literature, establishing some new values for Lower and Upper Bounds. In addition, an extension is presented for this problem. We introduce the problem of unrelated parallel machines, with dependent setup and restriction of renewable resources (UPMSR). For this problem, we present a mixed integer linear programming (MILP) model, a contraint programming (CP) model, and an adaptation of the approach of Fleszar and Hindi (2018). We also modify the Greedy Randomized Adaptive Evolutionary Path-Relinking and the hybrid approach developed for the UPMR. A set of instances is generated for UPMSR and the results show the potential that exists in the GRAEPR approach.

Key-words: *Unrelated parallel machines. Renewable resource constraint. Mixed-integer linear programming. Constraint programming. Path-relinking*.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E DE SIGLAS

NP-Hard *Non-deterministic Polynomial-time Hard*

AA *Approximation Algorithm*

ACO *Ant Colony Optmization*

AIS *Artificial Immune System*

EIG *Enriched Iterated Greedy*

EPR *Evolutionary Path-Relinking*

ESS *Enriched Scatter Search*

FFOA *Fruit Fly Optimization Algorithm*

GRAEPR *Greedy Randomized Adaptive Evolutionary Path Relinking*

GRAPR *Greedy Randomized Adaptive Path-Relinking*

HGA *Hybrid Genetic Algorithm*

HIM *heuristic improvement method*

He *Heuristic*

ILP *Integer Linear Programming*

LB *Lower Bound*

LPR *Linear Programing Relaxation*

LR *Lagrangian Relaxation*

MILP *Mixed-Integer Linear Programming*

NPM *Nested Partition Method*

PGA *Permuted-based Genetic Algorithm*

PSGA *Primary-Secondary Genetic Algorithm*

PTAS *Polynomial Time Approximation Schemes*

RCL *Restricted Candidate List*

RS *Random Search*

SBH *Static Based Heuristic*

TAD *Total Absolute Deviation*

TADD *Total Absolute Deviation of job completion times about the correspondig Due dates*

TCE *Total Cabono Emission*

UB *Upper Bound*

UPM *Unrelated Parallel Machine Problem*

UPMR *Unrelated Parallel Machine Problem with Renewable Resource Constraint*

UPMSR *Unrelated Parallel Machine Problem with Setup Dependent and Renewable Resource Constraint*

VND *Variable Neighborhood Descent*

VNS *Variable Neighborhood Search*

WoS *ISI Web of Science*

SUMÁRIO

1 INTRODUÇÃO

Milhares de empresas buscam diariamente obter um diferencial competitivo frente aos seus concorrentes, procurando desenvolver práticas que as coloquem em posição de vantagem no mercado em que atuam. Pela percepção dos clientes, inúmeros são os critérios que podem diferenciar uma empresa das demais. Preço e qualidade normalmente são as principais características procuradas pelos consumidores.

Para que os preços consigam ser competitivos, é importante que as empresas otimizem o uso dos seus recursos produtivos, buscando a excelência na gestão de operações. Desta forma, os custos fixos são diluídos em uma produção mais volumosa e, consequentemente, os preços podem ficar mais atrativos para os clientes. Neste sentido há um interesse crescente em estudos que visem dar apoio na tomada de decisão sobre Problemas de Programação da Produção, tais como adoção de sistemas de produção, ajustes de capacidade, dimensionamento de lotes, e sequenciamento de produção.

Os conceitos de problemas de programação da produção são utilizados para resolver diversos problemas, como por exemplo, o problema de distribuição de recursos em um *cluster computacional*. Pesquisadores geralmente devem executar diversos testes computacionais, necessitando assim, muitos recursos, como vários computadores com bons processadores, que podem fornecidos por estes *clusters*.

Este trabalho apresenta uma contribuição para a resolução do problema de sequenciamento em máquinas paralelas não relacionadas, com restrição de recursos renováveis (UPMR – *Unrelated Parallel Machine Problem with Renewable Resource Constraint*) e do problema de sequenciamento em máquinas paralelas não relacionadas, com *setup* dependente e restrição de recursos renováveis (UPMSR – *Unrelated Parallel Machine Problem with Setup Dependent and Renewable Resource Constraint*). Ambos os problemas são variantes do Problema de Sequenciamento em Máquinas Paralelas não Relacionadas (UPM - *Unrelated Parallel Machine scheduling problem*). Vale ressaltar que, o conceito do termo recursos renováveis é recursos, disponíveis e compartilhados por todas as máquinas, que são reutilizados após a execução de uma tarefa.

O UPM consiste na ordenação de um conjunto de tarefas em um conjunto de máquinas dispostas em paralelo, onde as tarefas podem ser processadas por qualquer uma das máquinas e o tempo de processamento das tarefas pode variar de uma máquina para outra. Normalmente o objetivo deste problema é a minimização do *makespan*, ou seja, terminar a última tarefa do conjunto no menor instante de tempo

possível.

Ambas as variantes estudadas neste trabalho consideram a utilização de recursos renováveis no processamento das tarefas. Para Błażewicz, Brauner e Finke (2004), recursos podem ser considerados objetos, operadores ou ferramentas, os quais são necessário para o processamento de uma tarefa além da máquina onde será processada. Estes recursos estão disponíveis em quantidade limitada e que, após sua utilização no processamento de uma tarefa, podem ser reutilizados na execução da próxima. Isto torna o modelo mais próximo à realidade complexa das empresas de manufatura e possibilita uma solução mais adequada ao problema. Utilizando novamente o exemplo de um *cluster computacional*, em que cada teste a ser executado pode ser considerado como uma tarefa, as configurações de memória e processamento requisitadas são os recursos necessários, e as máquinas são os computadores do *cluster*. O grande desafio é sequenciar todos estes testes de modo a otimizar a utilização dos recursos disponíveis pelo *cluster*.

De acordo com Lenstra, Rinnooy Kan e Brucker (1977) o problema de máquinas paralelas é NP-*Hard* até mesmo na sua versão mais simples, considerando duas máquinas paralelas idênticas. Para resolver problemas computacionalmente são utilizadas diversas abordagens como métodos exatos, métodos heurísticos e até mesmo a hibridização de ambos os métodos.

Métodos exatos tem por suas principais vantagens a exploração sistemática do espaço de busca, a garantia de uma solução ótima e a eficiência em instâncias de pequeno porte. Por outro lado, suas principais desvantagens são o seu custo computacional extremamente elevado, a ineficiência computacional e tempo de resolução alto para instâncias de grande porte e para problemas considerados NP-*Hard*. Para resolução de métodos exatos são geralmente utilizadas modelagem de programação linear inteira mista (MILP - *Mixed-Integer Linear Programming*) ou de programação por restrição (CP - *Constraint programming*).

Por sua vez, abordagens heurísticas também possuem suas vantagens e desvantagens. Entre suas principais vantagens se encontram a prova empírica de encontrar soluções boas em um tempo extremamente curto e a exploração das características do problema em busca de uma performance melhor. No entanto, suas desvantagens são não comprovar a otimalidade de uma solução encontrada e a falta de garantia da exploração completa do espaço de busca. Heurísticas como o *greedy randomized adaptive search procedure* (GRASP), desenvolvido por Feo e Resende (1995), são poderosas ferramentas para encontrar soluções aproximadas e de forma iterativa. Além do mais, são muito utilizados por outros métodos para geração de conjuntos de soluções, como é o caso do *path-relinking*. Glover (1997), desenvolveu esta estratégia que conecta duas soluções com o propósito de utilizar movimentos similares às buscas locais para

encontrar as melhores soluções na trajetória desta conexão.

Diversos estudos buscaram aprimorar o *path-relinking*, utilizando de características elementares de outros métodos. Binato, Faria Jr. e Resende (2001) aplicaram os conceitos de aleatoriedade do GRASP e introduziram o *Greedy Randomized Adaptive Path-Relinking* (GRAPR). Já Resende e Werneck (2004) introduziram o *Evolutionary Path-Relinking* (EPR), um método de pós-otimização para aprimorar a solução obtida pelo GRAPR.

Portanto, a primeira contribuição deste trabalho é desenvolver uma abordagem do GRAPR com o EPR na fase de pós-otimização, aqui denominado como *Greedy Randomized Adaptive Evolutionary Path-Relinking* (GRAEPR) para o problema UPMR. Além disso, dado a eficiência da utilização de métodos exatos em conjunto com métodos heurísticos, estabelecida pelos trabalhos anteriores presentes na literatura, também foi desenvolvida uma abordagem híbrida entre o GRAEPR, e o modelo de programação por restrição desenvolvido por Fleszar e Hindi (2018).

A segunda contribuição deste trabalho é a estender a definição do problema acrescentando dependência de *setup* entre as tarefas, apresentando o problema de sequenciamento em máquinas paralelas não relacionadas, com *setup* dependente e restrição de recursos renováveis (UPMSR). Dado que o UPMSR é um problema ainda não explorado na literatura, foram desenvolvidos um modelo de programação linear inteira mista e um modelo de programação por restrição para resolução. Além disso, ambas as abordagens utilizando o GRAEPR desenvolvido nesta pesquisa foram modificadas e aplicadas. Também, foi realizada uma expansão da abordagem de Fleszar e Hindi (2018) para o problema com *setup* dependente. De acordo com os resultados obtidos, as abordagens híbridas são superiores em relação aos métodos exatos. Ademais, a utilização apenas do GRAEPR pode fornecer soluções de qualidade em tempos computacionais mínimos.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é desenvolver um *Greedy Randomized Adaptive Evolutionary Path-Relinking* (GRAEPR) para solucionar o problema de sequenciamento em máquinas paralelas não relacionadas, com restrição de recursos renováveis. Além disso, expandir este problema para o problema de máquinas paralelas não relacionadas, com *setup* dependente e com restrição de recursos renováveis, apresentando possíveis abordagens de solução.

1.2 OBJETIVO ESPECÍFICO

Os objetivos específicos deste trabalho são:

- Introduzir o problema de sequenciamento em máquinas paralelas não relacionadas, com *setup* dependente e restrição de recursos renováveis (UPMSR) e adaptar as instâncias existentes para este problema;
- Desenvolver um modelo de programação linear inteira mista para o problema UPMSR;
- Desenvolver um modelo de programação por restrição para o o problema UPMSR;
- Apresentar uma abordagem do *Greedy Randomized Adaptive Evolutionary Path-Relinking*, baseada no *path-relinking* proposto por Glover (1997), para solucionar o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR) e o problema de máquinas paralelas não relacionadas, com *setup* dependente e restrição de recursos renováveis (UPMSR);
- Apresentar uma abordagem híbrida, utilizando o *Greedy Randomized Adaptive Evolutionary Path-Relinking* e modelos de programação por restrição para ambos os problemas. Para o problema UPMR será utilizado o modelo proposto por Fleszar e Hindi (2018) e para o problema UPMSR será utilizado o modelo desenvolvido nesta pesquisa;
- Adaptar a abordagem híbrida de Fleszar e Hindi (2018) para o problema UPMSR, utilizando o modelo de *constraint programming* desenvolvido.

1.3 JUSTIFICATIVA DO TRABALHO

De modo geral, trabalhos presentes na literatura sobre o problema de máquinas paralelas não relacionadas ainda possuem diversas classificações ainda não exploradas. Deste modo, a importância deste trabalho se deve à expansão do problema de sequenciamento em máquinas paralelas não relacionadas com restrição de recursos renováveis, considerando também *setup* dependente.

1.4 LIMITAÇÕES DO TRABALHO

Com base na complexidade de resolução destes problemas, determinado por Lenstra, Rinnooy Kan e Brucker (1977) como NP-*Hard*, as limitações existentes neste trabalho decorrem do elevado tempo computacional necessário para a obtenção de respostas satisfatórias. Embora não se alcance a solução ótima em todos os casos, um limite máximo de tempo será pré-determinado em todos os testes realizados.

1.5 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte forma:

- Capítulo 2: Apresenta as classificações e nomenclaturas do problema de máquinas paralelas. Apresenta a definição do problema clássico de máquinas paralelas não relacionadas (UPM). Define o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR) e o problema de máquinas paralelas não relacionadas com *setup* dependente e restrição de recursos renováveis (UPMSR), ambos abordados nesta pesquisa;
- Capítulo 3: Apresenta uma análise e classificação dos trabalhos presentes na literatura, com interesse particular em problemas que abordem restrição de recursos. Por fim, são apresentados as pesquisas correlatas a este trabalho;
- Capítulo 4: Apresenta os métodos de resolução propostos nesta pesquisa;
- Capítulo 5: Apresenta os experimentos computacionais. Primeiramente são apresentadas as instâncias do problema UPMR e os resultados obtidos são apresentados, comparados e discutidos, em relação aos resultados existentes na literatura. Em seguida, são apresentadas as adaptações realizadas para gerar as instâncias para o problema UPMSR e os resultados obtidos são comparados e discutidos.
- Capítulo 6: Descreve as conclusões e discute perspectivas para futuros trabalhos.

2 DEFINIÇÃO DO PROBLEMA

O problema de sequenciamento de máquinas paralelas consiste em um problema clássico na área de Pesquisa Operacional. Neste capítulo são apresentados a definição clássica deste problema e suas classificações. Da mesma forma, também são apresentadas as definições dos problemas escopo desta dissertação.

2.1 PROBLEMA DE SEQUENCIAMENTO DE MÁQUINAS PARALELAS

O problema de sequenciamento de máquinas paralelas está entre as áreas mais estudadas da literatura (EDIS; OGUZ; OZKARAHAN, 2013). Este problema pode ser descrito como, o planejamento do sequenciamento de um conjunto de tarefas J processadas em um conjunto de máquinas M, a fim de garantir a execução de todas as tarefas em um período de tempo razoável (CHENG; SIN, 1990). Ou seja, consiste na ordenação de um conjunto de tarefas J em um conjunto de máquinas M dispostas em paralelo, onde as tarefas podem ser processadas por qualquer uma das máquinas e o tempo de processamento das tarefas é variável entre as máquinas, de acordo com a configuração de máquinas pré-determinada. Também, cada tarefa deve ser realizada em apenas uma máquina e cada máquina deve efetuar apenas uma tarefa por vez. Normalmente o objetivo deste problema é a minimização do *makespan*, ou seja, terminar a última tarefa do conjunto no menor instante de tempo possível. Além disso, deve satisfazer as condições específicas do tipo de problema, como a configuração de máquinas, as características das tarefas e a função objetivo.

2.1.1 Configuração de Máquinas

Para abordar problemas de sequenciamento, segundo Chen, Potts e Woeginger (1998) existem dois tipos de configuração de máquinas, com suas respectivas ramificações, que podem processar as tarefas disponíveis.

- *Única Etapa*: As tarefas necessitam apenas de um processamento, ou seja, serão processadas apenas uma vez por alguma máquina;
  - Única Máquina (*Single Machine*): Todas as tarefas devem ser executadas em apenas uma máquina;
  - Máquinas Paralelas (*Parallel Machine*): Existem algumas máquinas e todas as tarefas podem ser executadas em qualquer uma delas;
    - Idênticas (*Identical*): O tempo de processamento das tarefas independe da máquina em que a tarefa está sendo processada;

    * Não Relacionadas (*Unrelated*): O tempo de processamento das tarefas depende da máquina em que a tarefa está sendo processada;
    * Uniformes (*Uniform*): O tempo de processamento das tarefas é independente em relação à máquina em que a tarefa está sendo processada. No entanto, a velocidade de processamento é variável, de acordo com a máquina em que está sendo executada.

- *Múltiplas Etapas*: Os sistemas de Múltiplas Etapas consideram etapas em que as tarefas devem ser processadas;
  - *Flow Shop*: Todas as tarefas têm a mesma sequencia de processamento no conjunto de máquinas;
  - *Open Shop*: A sequencia de processamento das tarefas não é especificada;
  - *Job Shop*: Cada tarefa possui sua própria ordem de processamento.

Segundo Cheng e Ding (1999), todas estas configurações dos problemas de máquinas paralelas são NP-*Hard* (NP-Difícil), ou seja, todos são considerados problemas cujo o tempo computacional tem um aumento exponencial em relação ao aumento do tamanho do problema. Dentre as configurações evidenciadas, o presente trabalho trata do problema de Máquinas Paralelas Não Relacionadas.

2.1.2 Características das Tarefas

Outro fator importante ao classificar o problema de sequenciamento de máquinas são as características das tarefas. De acordo com Chen, Potts e Woeginger (1998) as principais características são definidas como as seguintes:

- Tempo de Processamento(P): O tempo em que uma tarefa leva para ser processada, de acordo com o tipo de configuração de máquinas;
- Tempo de *Setup* (S): Tempo de preparo de uma tarefa antes de ser iniciada, este tempo pode ser dependente ou independente da tarefa anterior;
- Disponibilidade (r): Todas as tarefas podem estar disponíveis no instante inicial do sequenciamento, ou cada tarefa pode ser restrita ao seu *release date*, ou seja, o instante em que a tarefa se torna disponível para o seu processamento;
- Prazo de conclusão (d): As tarefas podem possuir prazo para serem concluídas, também conhecido como *due date*;
- Elegibilidade de máquina (Eg): As tarefas possuem máquinas específicas em que podem ser processadas;

- Dependência: As tarefas podem possuir dependência de processamento, ou seja, existem restrições, previamente determinadas, em relação à sequencia em que as tarefas devem ser executadas;
- Interrupções: Também conhecido como processo de preempção, o processamento de qualquer tarefa pode ser interrompido a qualquer momento e retomado instantes mais tarde, na mesma ou em outra máquina;
- Informações do sequenciamento: As informações sobre o sequenciamento podem estar totalmente disponíveis antes de iniciar o sequenciamento (*off-line)* ou podem ser disponibilizadas durante o sequenciamento (*on-line)*;
- Recursos (Res): As tarefas podem necessitar de um ou mais tipos de recursos para serem processadas ou em sua preparação. De acordo com Błażewicz, Ecker et al. (2007) as restrições dos recursos podem ser classificadas como:
  - Não Renováveis: A restrição é em relação ao consumo total dos recursos, ou seja, uma vez utilizado por uma tarefa, o recurso não poderá ser utilizado novamente;
  - Renováveis: A restrição é em relação ao total de recursos utilizados a cada instante de tempo, ou seja, os recursos são utilizados a cada instante de tempo e após a utilização por uma tarefa, pode ser reutilizado por outra tarefa;
  - Restrição Dupla: São considerados ambas restrições, renováveis e não renováveis.

Dentre as características das tarefas apresentadas, o presente trabalho aborda tempo de processamento de cada tarefa dependente da máquina em que será processada, utilização de recursos renováveis durante o processamento das tarefas e a disponibilidade de todas as informações antes de iniciar o sequenciamento, ou seja, *off-line*.

2.1.3 Função Objetivo

Além das configurações de máquinas e características das tarefas, outro fator importante a ser analisado é o critério de otimização ou função objetivo. Em problemas de sequenciamento em máquinas paralelas todos os critérios de otimização avaliam, de alguma forma, o instante de conclusão (C_j) de cada tarefa j. Os critérios mais utilizados são apresentados a seguir:

- *Makespan* ($C_{max} = max_j\{C_j\}$): Instante máximo do término de processamento de todas as tarefas;

- Tempo total de conclusão das tarefas $\left(\sum_j C_j\right)$: Soma de todos os tempos de conclusão (C_j) de todas as tarefas;
- Adiantamento máximo $(E_{max} = max_j \{E_j\})$: Valor máximo de adiantamento das tarefas (E_{max}), ou seja, a diferença máxima entre o *due date* (d_j) de uma tarefa j e o seu respectivo instante conclusão (C_j). Para cada tarefa j, é calculado seu possível adiantamento E_j, pela diferença entre o *due date* (d_j) e o instante de término de uma tarefa (C_j), tal que, $E_j = max\{d_j - C_j, 0\}$;
- Adiantamento total $\left(\sum_j E_j\right)$: Soma do adiantamento de todas as tarefas (E_j). O adiantamento, para cada tarefa j, é calculado pela diferença entre o *due date* (d_j) e o instante de término de uma tarefa (C_j), tal que, $E_j = max\{d_j - C_j, 0\}$;
- Atraso máximo $(L_{max} = max_j\{L_j\})$: Valor máximo de atraso das tarefas (L_{max}), ou seja, a diferença máxima entre o instante de término de uma tarefa j e o seu respectivo *due date*. Para cada tarefa j, é calculado seu possível atraso L_j, pela diferença entre o instante de término (C_j) e o seu respectivo *due date* (d_j), tal que, $L_j = max\{C_j - d_j, 0\}$;
- Atraso total $\left(\sum_j L_j\right)$: Soma do atraso de todas as tarefas (L_j). O atraso de cada tarefa j é calculado pela diferença entre o instante de término e seu respectivo *due date* (d_j), tal que, $L_j = max\{C_j - d_j, 0\}$;
- Quantidade de Tarefas Atrasadas $\left(\sum_j U_j\right)$: a quantidade de tarefas que sofreram atraso. Tal que, $U_j = 1$, se $C_j > d_j$; $U_j = 0$, caso contrário.

A função objetivo utilizada neste trabalho é a minimização do instante máximo do término de processamento de todas as tarefas, ou seja, a minimização do *makespan* (C_{max}).

2.1.4 Nomenclatura

Graham et al. (1979) introduziram uma nomenclatura para classificar os problemas de sequenciamento em máquinas paralelas. Esta nomenclatura foi expandida por Błażewicz, Ecker et al. (2007) e é conhecida como classificação de 3 campos $\alpha|\beta|\gamma$, pois abrange as três condições de problemas de máquinas paralelas, configuração de máquina (α), características das tarefas (β) e função objetivo (γ).

O primeiro campo, $\alpha = \{\alpha_1, \alpha_2\}$, representa as possíveis configurações de máquina. O parâmetro $\alpha_1 \in \{$Ø, $P, Q, R, O, F, J\}$ representa o tipo de máquinas utilizadas:

- $\alpha_1 =$ Ø[1] : Única máquina;

[1] Nesta notação Ø significa valor nulo, que pode ser omitido ao utilizar a nomenclatura.

- $\alpha_1 = P$: Máquinas paralelas idênticas;
- $\alpha_1 = Q$: Máquinas paralelas uniformes;
- $\alpha_1 = R$: Máquinas paralelas não relacionadas;
- $\alpha_1 = O$: *Open Shop*;
- $\alpha_1 = F$: *Flow Shop*;
- $\alpha_1 = J$: *Job Shop*.

O parâmetro $\alpha_2 \in \{\text{Ø}, k\}$ representa a quantidade de máquinas a serem consideradas no problema:

- $\alpha_2 = \text{Ø}$: Quantidade de máquinas variável;
- $\alpha_2 = k$: Quantidade de máquinas igual a k.

O segundo campo, $\beta = \{\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}\}$ representa as características das tarefas. O parâmetro $\beta_1 \in \{\text{Ø}, pmtn\}$ indica a possibilidade das tarefas serem interrompidas durante seu processamento, também conhecido como processo de preempção:

- $\beta_1 = \text{Ø}$: Preempção não é permitida;
- $\beta_1 = pmtn$: Preempção é permitida.

O parâmetro $\beta_2 \in \{\text{Ø}, res\lambda\delta\rho\}$ caracteriza a utilização de recursos:

- $\beta_2 = \text{Ø}$: Recursos não são necessários;
- $\beta_2 = res\lambda\delta\rho$: Existem restrições de recursos de acordo com os seguintes parâmetros, quantidade de tipos de recursos distintos (λ), quantidade de recursos disponíveis (δ) e quantidade de recursos necessários (ρ):

  O parâmetro $\lambda \in \{\cdot, k\}$ representa a quantidade de tipos de recursos distintos:

  - $\cdot$: A quantidade de tipos de recursos distintos é arbitrária;
  - k: A quantidade de tipos de recursos distintos é igual à k.

  O parâmetro $\delta \in \{\cdot, k\}$ denota a quantidade de recursos disponíveis ou limite de recursos:

  - $\cdot$: A quantidade de recursos disponíveis é arbitrária;
  - k: A quantidade de recursos disponíveis é igual à k.

O parâmetro $\rho \in \{\cdot, k\}$ indica a quantidade de recursos necessários:

– $\cdot$: A quantidade de recursos necessários é arbitrária;

– k: A quantidade de recursos necessários é igual à k.

O parâmetro $\beta_3 \in \{\text{Ø}, prec, uan, tree, chain\}$ denota a existência de restrições de precedência das tarefas:

- $\beta_3 = \text{Ø}$: Tarefas são independentes;
- $\beta_3 = prec$: Existe restrições gerais de precedência de tarefas;
- $\beta_3 = uan$: Existe restrições de precedência em formato de grafos unidirecionais;
- $\beta_3 = tree$: Existe restrições de precedência em formato de grafos árvores;
- $\beta_3 = chain$: Existe restrições de precedência em formato de grafos em cadeia;

O parâmetro $\beta_4 \in \{\text{Ø}, r_j\}$ indica o instante de disponibilidade das tarefas:

- $\beta_4 = \text{Ø}$: Todas as tarefas estão disponíveis para processamento no instante 0;
- $\beta_4 = r_j$: Cada tarefa possuí seu próprio instante de disponibilidade, *release time*.

O parâmetro $\beta_5 \in \{\text{Ø}, p_j = p, \underline{p} \leq p_j \geq \bar{p}\}$ descreve o tempo de processamento das tarefas:

- $\beta_5 = \text{Ø}$: As tarefas possuem tempo de processamento arbitrário;
- $\beta_5 = (p_j = p)$: Todas as tarefas possuem o mesmo tempo de processamento, com valor igual à p;
- $\beta_5 = (\underline{p} \leq p_j \geq \bar{p})$: Nenhum tempo de processamento é menor que $\underline{p}$ e/ou maior que $\bar{p}$.

O parâmetro $\beta_6 \in \{\text{Ø}, d_j\}$ especifica se as tarefas possuem prazo para serem concluídas:

- $\beta_6 = \text{Ø}$: Nenhuma tarefa possuí prazo para conclusão;
- $\beta_6 = d_j$: Cada tarefa possuí seu próprio prazo de conclusão, *due date*.

O parâmetro $\beta_7 \in \{\text{Ø}, n \leq k\}$ descreve o número máximo de tarefas que compõem um *Job*, para problemas de sequenciamento *Job Shop*:

- $\beta_7 =$ Ø: O número máximo de tarefas é arbitrário ou não é um problema de sequenciamento *Job Shop*;
- $\beta_7 = (n \leq k)$: O número máximo de tarefas para cada *Job* não é maior que k.

O parâmetro $\beta_8 \in \{\text{Ø}, no - wait\}$, apenas para problemas de sequenciamento *Open Shop*, *Flow Shop* e *Job Shop*, indica a possibilidade da existência de tempos ociosos, no processamento de uma determinada tarefa, e entre duas máquinas. Esta característica é conhecida como *no-wait*:

- $\beta_8 =$ Ø: Podem existir tempos ociosos no processamento de uma tarefa, entre uma máquina e outra;
- $\beta_8 = no - wait$: Não são permitidos tempos ociosos no processamento de uma tarefa, entre uma máquina e outra, ou seja, após sua finalização em uma máquina a tarefa deve iniciar imediatamente seu processamento na máquina seguinte.

O parâmetro $\beta_9 \in \{\text{Ø}, Eg_{jm}\}$ denota a restrição de elegibilidade de máquinas:

- $\beta_9 =$ Ø: As tarefas podem ser executadas em qualquer máquina;
- $\beta_9 = Eg_{jm}$: As tarefas somente podem ser executadas nas máquinas em que são elegíveis.

O parâmetro $\beta_{10} \in \{\text{Ø}, S_{ij}\}$ especifica se o tempo de preparo, ou *setup time*, é considerado no problema:

- $\beta_{10} =$ Ø: Não existe tempo de *setup* entre duas tarefas;
- $\beta_{10} = S_{ij}$: Existe tempo de *setup* entre duas tarefas.

O terceiro campo, γ representa o critério de otimização, ou também chamado de função objetivo:

- $\gamma = C_{max}$: *Makespan*;
- $\gamma = \sum_j C_j$: Tempo total de conclusão das tarefas;
- $\gamma = E_{max}$: Adiantamento máximo;
- $\gamma = \sum_j E_j$: Adiantamento total;
- $\gamma = L_{max}$: Atraso máximo;
- $\gamma = \sum_j L_j$: Atraso total;
- $\gamma = \sum_j U_j$: Quantidade de Tarefas Atrasadas.

2.2 PROBLEMA DE MÁQUINAS PARALELAS NÃO RELACIONADAS

O problema de máquinas paralelas não relacionadas (UPM) consiste na ordenação de um conjunto de tarefas J em um conjunto de máquinas M dispostas em paralelo, onde as tarefas podem ser processadas por qualquer uma das máquinas e o tempo de processamento das tarefas pode variar de uma máquina para outra. Normalmente o objetivo deste problema é a minimização do makespan, ou seja, terminar a última tarefa do conjunto no menor instante de tempo possível.

O problema de máquinas paralelas não relacionadas é NP-*Hard* (LENSTRA; RINNOOY KAN; BRUCKER, 1977), ou seja, sua complexidade indica que o tempo computacional necessário, para obter soluções em sua otimalidade, aumenta exponencialmente de acordo com o tamanho do problema. Com isto, uma considerável quantidade de pesquisas sobre este tema tem sido desenvolvida utilizando diferentes métodos de resolução (PFUND; FOWLER; GUPTA, 2004).

Potts (1985) deixa claro que para o problema clássico de máquinas paralelas não relacionadas, dado uma designação tarefa-máquina, o sequenciamento das tarefas não altera o valor do *makespan*. Portanto, o problema pode ser reduzido a apenas um problema de designação das tarefas em máquinas. Assim, Potts (1985) apresenta um modelo de programação inteira mista (MILP) para resolver este problema. A TABELA 1 apresenta as notações utilizadas e o modelo é descrito em (2.1) - (2.5).

TABELA 1 – NOTAÇÕES UTILIZADAS PARA O MILP PARA O PROBLEMA DE MÁQUINAS PARALELAS NÃO RELACIONADAS

Conjuntos e índices	
J	Tarefas; $j \in J = 1...\|J\|$;
M	Máquinas; $m \in M = 1...\|M\|$;
Parâmetros	
P_{jm}	Tempo de processamento da tarefa j na máquina m;
Variáveis de Decisão	
C_{max}	instante máximo de conclusão das tarefas ou *makespan*;
X_{jm}	= 1, se a tarefa j é realizada na máquina m; = 0, caso contrário;

FONTE: Potts (1985).

Min C_{max} (2.1)

s.a.

$$\sum_{m=1}^{M} X_{jm} = 1 \qquad \forall j = 1..|J|; \tag{2.2}$$

$$\sum_{j=1}^{|J|} P_{jm} \cdot X_{jm} \leq C_{max} \qquad \forall m = 1..|M|; \tag{2.3}$$

$$C_{max} \in \mathbb{Z} \tag{2.4}$$

$$X_{jm} \in \mathbb{B} \qquad \forall j \in J; \forall m; \tag{2.5}$$

A função objetivo (2.1) minimiza o *makespan*. O conjunto de restrições (2.2) impõe que cada tarefa somente pode ser realizada em uma única máquina. O conjunto de restrições (2.3) realiza o cálculo do *makespan*, garantindo que seja maior ou igual ao tempo de processamento total de cada máquina. Os conjuntos de restrições (2.4) e (2.5) representam o domínio das variáveis de decisão.

No entanto, a utilização de métodos exatos se torna desafiadora conforme o tamanho das instâncias aumenta, devido ao crescimento exponencial do tempo computacional. Por isto, Powell e Chen (1995) investigaram abordagens de decomposição e aplicaram o método de decomposição de Dantzig-Wolfe. Outros autores também investigaram esquemas de aproximação de tempo polinomial (PTAS - *Polynomial Time Approximation Schemes*), que para dado $\epsilon > 0$, é possível garantir uma solução com fator $(1 + \epsilon)$ da otimalidade (HOROWITZ; SAHNI, 1976; DE; MORTON, 1980; DAVIS; JEFFREY, 1981; LENSTRA; SHMOYS; TARDOS, 1990).

Martello, Soumis e Toth (1997) propõem um algoritmo de aproximação que utiliza os valores de *Lower Bounds* obtidos através de um relaxação lagrangiana. Hariri e Potts (1991) também utilizaram uma versão relaxada do modelo (2.1) - (2.5), como primeira fase da heurística de duas fases elaborada em sua pesquisa.

Musier e Evans (1989) desenvolveram um método de melhoria heurística (HIM - *heuristic improvement method*), a qual gera soluções iniciais e por meio de uma busca local as aprimora. Glass, Potts e Shade (1994) avaliaram a performance de métodos de resolução, como busca gradiente, busca local, busca tabu, *simulated annealing* e algoritmo genético. Tandon, Cummings e Le Van (1995) desenvolveram um *simulated annealing* e o compararam com o método apresentado por Musier e Evans (1989), aplicando-o no problema de máquinas paralelas não relacionadas.

2.3 PROBLEMA DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM RESTRIÇÃO DE RECURSOS RENOVÁVEIS

O problema de sequenciamento em máquinas paralelas não relacionadas, com restrição de recursos renováveis (UPMR), é uma expansão do problema de sequenciamento em máquinas paralelas não relacionadas (UPM). Da mesma forma que o problema clássico, este problema consiste na ordenação de um conjunto de tarefas J, em um conjunto de máquinas M dispostas em paralelo, onde as tarefas

podem ser processadas por qualquer uma das máquinas e o tempo de processamento das tarefas pode variar de uma máquina para outra. Normalmente o objetivo deste problema é a minimização do *makespan*, ou seja, terminar a última tarefa do conjunto no menor instante de tempo possível. Além disso os seguintes pressupostos devem ser assumidos:

- Cada tarefa j deve ser realizada em apenas uma máquina m;
- Cada máquina m deve efetuar apenas uma tarefa j por vez;
- Os tempos de processamento (P_{jm}), de cada tarefa j, são valores fixos, que dependem da máquina m em que a tarefa será realizada;
- Nenhuma tarefa j pode ser interrompida durante a sua execução, ou seja, o processo de preempção não é permitido durante o sequenciamento;
- Cada tarefa j requer uma quantidade de recursos renováveis (Res_{jm}) para ser executada;
- Estes recursos são compartilhados por todas as máquinas e são limitados (Res_{max}), a cada instante do horizonte de tempo.

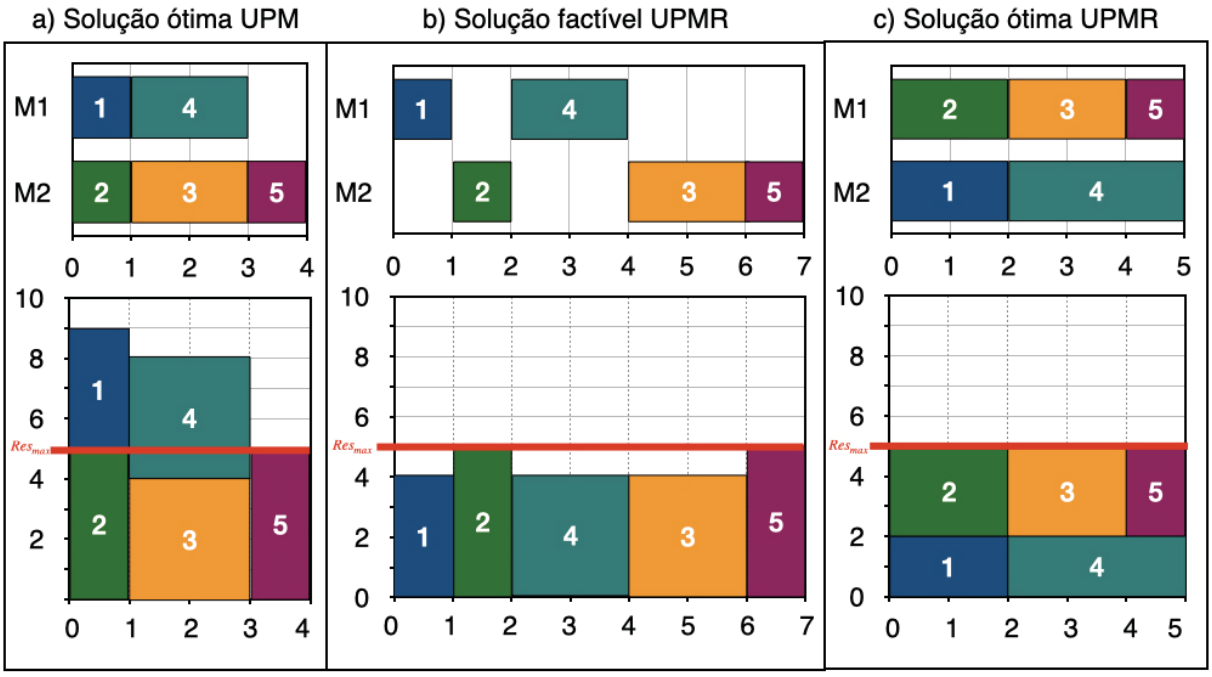
A principal característica, que diferencia o problema UPMR do problema clássico é a utilização de recursos renováveis. Neste problema é considerado que para realizar execução das tarefas existe uma quantidade de recursos necessários (Res_{jm}), os quais, após sua utilização em uma determinada tarefa, podem ser reutilizados. Estes recursos são compartilhados por todas as máquinas e são limitados (Res_{max}), a cada instante do horizonte de tempo. Podem ser considerados como recursos renováveis, operadores, ferramentas, consumo de energia ou capacidade de processamento em clusters. Este problema pode ser pode ser denotado por $R|res1 \cdot \cdot|C_{max}$, segundo a definição de Graham et al. (1979) e a expansão apresentada por Błażewicz, Ecker et al. (2007). Além disso, autores como Fanjul-Peyro, Perea e Ruiz (2015), Fanjul-Peyro, Perea e Ruiz (2017), Villa, Vallada e Fanjul-Peyro (2018), Fleszar e Hindi (2018) e Vallada, Villa e Fanjul-Peyro (2019), abordaram este tema anteriormente. Para melhor exemplificar este problema, Fanjul-Peyro, Perea e Ruiz (2017) apresentaram o exemplo a seguir:

Exemplo 1: É definido um conjunto de 5 tarefas, que pode ser realizado em uma das 2 máquinas com limite máximo de recurso igual a 5 ($Res_{max} = 5$). Considere também os seguintes tempos de processamento (P_{jm}) e recursos (Res_{jm}) necessários:

$$P_{jm} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \\ 2 & 2 \\ 2 & 3 \\ 1 & 1 \end{pmatrix} \qquad Res_{jm} = \begin{pmatrix} 4 & 2 \\ 3 & 5 \\ 3 & 4 \\ 4 & 2 \\ 2 & 5 \end{pmatrix} \tag{2.6}$$

Considerando, neste momento, que o exemplo acima deseja ser abordado apenas como o problema clássico de máquinas paralelas, ou seja, considerando apenas o tempo de processamento das tarefas nas máquinas. Para o UPM, o *makespan* na solução ótima obtida é de $C_{max} = 4$, conforme apresentada na FIGURA 1.a. No entanto, para o UPMR esta solução é infactível, pois no intervalo de tempo [0,3], os recursos necessários ultrapassam o limite estabelecido ($Res_{max} = 5$).

FIGURA 1 – SOLUÇÕES DA INSTÂNCIA APRESENTADA NO EXEMPLO 1.



FONTE: O autor (2020), adaptado de Fanjul-Peyro, Perea e Ruiz (2017).

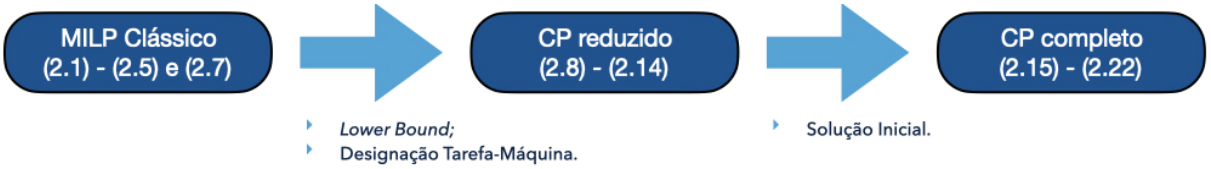
Mantendo a designação e o sequenciamento das tarefas obtidas na solução do UPM é possível encontrar factibilidade, para o problema com restrição de recursos, apenas aplicando um método de reparo na solução. Ao obter uma solução factível, em que o limite de recursos é respeitado, é possível notar a existência de tempos ociosos, os quais acarretam no aumento no *makespan*, $C_{max} = 7$ (FIGURA 1.b). Para este exemplo, é possível encontrar uma solução ótima para o problema UPMR. Esta

solução apresenta uma nova designação e um novo sequenciamento, com valor de *makespan* de $C_{max} = 5$, conforme a FIGURA 1.c.

O problema clássico de máquinas paralelas é NP-*Hard*, portanto, esta variante com restrição de recursos é ainda mais complexa para ser resolvida e também é NP-*Hard*. Pois, além de designar as tarefas às máquinas, onde serão executadas, o modelo possui restrições de recursos e deverá realizar estas designações de tal forma que o limite de recursos seja respeitado.

Recentemente, Fleszar e Hindi (2018), desenvolveram uma abordagem híbrida de três estágios para solucionar este problema. No primeiro estágio é realizado a definição de um *Lower Bound* e a designação tarefa-máquina inicial, que será utilizada para obter uma solução factível para o problema UPMR. O segundo estágio, um modelo, de programação por restrição, para a versão reduzida do problema. O objetivo deste modelo reduzido é definir o sequenciamento das tarefas, de acordo com a designação tarefa-máquina obtida no estagio anterior, obtendo uma solução factível para o UPMR. Por fim, a última fase é o modelo de *constraint programming*, que aborda o problema completo. Neste modelo, são utilizados o *Lower Bound*, obtido no primeiro estágio, e a solução do segundo estágio como *warm start*. Esta abordagem, desenvolvida por Fleszar e Hindi (2018), é apresentada na FIGURA 2.

FIGURA 2 – ABORDAGEM HÍBRIDA DE FLESZAR E HINDI (2018) PARA O UPMR



FONTE: O autor (2020), adaptado de Fleszar e Hindi (2018).

No primeiro estágio é realizado a definição do *Lower Bound*. Conforme observado por Fanjul-Peyro, Perea e Ruiz (2017), o modelo clássico de máquinas paralelas apresentado em (2.1) - (2.5), fornece um bom *Lower Bound* para o UPMR. Em seu estudo, Fleszar e Hindi (2018) também demonstrou que é possível acelerar a convergência do modelo ao adicionar a restrição apresentada por Grigoriev, Sviridenko e Uetz (2005), conforme a equação (2.7). Portanto, o resultado obtido do modelo (2.1) - (2.5) com a adição da restrição (2.7) é utilizado como *Lower Bound* das próximas fases desta abordagem.

$$\frac{\sum_{m}^{|M|}\sum_{j}^{|J|} Res_{jm}P_{jm}X_{jm}}{R_{max}} \leq C_{max} \tag{2.7}$$

O segundo estágio, é um modelo de programação por restrição, para uma versão reduzida do problema. No entanto, existem diversos solvers utilizados para resolver estes modelos, neste trabalho foi utilizado o CPLEX Studio versão 12.8. No entanto, modelos em CP codificados em um solver podem não ser utilizados em outros solvers, pois as funções com mesmo objetivo podem ser utilizadas de formas diferentes (GEDIK et al., 2017). Até onde se sabe, não é possível encontrar uma padronização para definir as variáveis, funções e restrições de modelos de programação por restrição. Para melhor entendimento do modelo, são apresentados a seguir os tipos de variáveis e as funções, que serão utilizadas no decorrer deste trabalho, de acordo com o manual IBM (2017):

- *IntervalVar:* Uma *IntervalVar* ou variável de intervalo representa um intervalo de tempo onde uma tarefa é executada e da qual a posição no horizonte de tempo é desconhecida. Estas variáveis possuem propriedades como, instante de início, instante de término e tempo de processamento. Além disso, uma variável de intervalo pode receber diversas variáveis opcionais, ou seja, o tempo de processamento desta varia de acordo com as variáveis opcionais. Assim sendo, a solução do modelo deverá selecionar apenas uma variável opcional para determinar os valores da variável de intervalo. Caso a variável de intervalo opcional esteja presente na solução, então valores serão atribuídos às suas propriedades e atribuídos à variável de intervalo. Caso contrário, as propriedades receberão valor 0 e não serão atribuídos. Portanto, podem ser consideradas variáveis de intervalo as tarefas que serão executadas, que possuem opções de tempo de processamento, de acordo com as máquinas disponíveis e que terão seus instantes, de início e término, definidos pelo modelo;

- *CumulFunction()*: Uma função acumulativa variável define a expressão que contabiliza o efeito de uma variável de intervalo na solução para cada instante de tempo;

- *IntervalSequenceVar*: Uma variável de sequência de intervalos representa um conjunto ordenado de variáveis de intervalo, ou seja, pode ser considerada como cada máquina disponível para sequenciar as tarefas;

- *TransitonDistance*: Variável de transição, representa o valor de tempo de transição entre duas variáveis de intervalo, ou seja, atribui o tempo de *setup*;

- *size( )*: Função que define o tamanho de uma variável de intervalo;

- *pulse( )*: Função que atribui o valor de efeito de uma variável de intervalo na solução;

- *endOf( )*: Função que retorna o instante de término de uma variável de intervalo;
- *noOverlap( )*: Função que garante a não sobreposição das variáveis de intervalo;
- *alternative( )*: Função que seleciona uma única variável de intervalo opcional para uma variável de intervalo;
- *optional_size( )*: Função que define o tamanho de variável de intervalo opcional.

Os parâmetros e as variáveis utilizadas por Fleszar e Hindi (2018), neste modelo de programação por restrição, para a versão reduzida, do problema de máquinas paralelas com restrição de recursos renováveis, são apresentados na TABELA 2.

TABELA 2 – NOTAÇÕES UTILIZADAS NO MODELO DE PROGRAMAÇÃO POR RESTRIÇÃO PARA O UPMR REDUZIDO

Conjuntos e índices	
J	Tarefas; $j \in J = 1...\|J\|$;
M	Máquinas; $m \in M = 1...\|M\|$;
Parâmetros	
m_j	Máquina m em que a tarefa j é pré-designada;
P_{jm_j}	Tempo de processamento da tarefa j na máquina m_j;
Res_{jm_j}	Recursos necessários para realizar a tarefa j na máquina m_j;
Res_{max}	Limite de recurso disponível para a realização das tarefas;
LB	*Lower Bound*;
Variáveis de Decisão	
C_{max}	Instante máximo de conclusão das tarefas ou *makespan*;
Y_j	Variável de intervalo relacionada à tarefa j;
$Resource$	Função Acumulativa;

FONTE: O autor (2020), adaptado de Fleszar e Hindi (2018)

$$\textbf{Min } C_{max} \tag{2.8}$$

s.a.

$$Y_j = size(P_{jm_j}) \qquad \forall j = 1..|J|; \tag{2.9}$$

$$noOverlap(Y_j\ \forall j, \text{tal que } m = m_j) \qquad \forall m = 1..|M|; \tag{2.10}$$

$$Resource = \sum_{j=1}^{|J|} pulse(Y_j, R_{jm_j}) \tag{2.11}$$

$$Resource \leq R_{max} \tag{2.12}$$

$$C_{max} \geq endOF(Y_j) \qquad \forall j = 1..|J|; \tag{2.13}$$

$$C_{max} \geq LB \tag{2.14}$$

Como este modelo reduzido, tem por objetivo somente realizar o sequenciamento das tarefas em máquinas pré-determinadas, é tomado como m_j a máquina m para a qual a tarefa j foi designada. A função objetivo (2.8) é minimizar o *makespan*. O conjunto de restrição (2.9) atribui o tempo de processamento P_{jm_j} à variável de intervalo Y_j relacionada à tarefa j. A não sobreposição das tarefas é garantida no conjunto de restrições (2.10). A restrição (2.11) define a função acumulativa $Resource$, como a soma dos recursos utilizados pela tarefa j na máquina pré-designada m_j. A garantia que os recursos respeitam o limite determinado é realizada pela restrição (2.12). O conjunto de restrições (2.13) garante que o *makespan* é o maior instante de conclusão de todas as tarefas. Por fim, a restrição (2.14) garante que o *makespan* respeite o valor de *Lower Bound*, afim de acelerar a convergência da solução.

A última fase é um modelo de *constraint programming* que considera o problema completo utiliza como parâmetros o *Lower Bound* e o *Warm Start*, definidos nas etapas anteriores. A TABELA 3 apresenta as funções, os parâmetros e as variáveis utilizadas neste modelo.

TABELA 3 – NOTAÇÕES UTILIZADAS NO MODELO DE PROGRAMAÇÃO POR RESTRIÇÃO PARA O UPMR

Conjuntos e índices	
J	Tarefas; $j \in J = 1...\|J\|$;
M	Máquinas; $m \in M = 1...\|M\|$;
Parâmetros	
P_{jm}	Tempo de processamento da tarefa j na máquina m;
Res_{jm}	Recursos necessários para realizar a tarefa j na máquina m;
Res_{max}	Limite de recurso disponível para a realização das tarefas;
LB	*Lower Bound*;
Variáveis de Decisão	
C_{max}	Instante máximo de conclusão das tarefas ou *makespan*;
X_{jm}	Variável de intervalo opcional; = Presente, se a tarefa j é executada na máquina m; = Ausente, caso contrário;
Y_j	Variável de intervalo relacionada à tarefa j;
$Resource$	Função Acumulativa;

FONTE: O autor (2020), adaptado de Fleszar e Hindi (2018)

Min C_{max} (2.15)

s.a.

$$X_{jm} = optional\_size(P_{jm}) \qquad \forall j = 1..|J|; \forall m = 1..|M|; \tag{2.16}$$

$$noOverlap(\forall j X_{jm}) \qquad \forall m = 1..|M|; \tag{2.17}$$

$$Y_j = alternative(\forall m X_{jm}) \qquad \forall j = 1..|J|; \qquad (2.18)$$

$$Resource = \sum_{j=1}^{|J|} \sum_{m=1}^{|M|} pulse(X_{jm}, R_{jm}) \qquad (2.19)$$

$$Resource \leq R_{max} \qquad (2.20)$$

$$C_{max} \geq endOF(Y_j) \qquad \forall j = 1..|J|; \qquad (2.21)$$

$$C_{max} \geq LB \qquad (2.22)$$

A função objetivo (2.15) tem como critério de otimização a minimização do *makespan*. O conjunto de restrições (2.16) atribuem o tempo de processamento da tarefa j na máquina m, à variável de intervalo opcional X_{jm}. As restrições (2.17) garantem que não existirá sobreposição das tarefas processadas na máquina m. O conjunto de restrição (2.18) é responsável por atribuir, através da função *alternative()*, uma única variável de intervalo opcional à variável Y_j. A restrição (2.19) define a função acumulativa $Resource$, como a soma dos recursos utilizados por uma tarefa j na máquina m. Além disso, é necessário que os recursos respeitem o limite determinado (2.20). O conjunto de restrições (2.21) garante que o *makespan* é o maior instante de conclusão das tarefas. Por fim, a restrição (2.22) garante que o *makespan* respeite o valor de *Lower Bound* definido nas fases anteriores.

2.4 PROBLEMA DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM *SETUP* DEPENDENTE E RESTRIÇÃO DE RECURSOS RENOVÁVEIS

O problema clássico de máquinas paralelas não relacionadas possuem algumas extensões de acordo com as características das tarefas que podem ser assumidas. Por muitas vezes, os sistemas de produção necessitam que as máquinas sejam reconfiguradas entre a execução de duas tarefas, gerando assim um tempo de preparo ou tempo de *setup*. Esta extensão, pode ser definida como o problema de máquinas paralelas não relacionadas com *setup* dependente. Incorporar a necessidade de reconfigurar as máquinas entre tarefas e a necessidade de utilizar recursos durante o processamento das mesmas define o problema de máquinas paralelas não relacionadas com *setup* dependente e com restrição de recursos renováveis (UPMSR). Dito isso, os pressupostos assumidos neste problema são apresentados a seguir:

- Cada tarefa j deve ser realizada em apenas uma máquina m;
- Cada máquina m deve efetuar apenas uma tarefa j por vez;
- Os tempos de processamento (P_{jm}), de cada tarefa j, são valores fixos, que dependem da máquina m em que a tarefa será realizada;

- Nenhuma tarefa j pode ser interrompida durante a sua execução, ou seja, o processo de preempção não é permitido durante o sequenciamento;
- Cada tarefa j necessita uma quantidade de recursos renováveis (Res_{jm}) para ser executada;
- Estes recursos são compartilhados por todas as máquinas e são limitados (Res_{max}), a cada instante do horizonte de tempo;
- Existe a necessidade de reconfiguração das máquinas entre duas tarefas;
- A máquina já está pronta para executar a primeira tarefa do sequenciamento;
- Os recursos não são utilizados durante o *setup* das tarefas.

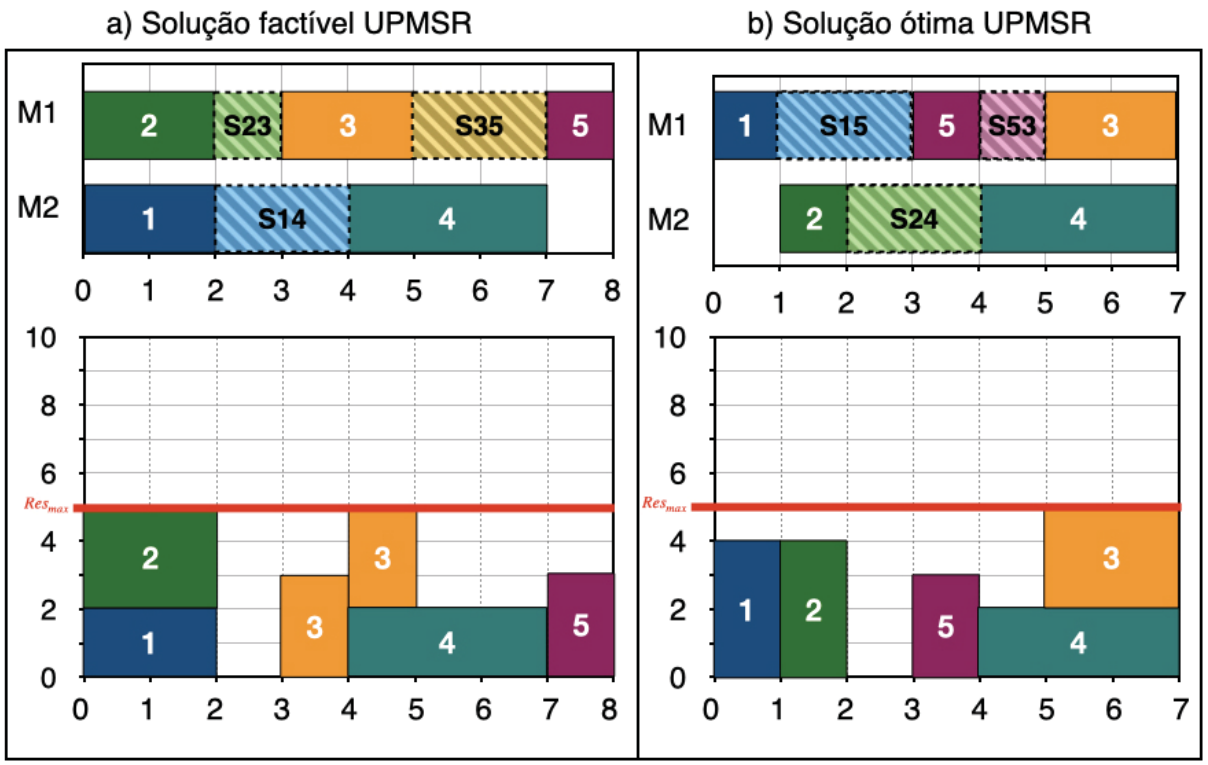
Segundo as definições de Graham et al. (1979) e Błażewicz, Ecker et al. (2007), este problema é denotado por $R|res1\cdot\cdot, S_{ij}|C_{max}$. O exemplo 2 é apresentado para melhor exemplificar e detalhar este problema.

Exemplo 2: São definidas 2 máquinas para 5 tarefas, com limite máximo de recurso igual a 5 ($Res_{max} = 5$). Considere também os tempos de processamento e recursos necessários apresentados no Exemplo 1 (2.6). Os tempos de *setup* são apresentados abaixo, onde S_{ij1} é a matriz dos tempos de *setup* na máquina 1, onde cada elemento representa o tempo de *setup* entre a tarefa i e a tarefa j na máquina 1. De forma análoga, S_{ij2} é a matriz dos tempos de *setup* na máquina 2, onde cada elemento representa o tempo de *setup* entre a tarefa i e a tarefa j na máquina 2:

$$S_{ij1} = \begin{pmatrix} - & 3 & 1 & 1 & 2 \\ 2 & - & 1 & 2 & 2 \\ 1 & 2 & - & 2 & 2 \\ 1 & 1 & 1 & - & 2 \\ 3 & 1 & 1 & 2 & - \end{pmatrix} \qquad S_{ij2} = \begin{pmatrix} - & 3 & 2 & 2 & 3 \\ 3 & - & 2 & 2 & 3 \\ 2 & 3 & - & 1 & 2 \\ 3 & 3 & 1 & - & 2 \\ 2 & 1 & 3 & 1 & - \end{pmatrix} \tag{2.23}$$

Utilizando os dados definidos no Exemplo 1 juntamente com os tempos de *setup* apresentados no Exemplo 2, é possível obter as soluções apresentadas na FIGURA 3. A FIGURA 3.a exibe uma solução factível do UPMSR, obtida através do sequenciamento apresentado pelo solução ótima do UPMR (FIGURA 1.c). Nela, é possível notar que há um aumento no valor do *makespan*, devido a inserção dos tempos de *setup*. Além disso, também é possível notar que durante o preparo das máquinas, os recursos não são necessários, pois os recursos abordados neste problema, são requeridos apenas durante o processamento das tarefas. A FIGURA 3.b exibe a solução ótima obtida para o problema UPMSR, com os parâmetros fornecidos por este exemplo.

FIGURA 3 – SOLUÇÕES DA INSTÂNCIA APRESENTADA NO EXEMPLO 2.



FONTE: O autor (2020).

Neste trabalho iremos abordar dois problemas, o problema de sequenciamento de máquinas paralelas não relacionadas com restrição de recursos renováveis (Seção 2.2) e o problema de máquinas paralelas não relacionadas com *setup* dependente e restrição de recursos renováveis (Seção 2.3), ambos com objetivo de minimizar o *makespan*.

3 REVISÃO DE LITERATURA

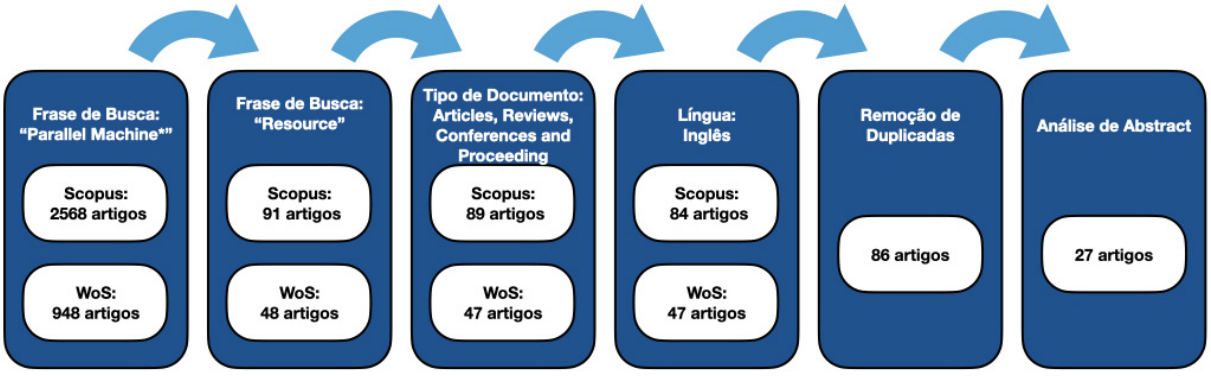
Neste capítulo é apresentada uma análise bibliométrica sobre este problema com enfoque na variante com restrição de recursos renováveis, onde são analisados os trabalhos correlacionados, suas classificações e principais abordagens utilizadas.

3.1 ANÁLISE BIBLIOMÉTRICA

A análise bibliométrica é um instrumento que permite realizar uma análise quantitativa da literatura. Segundo Gumpenberger e Gorraiz (2012) a análise bibliométrica foi desenvolvida como uma ferramenta para medir e monitorar a produção científica. Dentre as características normalmente investigadas, este trabalho irá dar ênfase em data de publicação, autoria, fontes de pesquisa e origem geográfica.

A amostra de artigos foi definida a partir da escolha de base de dados, identificação de palavras-chave, determinação do tipo de documentos publicados e idioma de publicação, além da exclusão de publicações duplicadas e não relevantes. A avaliação da relevância das publicações foi realizada através da leitura completa dos artigos. O fluxograma do processo de definição de amostra é apresentado na FIGURA 4.

FIGURA 4 – FLUXOGRAMA DO PROCESSO DE DEFINIÇÃO DE AMOSTRA



FONTE: O autor (2020).

Em relação à seleção das bases de dados, optou-se por utilizar as plataformas *ISI Web of Science* (WoS) e *Scopus*, por incluírem uma significante parte dos principais periódicos sobre o tema. Quanto à seleção dos dados, foram utilizadas duas expressões. A primeira a ser utilizada foi *"Parallel Machine"*, a qual resultou em 2568 artigos da base *Scopus* e 948 artigos da base WoS. A segunda expressão, *"Resources"*, foi selecionada devido à definição do problema selecionado para esta pesquisa. Esta expressão foi

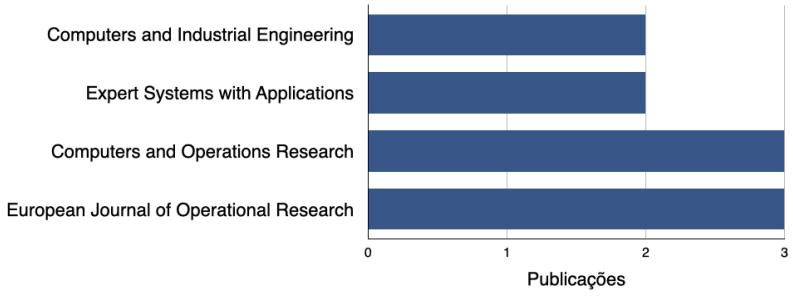
aplicada como filtro na primeira seleção, resultando em 91 artigos da base *Scopus* e 48 artigos da base WoS.

Em relação aos tipos de documentos (artigos de periódicos, artigos de conferências, livros, etc.) foi decidido por selecionar apenas nos formatos de artigos de periódicos, artigos de conferências, *reviews* e *proceedings*. Em geral estes tipos de publicação são considerados fontes seguras de pesquisa, devido ao alto rigor para serem publicados. O resultado obtido foi 89 artigos da base *Scopus* e 47 artigos da base *web of Science*.

Também foram selecionados apenas artigos de língua inglesa. Resultando assim em 84 artigos da base *Scopus* e 47 artigos da base *web of Science*. Para a finalização do procedimento de definição de amostra, foram excluídos 46 artigos contidos em ambas as bases, resultando em 86 artigos. Após a leitura e análise dos artigos, 59 foram excluídos devido a irrelevância para este trabalho. Consequentemente, um total de 27 artigos permaneceram para a análise bibliométrica.

Após a obtenção da amostra final buscou-se analisar a distribuição de artigos por periódico e conferências, a FIGURA 5 apresenta apenas periódicos com duas ou mais publicações. Nesta figura são destacados os periódicos *European Jounal of Operation Research* e o *Computers and Operations Research*, ambos com três publicações . Seguidos pelos periódicos *Computers and Industrial Engineering* e *Expert Systems with aplications*, ambos com duas publicações.

FIGURA 5 – DISTRIBUIÇÃO DAS PUBLICAÇÕES PELOS PRINCIPAIS PERIÓDICOS

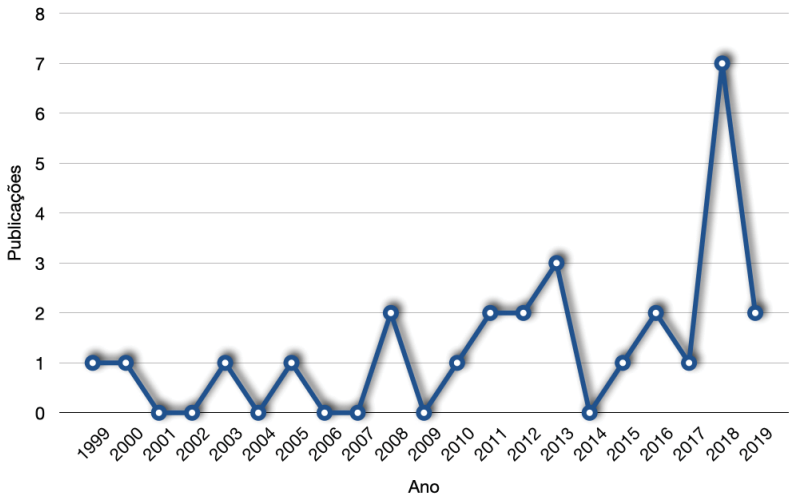


FONTE: O autor (2020).

Também buscou-se identificar tendências de aumento ou declínio do interesse em realizar estudos sobre o problema de Máquinas Paralelas com utilização de Recursos, classificando-as por ano de publicação (FIGURA 6). O desenvolvimento de trabalhos sobre este assunto inicia-se em 1999, sendo que nos anos seguintes, o volume de publicações apresenta um caráter cíclico entre nenhuma (anos 2001, 2002, 2004, 2006, 2007) e uma (anos 2000, 2003 e 2005) publicação. O crescimento de interesse pode ser notado entre os anos de 2010 e 2013, e sendo retomado a partir

de 2015. Os anos de 2013 e 2018 apresentaram picos nos volumes de publicações, sendo este último a apresentar o maior volume com sete artigos no total.

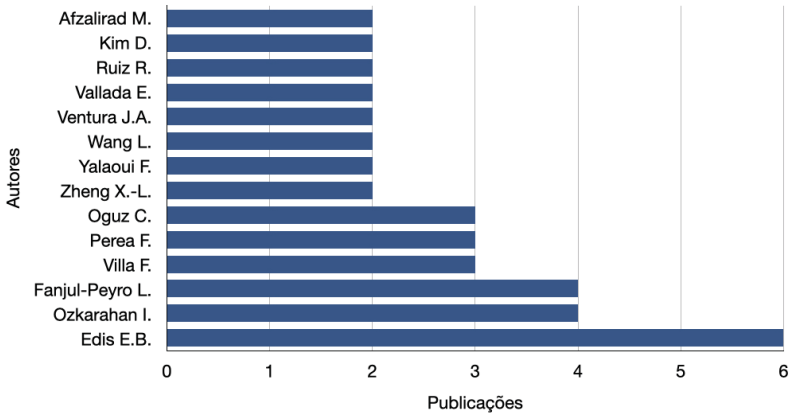
FIGURA 6 – EVOLUÇÃO DAS PUBLICAÇÕES AO LONGO DO TEMPO



FONTE: O autor (2020).

Em relação à distribuição das publicações pelos principais autores (FIGURA 7), a análise indicou que 14 autores possuem dois ou mais artigos publicados. Os autores com mais destaque foram E. B. Edis, da *Celar Bayar University*, Turquia, com seis publicações, L. Fanjul-Peyro, do *Grupro de Sistemas de Optmización Aplicada*, Espanha e I. Ozkarahan, da *Troy University*, Estados Unidos, ambos com quatro publicações. Dentre estes três autores destacados, E. B. Edis e I. Ozkarahan publicaram alguns trabalhos em conjunto sobre Máquinas Paralelas com Restrições de Recursos.

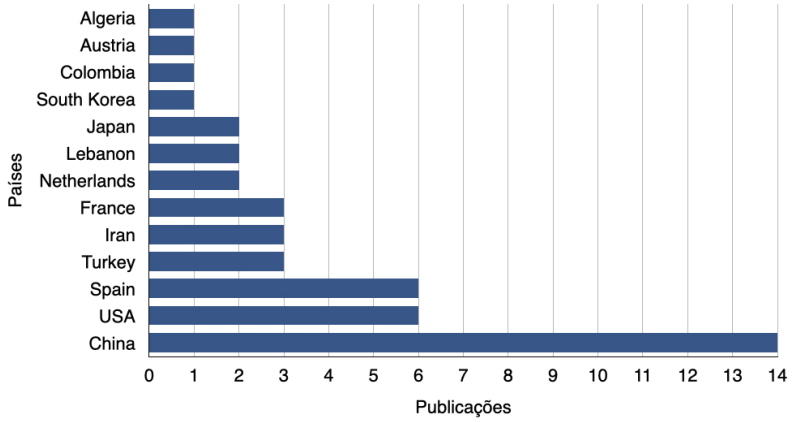
FIGURA 7 – DISTRIBUIÇÃO DAS PUBLICAÇÕES PELOS PRINCIPAIS AUTORES



FONTE: O autor (2020).

A estratificação dos autores por país de origem (FIGURA 8) mostra a clara predominância da China, seguida pelos Estados Unidos e pela Espanha. Para as representações geográficas os autores são representados por seu país de origem.

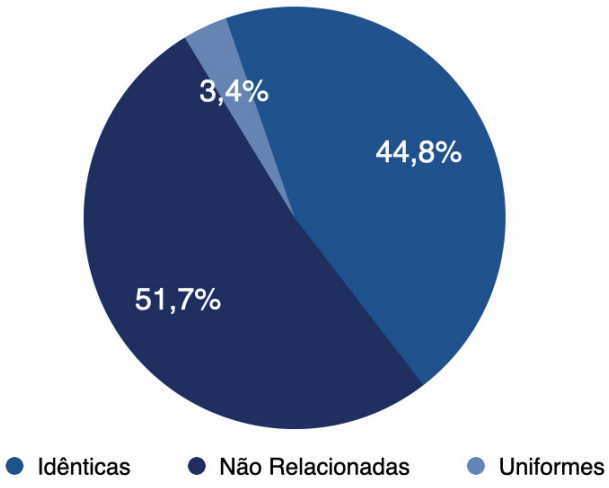
FIGURA 8 – DISTRIBUIÇÃO DOS AUTORES PELO PAÍS DE ORIGEM



FONTE: O autor (2020).

Por fim, os trabalhos são analisados em relação às três categorias de configuração de máquinas paralelas e os critérios de otimização utilizados. Vale ressaltar que, dentre os trabalhos avaliados, alguns deles abordam mais de um problema de máquinas paralelas com restrição de recurso, totalizando 29 problemas propostos. Portanto, as análises foram realizadas de acordo com o total de problemas propostos.

FIGURA 9 – DISTRIBUIÇÃO DAS PUBLICAÇÕES POR CONFIGURAÇÃO DE MÁQUINAS PARALELAS

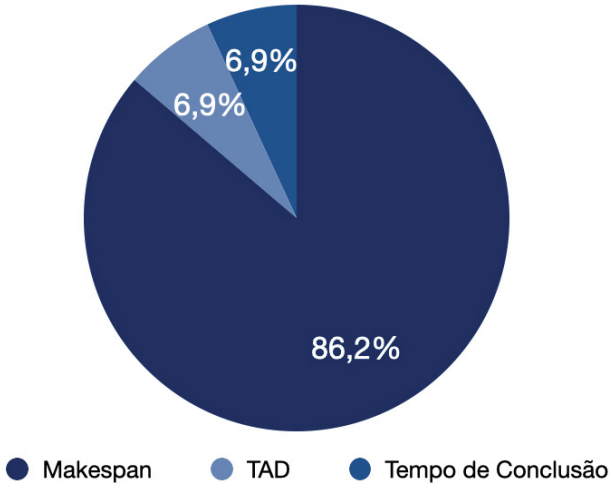


FONTE: O autor (2020).

De acordo com a amostra obtida, nota-se a predominância de trabalhos referentes às máquinas paralelas não relacionadas (51,5% ou 15 problemas propostos),

seguidos por publicações sobre máquinas paralelas idênticas (44,8% ou 13 problemas propostos) e máquinas paralelas uniformes (13,4% ou 1 problema proposto), conforme apresentado na FIGURA 9.

FIGURA 10 – DISTRIBUIÇÃO DAS PUBLICAÇÕES POR FUNÇÃO OBJETIVO



FONTE: O autor (2020).

Na FIGURA 10 verifica-se as funções objetivos utilizadas, sendo o instante máximo do término de processamento de todas as tarefas, também denotado como *makespan* predominante com 86,2% das publicações. Além disso, são utilizados outros dois critérios de otimização, ambos presentes em 6,9% dos problemas propostos. O total de tempo de conclusão de todas as tarefas e o Desvio Absoluto Total (TAD - *Total Absolute Deviation*) do *due-date*, ou seja, é a minimização conjunta do adiantamento total e do atraso total $\left(\sum_j E_j + L_j\right)$.

3.2 TRABALHOS CORRELATOS

Nesta sessão serão apresentados trabalhos correlatos ao problema de sequenciamento de máquinas, especificamente a variante com a utilização de recursos, selecionados a partir da análise bibliométrica apresentada na Seção 3.1.

Edis, Oguz e Ozkarahan (2013) apresentam uma revisão sistemática e uma discussão sobre os estudos já realizados relacionados ao problema de sequenciamento de máquinas paralelas com restrições de recursos. Neste trabalho foram apresentadas as classificações dos problemas, baseadas em configuração de máquina, função objetivo e características de recursos. Também é utilizada a caracterização de recursos de Błażewicz, Brauner e Finke (2004), em que recursos podem ou não ser caracterizados como recursos *input-output*, ou seja, os recursos são necessários antes ou depois do processamento de uma tarefa, ou seja se ele é necessário durante a preparação de uma máquina, ou no descarregamento da mesma.

TABELA 4 – CLASSIFICAÇÕES E ABORDAGENS ENCONTRADAS NA LITERATURA

Referência	Classificação	Abordagem
Daniels, Hua e Webster (1999)	$P\|res1 \cdot \cdot\|C_{max}$	Heurística de Decomposição *Tabu Search*
Ventura e Kim (2000)	$P\|res1 \cdot 1, p_j, d_j = d\|TAD$	MILP
Ventura e Kim (2003)	$P\|res \cdot \cdot\cdot, r_j, p_j = 1, d_j\|TADD$	Relaxação Lagrangeana
Grigoriev, Sviridenko e Uetz (2005)	$R\|res1 \cdot \cdot\|C_{max}$ $PD\|res1 \cdot \cdot\|C_{max}$	Algoritmo de Aproximação Algoritmo de Aproximação
Edis, Araz e Ozkarahan (2008)	$P\|res1 \cdot 2, Eg_{jm}, p_j = 1\|\sum C_j$	Relaxação Lagrangeana
Kellerer (2008)	$P\|res1 \cdot \cdot\|C_{max}$	Algoritmo de Aproximação
Qiong et al. (2010)	$P\|res1 \cdot \cdot, prec\|C_{max}$	*Ant Colony Optmization*
Edis e Oguz (2011)	$R\|res1 \cdot \cdot\|\sum C_j$	ILP, CP, CP + Relaxação Lagrangeana
Edis e Ozkarahan (2011)	$P\|res1 \cdot 2, Eg_{jm}\|C_{max}$	ILP, CP , ILP/CP
Edis e Oguz (2012)	$PD\|res1 \cdot \cdot\|C_{max}$ $R\|res1 \cdot \cdot\|C_{max}$	ILP, 2 Fases (ILP - CP) ILP, 2 Fases (ILP - CP)
Edis e Ozkarahan (2012)	$P36\|res1 \cdot 2, Eg_{jm}\|C_{max}$	2 Fases (IP-IP) 2 Fases (ILP - CP)
Belkaid, Sari e Yalaoui (2013)	$P\|res \cdot \cdot \cdot\|C_{max}$	ILP, *Genetic Algorithm*, *Genetic Algorithm + Local Search* *Constructive Heuristic*
Sun, Liu e Hou (2013)	$Q\|res1 \cdot \cdot\|C_{max}$	*Nested Partition Method*
Fanjul-Peyro, Perea e Ruiz (2015)	$R\|res1 \cdot \cdot\|C_{max}$	ILP, *Constructive Heuristic*
Afzalirad e Rezaeian (2016)	$R\|res \cdot \cdot\cdot, prec, r_j, Eg_{jm}, S_{ij}\|C_{max}$	MILP, *Genetic Algorithm*, *Artificial Immune System*
Zheng e Wang (2016)	$R\|res1 \cdot \cdot\|C_{max}$	ILP, *Fruit Fly Optimization Algorithm*
Fanjul-Peyro, Perea e Ruiz (2017)	$R\|res1 \cdot \cdot\|C_{max}$	ILP, Math Heuristc
Afzalirad e Shafipour (2018)	$R\|res \cdot \cdot\cdot, Eg_{jm}\|C_{max}$	MILP, *Genetic Algorithm* *Hybrid Genetic Algorithm*
Akbar e Irohara (2018)	$P\|res1 \cdot \cdot, S_{ij}\|C_{max}$	MILP, *Permuted-based Genetic Algorithm* *Random Search*
Arbaoui e Yalaoui (2018)	$R\|res1 \cdot \cdot\|C_{max}$	CP
Fleszar e Hindi (2018)	$R2\|res1 \cdot \cdot\|C_{max}$ $R\|res1 \cdot \cdot\|C_{max}$	MILP MILP + CP_{assgin} + CP_{FULL}
Fu et al. (2018)	$P\|res \cdot \cdot\cdot, p_j\|C_{max}$	Primary-Secondary Genetic Algorithm
Villa, Vallada e Fanjul-Peyro (2018)	$R\|res1 \cdot \cdot\|C_{max}$	GRASP
Zheng e Wang (2018)	$R\|res1 \cdot 1\|C_{max}, TCE$	*Fruit Fly Optimization Algorithm*
Vallada, Villa e Fanjul-Peyro (2019)	$R\|res1 \cdot \cdot\|C_{max}$	*Enriched Scatter Search* *Enriched Iterated Greedy*
Yepes-Borrero et al. (2020)	$R\|res1 \cdot \cdot, S_{ij}\|C_{max}$	GRASP +*Local Search*

FONTE: O autor (2020).

Para facilitar a classificação dos trabalhos correlatos, será utilizada a classificação introduzida por Graham et al. (1979), e depois expandida por Błażewicz, Ecker et al.

(2007) com suas definições de recursos, conforme apresentado na subseção 2.1.4. A TABELA 4 apresenta um resumo da classificação dos trabalhos relacionados de acordo com estas notações.

Daniels, Hua e Webster (1999) propõem um problema de sequenciamento de máquinas paralelas idênticas com recursos flexíveis, que podemos classificar como recursos não renováveis. Como abordagem de solução foi utilizado uma heurística de decomposição para criar uma solução inicial, seguida de um algoritmo de busca local, *Tabu Search*, com objetivo de aprimorar a solução.

Ventura e Kim (2000), em seu estudo sobre sequenciamento em máquinas paralelas idênticas, utilizam como critério de otimização minimizar o Desvio Absoluto Total (TAD) de um *due date* comum (d), definido como $TAD = \sum_j |max\{c_j - d, 0\} + max\{0, d - c_j\}|$, que também pode ser denotado como a soma entre o adiantamento e o atraso total da conclusão das tarefas. O *due date*, para o problema apresentado, é considerado como apenas um instante de tempo em comum, no qual todas as tarefas devem terminar. Também é considerada a utilização de um tipo de recurso renovável (limitado a uma unidade por tarefa). Para resolver o problema é apresentado um MILP. Dado a utilização de apenas uma unidade de recurso por tarefa, os autores provaram que é possível simplificar o modelo proposto utilizando algumas propriedades de dominância, e assim, obter melhores resultados computacionais.

Ventura e Kim (2003) deram continuidade ao seu estudo anterior, propondo dois problemas, sendo o segundo uma particularidade do primeiro. Nesta nova proposta, existem tipos de recursos distintos, onde cada tarefa necessita uma quantia de cada tipo de recurso requerida. Além disso, cada tipo de recurso tem seu respectivo limite de utilização por instante de tempo, ou seja, estes recursos são considerados renováveis. Também são considerados que o tempo de processamento das tarefas tem valor unitário. No segundo problema proposto, os autores consideram apenas um tipo de recurso, no entanto, as tarefas podem ou não utilizar uma unidade de recurso renovável. Em ambos os problemas cada tarefa tem seu respectivo *due date*. O critério de otimização utilizado é minimizar Desvio Absoluto Total de um *due date* correspondente de cada tarefa (TADD [1] - *Total Absolute Deviation of job completion times about the corresponding due dates*). Para tanto, o modelo é solucionado utilizando uma abordagem Relaxação Lagrangiana (LR - *Lagrangian Relaxation*) com a finalidade de encontrar o maior *Lower Bound* para o problema primal. Além disso, duas heurísticas construtivas são desenvolvidas, a primeira para encontrar uma solução inicial factível para o modelo primal e a segunda para converter as soluções obtidas no LRP em uma solução factível para o problema original e utilizá-la com *Upper Bound* (UB).

[1] Nomenclatura apresentada por Edis, Oguz e Ozkarahan (2013) com objetivo de diferenciar os trabalhos dos autores.

Grigoriev, Sviridenko e Uetz (2005) consideram o problema de máquinas paralelas não relacionadas com recursos renováveis distribuídos entre as tarefas com intuito de aprimorar seu tempo de processamento, ou seja, quanto mais recurso alocado em uma tarefa, menor será seu tempo de processamento. O objetivo definido é o de minimizar o *makespan*. Neste trabalho é proposto um algoritmo de aproximação (AA - *Approximation Algorithm*), utilizando uma abordagem de Relaxação Linear (LPR - *Linear Programing Relaxation*) é obtido um *Lower Bound* (LB), provando que a solução obtida tem um fator de $(4 + 2\sqrt{2})$ de distância da otimalidade. Além disso, os autores também abordam o problema específico de máquinas idênticas dedicadas e para este caso, provam que sua abordagem tem uma performance de $(3 + 2\sqrt{2})$.

Edis, Araz e Ozkarahan (2008) propõem um modelo de programação linear inteira (ILP - *Integer Linear Programming*) para o problema de máquinas paralelas idênticas com restrição de recursos renováveis e também considerando restrições de elegibilidade de máquina. Os autores utilizam da abordagem de LR com otimização de sub gradientes, para obter uma solução próxima do ótimo. Além disso, demonstram que sua abordagem fornece resultados eficientes com pequeno *gap* de otimalidade.

Kellerer (2008) apresenta uma abordagem para o problema de máquinas paralelas idênticas com tempo de processamento dependente dos recursos não renováveis, ou seja, o tempo de processamento depende da quantidade de recursos adicionada na tarefa. Foi desenvolvido um algoritmo de aproximação $(3.5 + \varepsilon)$, que segue uma abordagem similar à apresentada por Grigoriev, Sviridenko e Uetz (2005), no entanto a formulação apresentada é baseada no problema da mochila de múltipla escolha.

Qiong et al. (2010) apresentam um algoritmo de Otimização de Colonia de Formigas (ACO - *Ant Colony Optimization*), com o objetivo de minimizar o *makespan* no problema de máquinas paralelas idênticas com restrição de precedência de tarefas e de recursos renováveis, considerados neste trabalho como operadores.

Edis e Oguz (2011) apresentam um modelo em ILP e um modelo em CP para o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis. Além disso, também apresentam um modelo em CP relaxado, utilizando LR para relaxar a restrição de recursos do problema.

Edis e Ozkarahan (2011) abordam o problema de sequenciamento em máquinas paralelas idênticas com restrição de recurso renováveis e elegibilidade de máquinas. Para tanto, utilizaram três abordagens de solução: i) modelo de programação linear inteira (ILP); ii) *constraint programming* (CP); iii) combinação dos dois modelos (IP/CP). Foi também desenvolvido um procedimento de busca para ser utilizado nos modelos de *constraint programming*.

Edis e Oguz (2012) abordam o problema de sequenciamento de máquinas

paralelas idênticas com recursos renováveis, propondo três abordagens distintas para resolução deste problema. Os autores apresentam uma forma estendida do modelo ILP apresentado por Daniels, Hoopes e Mazzola (1996), adicionando dois conjuntos de restrições baseado no estudo de Grigoriev, Sviridenko e Uetz (2005). A terceira proposta, é a utilizar o modelo ILP apresentado por Grigoriev, Sviridenko e Uetz (2005), seguido de um modelo CP proposto pelos próprios autores. Além disso, é abordado o problema de sequenciamento de máquinas paralelas não relacionadas com recursos flexíveis, para este problema os autores apresentam um novo modelo ILP e utilizam uma abordagem sequencial, para o problema de máquinas paralelas não relacionada. Nesta abordagem o modelo ILP, apresentado por Grigoriev, Sviridenko e Uetz (2005), é aplicado sequencialmente com o modelo CP proposto anteriormente pelos autores.

Edis e Ozkarahan (2012) estudam uma aplicação real do problema de sequenciamento em máquinas paralelas idênticas com restrição de recurso e elegibilidade de máquinas e propuseram duas abordagens de solução em que o problema é particionado em dois sub problemas, *loading* e *scheduling*. Em ambas as abordagens, para resolver o sub problema de *loading* é utilizado o modelo de programação linear inteira (ILP) proposto por Edis e Ozkarahan (2011). Em ambas as abordagens os subproblemas são resolvidos sequencialmente, sendo o que diferencia uma abordagem da outra é a utilização de um modelo de programação inteira (IP/IP) ou de *constraint programming*(IP/CP) na fase de *scheduling*. Ambas as abordagens, assim como o modelo completo em IP foram testados em 36 instâncias geradas a partir do problema real estudado. Os resultados obtidos comprovam que as abordagens propostas são melhores que o modelo exato.

Belkaid, Sari e Yalaoui (2013) apresentam uma nova abordagem de ILP para o problema de máquinas paralelas idênticas com recursos não renováveis. Nesta nova abordagem, os autores exploram uma formulação baseada na posição das variáveis, indicando a posição de cada tarefa no sequenciamento. Além disso, propõem um algoritmo genético, com duas abordagens de solução, uma utilizando apenas o GA e outra com a aplicação de um *local search* para melhorar as soluções. Os autores também apresentam uma heurística construtiva generalizada (HE - *Heuristic*) para o problema de máquinas paralelas não relacionadas.

Sun, Liu e Hou (2013) pesquisam o problema de máquinas paralelas uniformes com restrição de recursos renováveis. Os autores propõem um MILP e um método de partição aninhada (NPM - *Nested Partition Method*) para resolução do problema.

Fanjul-Peyro, Perea e Ruiz (2015) abordam o problema de máquinas paralelas não relacionadas com recursos renováveis. Neste trabalho os autores propõem um ILP, mas dada a magnitude do problema, também é desenvolvida uma abordagem heurística de duas fases. Na primeira fase desta abordagem, os autores resolvem o

problema clássico de máquinas paralelas não relacionadas. Na segunda, eles fixam a designação tarefa-máquina obtida na solução da primeira fase e resolvem o problema com recursos renováveis. Os resultados mostraram que ao aumentar o número de tarefas a ser processadas a abordagem heurística tem melhor performance. No caso contrário, ao aumentar o número de máquinas disponíveis, o ILP apresenta ser melhor.

O estudo de Afzalirad e Rezaeian (2016) aborda diversos aspectos envolvidos no problema de sequenciamento em máquinas paralelas. No modelo proposto são considerados a utilização de recursos renováveis, o tempo de *setup* dependente da sequência e da máquina, a elegibilidade das máquinas, o instante de disponibilidade das tarefas e restrições de precedência entre tarefas. O MILP formulado foi validado com um exemplo numérico considerado pequeno, mas não foi considerado adequado para a resolução de problemas de grande porte, tendo em vista a complexidade e abrangência do modelo. Os autores, então, apresentaram duas abordagens metaheurísticas para a solução do problema, sendo um GA e um Sistema Imunológico Artificial (AIS - *Artificial Immune System*). Foram comparadas as soluções, dos três métodos apresentados, utilizando o valor de *Lower Bound* obtido pelo MILP como parâmetro de comparação. Nos resultados, o método AIS demonstrou uma performance melhor em relação aos outros métodos.

Zheng e Wang (2016) consideram em seu estudo um problema de sequenciamento em máquinas paralelas não relacionadas com restrição de recursos renováveis, cuja utilização está limitada a uma unidade de recurso por tarefa, sendo assim, a disponibilidade total de recursos é menor do que a quantidade de máquinas. Para este problema, apresentam um modelo de programação linear inteira mista (MILP) e um algoritmo de otimização adaptativo *fruit fly* de duas fases (FFOA - *fruit fly optimization algorithm)*. Na primeira fase, as tarefas, cujos tempos de processamento são menores, são designadas para as máquinas e são ordenadas do maior para o menor tempo de processamento em cada máquina. A solução obtida é utilizada como local central inicial para a busca pelas *fruit flies*, reduzindo assim o espaço de solução. A abordagem de solução demonstrou efetividade e eficiência na solução do problema proposto.

Fanjul-Peyro, Perea e Ruiz (2017) propõem o problema de sequenciamento em máquinas paralelas não relacionadas com restrição de recursos renováveis e apresentam dois MILPs para a resolução deste. O primeiro deles é uma adaptação do modelo apresentado por Edis e Oguz (2012), que considera que a utilização de recursos pode tornar menores os tempos de processamento das tarefas. O segundo, e mais importante modelo apresentado, é baseado no modelo de *strip-packing* 2D, onde fixa-se uma das dimensões da tira (altura, que analogamente significa a quantidade de recursos disponíveis) e busca-se a minimização da outra dimensão (largura, que corresponde ao *makespan*), tendo como variáveis, tarefas com dimensões que variam

de uma máquina para outra (tanto em uso de recurso renovável quanto em tempo de processamento). Entretanto, este modelo não se mostrou capaz de resolver problemas de instâncias consideradas de tamanho médio. Desta forma, os autores propuseram três Matheurísticas para serem aplicadas aos MILPs proposto. Como resultado, verificou-se que os MILPs obtiveram respostas para os problemas pequenos, mas que os problemas médios somente foram resolvidos utilizando as Matheurísticas. Além disso, foi possível verificar que a aplicação das Matheurísticas ao primeiro modelo obtiveram melhores resultados que as aplicadas no segundo modelo. Testes iniciais com instância grandes demostraram que estudos precisam ser desenvolvidos para que se obtenham boas soluções para instâncias maiores.

Afzalirad e Shafipour (2018) apresentam o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis e de elegibilidade de máquina, ou seja, se a máquina é elegível para executar dada tarefa. Os autores desenvolvem um MILP, um GA com esquema de codificação de três níveis e um algoritmo genético híbrido (HGA - *Hybrid Genetic Algorithm*), que utiliza uma heurística para executar a busca em um esquema de codificação de dois níveis. Os resultados demonstraram que o algoritmo genético é uma abordagem apropriada e eficaz para o problema abordado.

Akbar e Irohara (2018) apresentam um problema de máquinas paralelas idênticas com restrição de recursos renováveis. Para este problema, os operadores são considerados os recursos, os quais possuem quatro modos de operação, três considerados como recursos *input-output* e um como recurso adicional: i) carregamento; ii) *Setup*; iii) descarregamento; iv) controle de máquina. Também é considerado o tempo de locomoção do operador entre máquinas. Os autores apresentam três abordagens de solução um MILP, um algoritmo genético baseado em permutação (PGA - *Permuted-Based Genetic Algorithm*) e uma busca randômica (RS - *Random Search* que utiliza os mesmos esquemas de codificação e de decodificação do PGA. Os resultados mostram que o PGA pode resolver o problema em um tempo razoavelmente menor que o MILP e com uma qualidade de solução melhor que o RS.

Arbaoui e Yalaoui (2018) também abordam o problema de sequenciamento em máquinas paralelas não relacionadas com restrição de recursos renováveis, propondo um modelo em *constraint programming*. Os resultados do modelo proposto foram comparados com as abordagens propostas por Fanjul-Peyro, Perea e Ruiz (2017), comprovando que o modelo tem melhor performance para instâncias de porte pequeno.

Fleszar e Hindi (2018) apresentam uma nova abordagem de solução para o problema apresentado por Fanjul-Peyro, Perea e Ruiz (2015) e Fanjul-Peyro, Perea e Ruiz (2017), que é o problema de sequenciamento de máquinas paralelas não relacionadas com restrição de recursos renováveis. Para o problema com apenas duas máquinas, os autores desenvolveram um MILP. Para problemas com mais de duas

máquinas, propuseram uma abordagem heurística de três fases, utilizando um bom *Lower Bound*. Para o cálculo do *Lower Bound*, é utilizado um MILP do problema clássico de máquinas paralelas não relacionadas. Ainda, a segunda fase da heurística, utiliza de um modelo em *constraint programming*, chamado de CP *assignment*, para obter uma solução do problema de sequenciamento de máquinas paralelas não relacionadas com restrição de recursos renováveis para a designação obtida no modelo clássico. Para a terceira fase da abordagem é utilizado um CP do problema completo, chamado de CP *Full*, utilizando a solução obtida na fase anterior como solução inicial. Esta abordagem obteve ótimos resultados em relação às instâncias propostas anteriormente (FANJUL-PEYRO; PEREA; RUIZ, 2017). Além disso, os autores propuseram novas instâncias para o problema. Para a variante de duas máquinas, o MILP soluciona todas as instâncias na otimalidade. Nos casos de mais de duas máquinas, a abordagem apresentada encontra soluções melhores ou iguais às que propostas anteriormente apresentadas em tempos computacionais consideravelmente menores.

Fu et al. (2018) propõem uma abordagem diferente para o problema de máquinas paralelas idênticas com utilização de recursos renováveis. Nesta abordagem as tarefas possuem diferentes modos de processamento, os quais podem acelerar o tempo de processamento, e para cada modo de processamento existe uma quantidade necessária de recursos renováveis. Para solucionar este problema os autores propões um algoritmo genético primário-secundário (PSGA - *Primary-Secondary Genetic Algorithm*), onde o cromossomo primário representa a decisão de alocação de recursos e o cromossomo secundário representa as decisões de designação e sequenciamento das tarefas. Os resultados mostraram que o algoritmo proposto é eficiente para a abordagem deste trabalho.

Villa, Vallada e Fanjul-Peyro (2018) continuam com o mesmo problema apresentado por Fanjul-Peyro, Perea e Ruiz (2017) e se concentram em apresentar heurísticas mais eficientes e efetivas do que as apresentadas anteriormente, por meio de duas abordagens. Uma abordagem baseada nos recursos que consiste de três fases: i) ordenação das tarefas (desconsiderando a utilização dos recursos); ii) construção da solução (conferindo a factibilidade); e iii) aplicação de uma busca local para o aprimoramento da solução. Uma segunda abordagem que é baseada na designação tarefa-máquina, consiste de duas fases: i) tem-se a construção de oito soluções iniciais distintas, de acordo com as oito regras de designação, estas soluções não consideram a verificação de factibilidade dos recursos, portanto é possível obter soluções infactíveis. Em seguida, é realizado a verificação da factibilidade e se necessário é aplicado um mecanismo de reparo para correção da solução (caso infactível); e ii) após a fase de construção é aplicado um *Local Search* multi-fases em todas as soluções encontradas. A estrutura do *Local Search* é definida como: i) Inserção das tarefas da máquina do *makespan* para as outras máquinas; ii) Inserção das tarefas em todas as máquinas; iii)

Swap externo da máquina do *makespan* para as outras máquinas; iv) *Swap* externo de todas as tarefas para todas as máquinas. Após a aplicação do *Local Search* é selecionada a melhor solução dentre as oito encontradas.

Zheng e Wang (2018) apresentam um novo problema baseado em máquinas paralelas não relacionadas com recursos renováveis, denominado *resource constraint unreleated parallel machine green scheduling problem*, este problema é abordado com dois objetivos, minimizar o *makespan* e minimizar o total de emissões de carbono. Os autores utilizaram a abordagem de meta-heurística populacional *Fruit fly optimization algorithm* (FFOA), apresentada anteriormente por Zheng e Wang (2016).

Vallada, Villa e Fanjul-Peyro (2019), dando continuidade aos trabalhos de Villa, Vallada e Fanjul-Peyro (2018) e Fanjul-Peyro, Perea e Ruiz (2017), apresentam uma busca de dispersão enriquecida (ESS - *Enriched Scatter Search*) e um *Enriched Iterated Greedy*(EIG). Ambos métodos utilizam como solução inicial as soluções do algoritmo de Villa, Vallada e Fanjul-Peyro (2018), assim permitindo soluções não factíveis e se faz necessário a aplicação de um mecanismo reparador de soluções. Os algoritmos propostos utilizam diferentes métodos de LS, baseados em inserção, *swap* e busca em vizinhanças restritas. Os resultados obtidos superaram os melhores resultados para este problema.

Yepes-Borrero et al. (2020) propõem o problema de máquinas paralelas não relacionadas com *setup* dependente e recursos renováveis no período de *setup* (*input-output*), com objetivo de minimizar o *makespan*. Em sua nova abordagem, os autores, diferente das pesquisas anteriores, consideram a utilização do recurso, durante o tempo de *setup* e não durante o processamento das tarefas, como nas pesquisas anteriores. Para solucionar este problema os autores apresentam um MILP e um GRASP. Para a fase construtiva, do GRASP proposto, são avaliados três algoritmos distintos. Como a fase de construção pode fornecer soluções não factíveis, é desenvolvido um mecanismo de reparação. Para a fase da busca local, são realizadas buscas baseadas em *swap* interno, *swap* externo e inserção. Dado a falta de instâncias para o novo problema apresentado, os autores adicionam a informação de *setup* nas instâncias existentes apresentadas por Fanjul-Peyro, Perea e Ruiz (2017). Os resultados mostram que o GRASP apresentado supera o MILP desenvolvido para este problema. Vale notar que existe uma diferença entre o problema abordado no trabalho de Yepes-Borrero et al. (2020) e o problema apresentado nesta pesquisa. Ambos abordam o problema de máquinas paralelas não relacionadas com *setup* dependente e recursos renováveis, no entanto, Yepes-Borrero et al. (2020) consideram que os recursos são necessário para executar o *setup* das tarefas e esta pesquisa considera que os recursos são necessários para o processamento das tarefas.

4 MÉTODOS DE RESOLUÇÃO

Este capítulo apresenta os cinco métodos de resolução, desenvolvidos nesta pesquisa, para os problemas de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR) e de máquinas paralelas não relacionadas com *setup* dependente e com restrição de recursos renováveis (UPMSR). Como um dos objetivos deste trabalho é introduzir como um novo problema o UPMSR, que é uma nova variante desta classe de problemas de máquinas paralelas com restrição de recursos, são exclusivamente desenvolvidos dois novos modelos matemáticos, um modelo de programação linear inteira mista (MILP) e um modelo de programação por restrição (CP). Vale ressaltar que, para o UPMR não existe a necessidade de desenvolver estas duas abordagens, pois já foram propostas anteriormente por trabalhos presentes na literatura. Além disso, para ambos os problemas, dois outros métodos são propostos, o Greedy Randomized Adaptive Evolutionary Path Relinking (GRAEPR) e uma abordagem híbrida, a qual utiliza o GRAEPR desenvolvido em conjunto com um modelo de programação por restrição (GRAEPR + CP). Dada a eficiência obtida na abordagem proposta por Fleszar e Hindi (2018) (AFH) para o problema UPMR, apresentada na Seção 2.3, esta abordagem é devidamente adaptada para o UPMSR. A TABELA 5 identifica os métodos de resolução desenvolvidos para ambos os problemas.

TABELA 5 – PROBLEMAS E OS RESPECTIVOS MÉTODOS DE RESOLUÇÃO

Método de Resolução	**Problemas**	
	UPMR	**UPMSR**
MILP	-	✓
CP	-	✓
GRAEPR	✓	✓
GRAEPR + CP	✓	✓
AFH	-	✓

FONTE: O autor (2020).

4.1 MODELO DE PROGRAMAÇÃO LINEAR INTEIRA MISTA

O primeiro método de resolução apresentado é um modelo de programação linear inteira mista (MILP), desenvolvido apenas para o problema de máquinas paralelas não relacionadas com *setup* dependente e com restrição de recursos renováveis. Conforme já mencionado, as pesquisas anteriores apresentaram modelos de programação linar inteira mista para o problema UPMR. O desenvolvimento de um modelo MILP

se faz devido a possibilidade em obter as soluções em sua otimalidade, pois é um método exato. As notações utilizadas na modelagem matemática, para o problema d são apresentadas na TABELA 6.

TABELA 6 – NOTAÇÕES UTILIZADA NO MODELO DE PROGRAMAÇÃO LINEAR INTEIRA MISTA PARA O UPMSR

Conjuntos e índices	
J	Tarefas; $h, i, j \in J = 0...\|J\|$; $i = 0$ é uma tarefa fictícia;
M	Máquinas; $m \in M = 1...\|M\|$;
T	Horizonte de Tempo; $t \in T = 1...\|T\|$;
Parâmetros	
P_{jm}	Tempo de processamento da tarefa j na máquina m;
S_{ijm}	Tempo de *Setup* entre as tarefas i e j (sendo a tarefa i imediatamente antecessora da tarefa j) na máquina m;
Res_{jm}	Recursos necessários para realizar a tarefa j na máquina m;
Res_{max}	Limite de recurso disponível para a realização das tarefas;
L	Número suficientemente grande;
Variáveis de Decisão	
C_{max}	Instante máximo de conclusão das tarefas ou *makespan*;
C_{jm}	tempo de conclusão da tarefa j na máquina m;
X_{ijm}	= 1, se a tarefa i é imediatamente antecessora da tarefa i na máquina m; = 0, caso contrário;
Y_{jmt}	= 1, se a tarefa j é concluída na máquina m no instante t; = 0, caso contrário;

FONTE: O autor (2020).

Min C_{max} (4.1)

s.a.

$$\sum_{i=0;i\neq j}^{|J|} \sum_{m=1}^{|M|} X_{ijm} = 1 \qquad \forall j = 1..|J| \tag{4.2}$$

$$\sum_{j=1;j\neq i}^{|J|} \sum_{m=1}^{|M|} X_{ijm} \leq 1 \qquad \forall i = 1..|J| \tag{4.3}$$

$$\sum_{i=0;i\neq h}^{|J|} X_{ihm} - \sum_{j=0;j\neq h}^{|J|} X_{hjm} = 0 \qquad \forall m = 1..|M|; \forall h = 1..|J|; \tag{4.4}$$

$$\sum_{j=1}^{|J|} X_{0jm} \leq 1 \qquad \forall m = 1..|M|; \tag{4.5}$$

$$C_{0m} = 0 \qquad \forall m = 1..|M|; \tag{4.6}$$

$$C_{max} \geq C_{jm} \qquad \forall j = 0..|J|; \forall m = 1..|M|; \tag{4.7}$$

$$C_{im} \leq L \sum_{i=0; i \neq j}^{|J|} X_{ijm} \qquad \forall i = 0..|J|; \forall m = 1..|M|; \tag{4.8}$$

$$C_{jm} - C_{im} \geq S_{ijm} + P_{jm} + (X_{ijm} - 1)L \quad \forall i = 0..|J|; \forall j = 1..|J|; i \neq j; \forall m = 1..|M|; \tag{4.9}$$

$$\sum_{m=1}^{|M|} \sum_{t=1}^{|T|} Y_{jmt} = 1 \qquad \forall j = 1..|J|; \tag{4.10}$$

$$\sum_{t=1}^{|T|} t Y_{jmt} = C_{im} \qquad \forall j = 0..|J|; \forall m = 0..|M|; \tag{4.11}$$

$$\sum_{j=0}^{|J|} \sum_{m=0}^{|M|} \sum_{s=t}^{t+P_{jm}} Res_{jm} Y_{jms} \leq Res_{max} \qquad \forall t = 0..|T|; \tag{4.12}$$

$$\frac{1}{R_{max}} \sum_{m=1}^{|M|} \sum_{i=1}^{|J|} \sum_{j=1; i \neq j}^{|J|} R_{im} P_{im} X_{ijm} \leq C_{max} \tag{4.13}$$

$$C_{max} \in \mathbb{R}_+^n \tag{4.14}$$

$$C_{im} \in \mathbb{R}_+^n \qquad \forall i = 0..|J|; \forall m = 1..|M|; \tag{4.15}$$

$$X_{ijm} \in \mathbb{B} \qquad \forall i = 0..|J|; \forall j = 0..|J|; i \neq j; \forall m = 1..|M|; \tag{4.16}$$

$$Y_{imt} \in \mathbb{B} \qquad \forall i = 0..|J|; \forall m = 1..|M|; \forall t = 1..|T|; \tag{4.17}$$

A função objetivo (4.1) é responsável pela minimização do *makespan*. Os conjunto de restrições (4.2) e (4.3) garantem que as tarefas sejam executadas em apenas uma máquina. Além disso, juntamente com as restrições (4.4), garantem a conservação de fluxo, ou seja, garantem que todas as tarefas possuam apenas uma tarefa imediatamente antecessora e apenas uma tarefa imediatamente sucessora,

respectivamente. As restrições (4.5) garantem que a tarefa artificial esteja posicionada no início de todas as máquinas, sendo que as máquinas podem ou não serem utilizadas. O conjunto de restrições (4.6) estabelece que o instante de conclusão da atividade artificial é nulo em todas as máquinas. O conjunto de restrições (4.7) garante que o *makespan* deve ser maior ou igual ao instante de conclusão de todas as tarefas, em todas as máquinas. Em (4.8) é garantido que o tempo de conclusão de uma tarefa em uma máquina é maior que zero apenas se ela for realizada na mesma. As restrições (4.9) asseguram que o instante de término da tarefa j deve ser maior ou igual a soma entre o instante de término da atividade i (imediatamente antecessora), o tempo de *setup* entre as duas tarefas e o tempo de de processamento da tarefa j.

As restrições (4.10) garantem que cada tarefa deve ser concluída em apenas uma máquina e em apenas um instante de tempo. O conjunto de restrições (4.11) atribui valor à variável Y_{imt}, definindo assim, o instante de conclusão da tarefa de acordo com o valor atribuído à variável C_{im}. O conjunto de restrições (4.12) garante que o limite de recursos total não seja ultrapassado em cada instante do horizonte de tempo. A restrição (4.13) auxilia na convergência da solução, garantindo que o *makespan* respeite o *Lower Bound* proposto por Grigoriev, Sviridenko e Uetz (2005) (2.7). Por fim, os conjuntos de restrições (4.14) - (4.17) representam o domínio das variáveis de decisão.

Vale ressaltar que é necessário realizar o cálculo do *Upper Bound* (UB) da solução para determinar o tamanho do conjunto T, conjunto do horizonte de tempo, responsável por determinar a quantidade de variáveis Y_{imt} a serem criadas, pois quanto menor o horizonte, menor a quantidade de variáveis utilizadas, logo, maior a facilidade de convergência do modelo. Para isto, é utilizado o seguinte cálculo:

$$UB = 1.25 \cdot max\{P_{im}\} \cdot \frac{|J|}{|M|} \tag{4.18}$$

O valor é obtido através da multiplicação de 1.25 pelo maior tempo de processamento, entre todas as tarefas e máquinas, multiplicado também pela fração da quantidade de tarefas pela quantidade de máquinas. O valor definido de 1.25 é justificado pelos valores definidos na geração dos tempos de *setup* das instâncias do problema de máquinas paralelas não relacionadas com *setup* dependente e com restrição de recursos renováveis.

4.2 MODELO DE PROGRAMAÇÃO POR RESTRIÇÃO

Modelos de programação por restrição (CP) estão sendo utilizados para resolução de problemas de máquinas paralelas com restrições de recursos. Edis e Oguz (2011) apresentaram um modelo CP combinado com Relaxação Lagrangiana. Edis

e Ozkarahan (2011, 2012), Edis e Oguz (2012) e Fleszar e Hindi (2018) utilizaram abordagens híbridas, em que os modelos de *constraint programming* foram utilizados na fase final de suas respectivas abordagens. Além destes, Arbaoui e Yalaoui (2018) também apresentaram uma versão do modelo de programação de restrição para o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR). Utilizando destes estudos prévios como grandes motivadores, foi desenvolvido um modelo CP para o problema de máquinas paralelas não relacionadas com *setup* dependente e restrição de recursos renováveis (UPMSR). O modelo foi desenvolvido baseado no modelo proposto por Fleszar e Hindi (2018). As notações utilizadas para a modelagem em *constraint programming* são apresentadas na TABELA 7.

TABELA 7 – NOTAÇÕES UTILIZADAS NO MODELO DE *CONSTRAINT PROGRAMMING* PARA O UPMSR

Conjuntos e índices	
J	Tarefas; $i, j \in J = 1...\|J\|$;
M	Máquinas; $m \in M = 1...\|M\|$;
Parâmetros	
P_{jm}	Tempo de processamento da tarefa j na máquina m;
S_{ijm}	Tempo de *Setup* entre as tarefas i e j (sendo a tarefa i imediatamente antecessora da tarefa j) na máquina m;
Res_{jm}	Recursos necessários para realizar a tarefa j na máquina m;
Res_{max}	Limite de recurso disponível para a realização das tarefas;
LB	*Lower Bound*;
Variáveis de Decisão	
C_{max}	Instante máximo de conclusão das tarefas ou *makespan*;
X_{jm}	Variável de intervalo opcional; = Presente, se a tarefa j é executada na máquina m; = Ausente, caso contrário;
Y_j	Variável de intervalo relacionada à tarefa j;
W_m	Variável de sequência de intervalos relacionada ao sequenciamento que será executado na máquina m;
Z_m	Variável de transição que representa os tempos de *setup* executados no sequenciamento da máquina m;
$Resource$	Função Acumulativa;

FONTE: O autor (2020).

Min C_{max} (4.19)

s.a.

$$X_{jm} = optional\_size(P_{jm}) \qquad \forall j = 1..|J|; \forall m = 1..|M|; \tag{4.20}$$

$$noOverlap(\forall j X_{jm}) \qquad \forall m = 1..|M|; \tag{4.21}$$

$$Y_j = alternative(\forall m X_{jm}) \qquad \forall j = 1..|J|; \tag{4.22}$$

$$Z_m = S_{ijm} \qquad \forall m = 1..|M|; \tag{4.23}$$

$$noOverlap(W_m, Z_m) \qquad \forall m = 1..|M|; \tag{4.24}$$

$$Resource = \sum_{j=1}^{|J|} \sum_{m=1}^{|M|} pulse(X_{jm}, R_{jm}) \tag{4.25}$$

$$Resource \leq R_{max} \tag{4.26}$$

$$C_{max} \geq endOF(Y_j) \qquad \forall j = 1..|J|; \tag{4.27}$$

$$C_{max} \geq \frac{\sum_{m}^{|M|} \sum_{j}^{|J|} Res_{jm} P_{jm} X_{jm}}{R_{max}} \tag{4.28}$$

A equação (4.19) define o objetivo de minimizar o *makespan*. O tempo de processamento, da tarefa j na máquina m, é atribuído à variável de intervalo opcional X_{jm} (4.20). A não sobreposição das tarefas, caso sejam designadas à máquina m, é garantida no conjunto de restrições (4.21). Uma única variável de intervalo opcional é atribuída à variável Y_j nas restrições (4.22). O conjunto de restrição (4.23) atribui os valores da matriz de *setup*, relacionada à máquina m, à variável de transição Z_m. A não sobreposição dos *setups* é garantida pelas restrições (4.24). A função acumulativa $Resource$ é definida na restrição (4.25). O limite dos recursos é assegurado na restrição (4.26). O *makespan* é calculado pelo conjunto de restrições (4.27). O valor do *makespan* é limitado pelo valor de *Lower Bound*, proposto por Grigoriev, Sviridenko e Uetz (2005), na restrição (4.28).

4.3 GREEDY RANDOMIZED ADAPTIVE EVOLUTIONARY PATH-RELINKING

O *greedy randomized adaptive search procedure* (GRASP), introduzido por Feo e Resende (1995), é uma meta-heurística iterativa probabilística para resolução de diversos problemas de otimização combinatória. Ao longo dos anos, o GRASP é considerado efetivo e de fácil implementação, sendo capaz de encontrar soluções ótimas ou aproximadas (RESENDE; RIBEIRO, 2016), que consiste em duas fases: construtiva e busca local (RESENDE; RIBEIRO, 2005b).

Pesquisas relacionadas a aplicação do GRASP em problemas de sequenciamento de máquinas foram incentivadoras para abordar este método de resolução. Feo, Venkatraman e Bard (1991) e Feo, Sarathy e McGahan (1996) aplicaram o GRASP para solucionar problemas de sequenciamento em única máquina, enquanto Resende, Binato et al. (2002) propuseram uma aplicação para problemas de sequenciamento *Job Shop*. Em um estudo mais recente, Yepes-Borrero et al. (2020) utilizam o GRASP para o problema o problema de máquinas paralelas não relacionadas com *setup* dependente e recursos renováveis no período de *setup*. Vale ressaltar que existe uma pequena diferença no problema abordado por Yepes-Borrero et al. (2020) e o problema abordado nesta pesquisa. Enquanto o primeiro considera a utilização dos recursos no período de

setup, este trabalho aborda a a utilização de recursos no período de processamento das tarefas. Para mais informações sobre o GRASP, aqui referenciamos Festa e Resende (2002) e Resende e Ribeiro (2016).

Algoritmo 1 Pseudocódigo do GRASP

1: **Input:** α_{GRASP}
2: $x \leftarrow$ Construir_Solução(α_{GRASP});
3: $x \leftarrow$ Bulsca_Local(x);
4: **return** x;

O Algoritmo 1 apresenta um pseudocódigo generalizado para o GRASP, onde explicita o parâmetro de entrada utilizado, o índice de gulosidade, α_{GRASP} (Linha 1), e suas duas fases: construtiva (Linha 2) e busca local (Linha 3).

A primeira fase é responsável pela construção da solução. Nesta fase existe um conjunto de todos os elementos que podem ser incorporados à solução. São avaliados os custos de inserção destes elementos na função objetivo e após esta avaliação, é gerada uma lista restrita de candidatos (RCL - *Restricted Candidate List*) com os melhores candidatos. O tamanho da RCL é determinada de acordo com o índice de gulosidade $\alpha_{GRASP} \in [0, 1]$, onde $\alpha_{GRASP} = 0$, significa puramente guloso e $\alpha_{GRASP} = 1$, totalmente aleatório. Então, o elemento a ser inserido na solução parcial é selecionado a partir desta RCL. Após a seleção e inserção deste elemento, a RCL e os custos de inserção são atualizados. Estes passos são repetidos até que a solução esteja completa. É possível que a fase construtiva não forneça soluções factíveis, logo, pode ser necessária a aplicação de mecanismos de reparo de solução. A fase construtiva do GRASP, para problemas de máquinas paralelas, é apresentada no Algoritmo 2.

Algoritmo 2 Pseudocódigo da fase construtiva do GRASP

1: **Input:** α_{GRASP}
2: $RCL \leftarrow$ Inicializar_RCL(α_{GRASP});
3: **while** ($RCL \neq \emptyset$) **do**
4: $\quad (j, m) \leftarrow$ Seleção_Aleatória(RCL);
5: $\quad x \leftarrow$ adiciona a tarefa j na máquina m;
6: $\quad RCL \leftarrow$ Atualizar_RCL(α_{GRASP});
7: **end while**
8: $x \leftarrow$ Reparar_Solução(x);
9: **return** x;

O algoritmo tem como parâmetro de entrada o índice de gulosidade, α_{GRASP} (Linha 1). Na linha 2, a lista restrita de candidatos é inicializada e seu tamanho é determinado pelo índice de gulosidade α_{GRASP}. Enquanto existirem candidatos na RCL, os procedimentos de construção da solução serão realizados (Linha 3). Na linha 4 é selecionada uma tarefa j e a máquina m em que esta será designada. A solução é atualizada com a inserção da tarefa j na última posição da máquina m (Linha 5).

Caso o problema considere os tempos de *setup*, a inserção deve ocorrer utilizando esta informação. Então, a RCL é atualizada, removendo a tarefa j da lista (Linha 6). A construção da solução não garante factibilidade para o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis, portanto, após a fase de construção, onde as tarefas são designadas e sequenciadas, é fundamental avaliar a solução e verificar se as restrições de recurso são respeitadas. Ocorrendo o excesso de recursos em algum instante, se faz necessária a reparação da solução (Linha 8). Yepes-Borrero et al. (2020) apresentam um método de reparação, que pode ser adaptado e utilizado para os problemas abordados neste trabalho. Pata obter melhor performance computacional do método, proposto por Yepes-Borrero et al. (2020), foram feitas algumas alterações que são apresentadas no Algoritmo 3.

Algoritmo 3 Pseudocódigo do procedimento de reparo

1: **Input:** x
2: **for each** $t < C_{max}$, tal que $\exists\, t = C_{jm_j}$, $\forall j \in J$ **do**
3: $\quad\lambda_t \leftarrow \sum_j R_{jm_j}$, tal que $C_{jm_j} - P_{jm_j} < t \leq C_{jm_j}\ \forall j \in J$;
4: $\quad$**if** $(\lambda > R_{max})$ **then**
5: $\quad\quad j^* \leftarrow argmax_j\{C_{jm_j} - P_{jm_j}\}$, tal que $C_{jm_j} - P_{jm_j} < t \leq C_{jm_j}\ \forall j \in J$;
6: $\quad\quad x \leftarrow$ Adiar_Start(j^*);
7: $\quad\quad t \leftarrow t - 1$;
8: $\quad$**end if**
9: **end for**
10: **return** x;

O parâmetro de entrada deste algoritmo é a solução x (Linha 1). Diferente do método apresentado por Yepes-Borrero et al. (2020), o procedimento de reparo desta pesquisa consiste em avaliar apenas os instantes do horizonte de tempo em que as tarefas foram concluídas (Loop 2 - 9). Vale ressaltar que em toda solução, é assumido que toda tarefa j é designada para uma máquina $m_j \in M$. A Linha 3 calcula o coeficiente λ_t, somando apenas os recursos utilizados pelas tarefas, em suas respectivas máquinas, que estão sendo processadas no instante t. Então é verificado se o valor do coeficiente não ultrapassa o limite de recursos R_{max} (Linha 4). Caso o limite seja ultrapassado, é selecionada, entre as tarefas que se sobrepõem neste instante de tempo, a tarefa que se inicia por último (Linha 5). Então, a tarefa selecionada tem seu início adiado para o próximo instante de tempo em que a restrição é respeitada (Linha 6). Por fim, o instante de tempo é decrementado para que possa ser reavaliado, garantindo assim, que o limite de recursos seja respeitado no instante t após a atualização da solução.

A segunda fase do GRASP tem por objetivo aprimorar a solução gerada pela fase anterior, avaliando a vizinhança de soluções até que a solução encontrada seja um ótimo local. Para otimizar esta fase, Mladenović e Hansen (1997) apresentaram um *Variable Neighborhood Descent* (VND), uma variante do *Variable Neighborhood Search*

(VNS) proposto por Hernández-Pérez, Rodríguez-Martín e Salazar-González (2009). O VND é um algoritmo interativo com objetivo de realizar uma busca local, com alteração determinística de vizinhança. Ou seja, performa uma busca local em uma vizinhança, até encontrar o mínimo local, e após isto utiliza a solução encontrada como ponto de partida para performar a busca local da próxima vizinhança. Para exemplificar o VND, considere um conjunto de vizinhanças $N_k(x)$, o qual contém k vizinhanças distintas e determinadas previamente e uma solução x. Enquanto o critério de parada não é satisfeito, a cada iteração do método, após a aplicação da busca local na solução x, utilizando uma vizinhança do conjunto $N_k(x)$, é verificada se a solução é aprimorada. Caso positivo, a solução atual x é atualizada. Em seguida, é realizada a aplicação da busca local utilizando a próxima vizinhança, até que todas as vizinhanças sejam utilizadas. Para este trabalho, foi desenvolvido um VND para a fase da busca local do GRASP.

Inspirado pelo trabalho de Yepes-Borrero et al. (2020), o conjunto de vizinhanças é composto por três vizinhanças, $N_k(x) \in \{insertion, 2opt, swap\}$: *External Insertion* ($insertion$), *External Insertion 2-OPT* ($2opt$) e *External Swap* ($swap$). Yepes-Borrero et al. (2020) utilizam um *Internal Swap* em sua busca local, no entanto foi observado que, para os problemas abordados nesta pesquisa, a utilização de um *External Insertion 2-OPT* é mais eficiente. A fase de busca local é apresentada no Algoritmo 4.

Algoritmo 4 Pseudocódigo da fase de Busca Local do GRASP

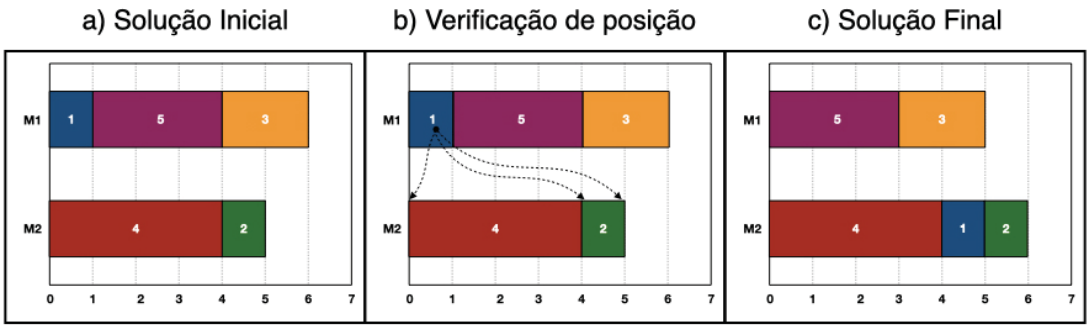
```
1:  Input: x
2:  x' ← x;
3:  x* ← x;
4:  while (Condição de Parada) do
5:      for each N_k(x) ∈ {insertion, 2opt, swap} do
6:          x' ← Remover_Ocioso(x);
7:          x' ← Aplicar_Vizinhança(x, N_k(x));
8:          x' ← Reparar_Solução(x);
9:          if (C(x') < C(x*)) then
10:             x' ← x*;
11:         end if
12:     end for
13: end while
14: return x*;
```

O parâmetro de entrada do VND é a solução x (Linha 1). As linhas 2 e 3 atribuem esta solução, obtida na fase inicial, à solução atual x' e à melhor solução x^*, respectivamente. Enquanto o critério de parada não for atingido, ou seja, enquanto houver aprimoramento da solução (Linha 4), para cada vizinhança determinada no conjunto $N_k(x)$, é primeiramente aplicado um procedimento que remove os tempos ociosos contidos na solução (Linha 6). Esta operação é justificada por dois fatores,

primeiro pelo critério de otimização do problema, que é a minimização do *makespan*, isto significa, que a minimização dos tempos ociosos influencia na minimização do *makespan*. Em contrapartida, isto afeta diretamente a factibilidade da solução. O segundo fator são os coeficientes β (4.29) e γ (4.31) que avaliam a utilização e o excesso dos recursos na solução, respectivamente. Então, na Linha 7 a busca local é aplicada para a vizinhança determinada, até que o local ótimo seja encontrado. Devido a utilização do procedimento de remoção dos tempos ociosos e a possibilidade de gerar uma solução infactível, se torna necessário utilizar o procedimento de reparo antes da avaliação das soluções (Linha 8). Por fim, o valor da solução atual $C(x')$ é avaliada e comparada com o valor da melhor solução $C(x^*)$ (Linha 9). Em caso de aprimoramento, a solução é atualizada (Linhas 10), mantendo sempre a melhor solução.

A primeira vizinhança, a ser aplicada a busca local, é o *External Insertion*, $N_k(x) = insertion$. O *External Insertion* realiza a verificação da inserção de cada tarefa j, em todas as posições p das outras máquinas m. A FIGURA 11 apresenta um exemplo de uma solução após a aplicação do *External Insertion*. Neste exemplo é considerado que a máquina em que a tarefa será inserida já está definida.

FIGURA 11 – EXEMPLO *EXTERNAL INSERTION*.



FONTE: O autor (2020).

Para melhor performance deste método, são utilizados dois coeficientes de avaliação β e γ. Para selecionar a tarefa j e a máquina m_j^*, onde será realizada a inserção é utilizado o coeficiente β, pois a troca de designação da tarefa j, impacta na quantidade total de recursos no sequenciamento. Este coeficiente é definido pela razão do total de recursos no sequenciamento, calculado pela multiplicação entre os recursos necessários (R_{jm_j}) e o tempo de processamento (P_{jm_j}) da tarefa j na máquina m_j designada, pelo limite de recursos.

$$\beta = \frac{\sum_j^{|J|} R_{jm_j} P_{jm_j}}{R_{max}} \tag{4.29}$$

Grigoriev, Sviridenko e Uetz (2005) atestaram que é possível utilizar o coeficiente β como *Lower Bound* para a solução deste problema, assim, justificando a

utilização deste como coeficiente para avaliação. Para definir a posição p em que será inserida a tarefa j, é utilizado o coeficiente γ, que considera a quantidade total de recursos em excesso e a soma de instante de tempos em que o limite é excedido. Para cada instante de tempo t, o coeficiente θ_t é determinado se a quantidade de recursos utilizados, pelas tarefas j processadas neste instante t em suas respectivas máquinas m_j, ultrapassa ou não o limite. O cálculo de θ_t é apresentado na Equação 4.30:

$$\theta_t = \begin{cases} 1 & , \sum_j^{|J|} R_{jm_j} > R_{max}, \text{ tal que } C_{jm_j} - P_{jm_j} < t \leq C_{jm_j} \forall j \in J \\ 0 & , c.c. \end{cases} \tag{4.30}$$

O coeficiente γ é determinado pelo somatório do coeficiente θ_t, que representa a quantidade de instantes de tempo em que a quantidade de recursos necessários ultrapassam o limite R_{max}. Se, em algum instante isto acontecer, o valor de γ é calculado. Caso contrário, é atribuído o valor 0, que significa que a solução é factível sem ociosidade no sequenciamento das tarefas. O coeficiente γ é definido por:

$$\gamma = \begin{cases} \frac{\sum_t max\{\lambda_t - R_{max}, 0\}}{\sum_t \theta_t} & , \sum_t \theta_t > 0 \\ 0 & , c.c. \end{cases} \tag{4.31}$$

Após avaliar todas as trocas possíveis, a troca que apresenta o menor valor de γ é realizada, desde que seja menor que o valor do coeficiente da solução atual γ^* e que apresente melhoria no *makespan*. Vale ressaltar que o *makespan* utilizado para a avaliação é o da solução após a remoção dos tempos ociosos. O procedimento é detalhado no Algoritmo 5:

Algoritmo 5 Pseudocódigo do External Insertion

```
1:  Input: x
2:  while (Condição de Parada) do
3:      C*_max ← Calcular_Makespan(x);
4:      β* ← Calcular_Beta(x);
5:      γ* ← Calcular_Gamma(x);
6:      Best_Insertion(j*, m*, p*) ← Ø;
7:      for each j ∈ J do
8:          for each m ∈ M and m_j ≠ m do
9:              β_aux ← Calcular_Beta(j, m);
10:             if (β_aux < β*) then
11:                 β* ← β_aux;
12:                 Best_Insertion(j*, m*, p*) ← (j, m);
13:             end if
14:         end for
15:     end for
```

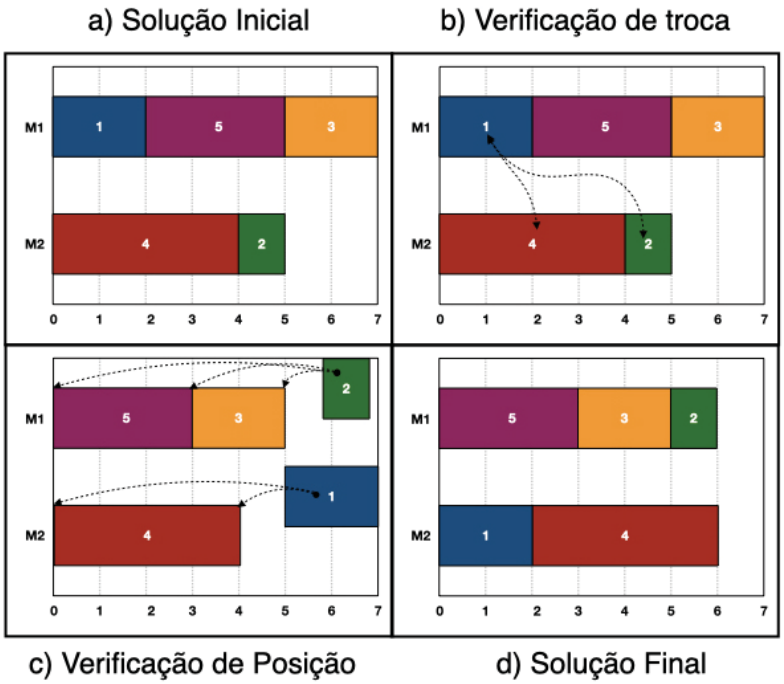
```
16:     for each p ∈ m* do
17:         C_aux ← Calcular_Makespan(j*, m*, p);
18:         γ_aux ← Calcular_Gamma(j*, m*, p);
19:         if (γ_aux < γ*) and (C_aux ≤ C*_max) then
20:             γ* ← γ_aux;
21:             C*_max ← C_aux;
22:             Best_Insertion(j*, m*, p*) ← (p);
23:         end if
24:     end for
25:     x ← Best_Insertion(j*, j*, m*);
26: end while
27: return x;
```

O parâmetro de entrada do *External Insertion* é a solução x (Linha 1). Inicialmente, o *External Insertion* calcula o *makespan* C^*_{max} e os coeficientes β^* e γ^* da solução atual (Linhas 3 - 5). Em seguida, na linha 6, é inicializada a variável que armazena a melhor inserção. É calculado o coeficiente auxiliar β_{aux}, com o valor do coeficiente β de cada tarefa j sendo inserida em cada máquina m possível, na linha 9. Em caso de melhoria da solução, o β^* e a melhor inserção são atualizados (Linhas 11-12). Após a definição da tarefa j^* e a máquina m^* em que será inserida, são realizados os cálculos do *makespan* e do coeficiente γ_{aux} auxiliares, para a inserção da tarefa j^* em todas as posições p da máquina m^* (Linhas 17 - 18). Os coeficientes são avaliados e se houver aprimoramento do coeficiente e do valor de *makespan*, são atualizados os valores do coeficiente γ^*, do *makespan* e da melhor inserção (Linhas 20 - 22). Por fim, se a solução é aprimorada, na linha 25 é realizada a melhor inserção. O *External Insertion* é repetido enquanto houver melhoria da solução.

A segunda vizinhança, a ser aplicada a busca local, é o *External Insertion 2-OPT*, $N_k(x) = 2opt$. De forma similar à primeira vizinhança, o *External Insertion 2-OPT* tem por objetivo avaliar as possíveis trocas entre uma tarefa i, designada em uma máquina m_1 e uma tarefa j, designada em uma máquina m_2. Para esta avaliação é utilizado o coeficiente β, apresentado na Equação 4.29. Após a definição das tarefas e suas respectivas máquinas, o algoritmo utiliza o coeficiente γ (4.31) para definir a posição em que cada uma das tarefas será inserida. Ou seja, a tarefa i, originalmente designada na máquina m_1, será inserida na posição p_2 da máquina m_2. De forma análoga, a tarefa j, originalmente designada na máquina m_2, será inserida na posição p_1 da máquina m_1.

A FIGURA 12 apresenta um exemplo de uma solução após a aplicação do *External Insertion 2-OPT*. Neste exemplo, já estão definidas as máquinas em que as tarefas serão primeiramente trocadas e depois inseridas.

FIGURA 12 – EXEMPLO *EXTERNAL INSERTION 2-OPT*.



FONTE: O autor (2020).

O Algoritmo 6 apresenta o procedimento do *External Insertion 2-OPT*:

Algoritmo 6 Pseudocódigo do External Insertion 2-OPT

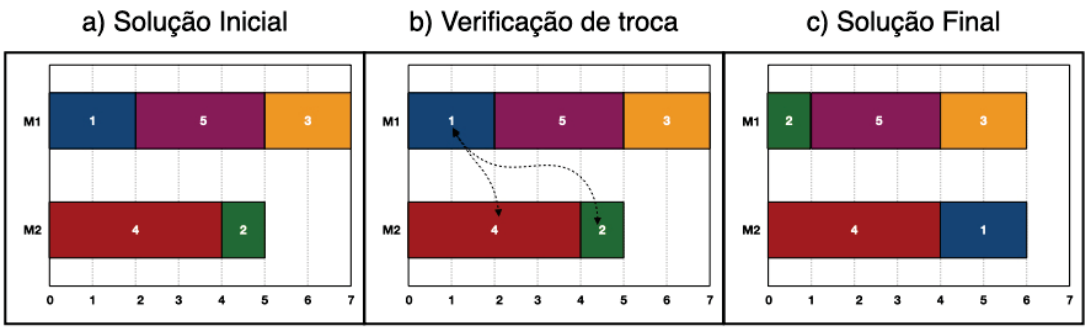
Input: x
while (Condição de Parada) **do**
  $C^*_{max} \leftarrow$ Calcular_Makespan(x);
  $\beta^* \leftarrow$ Calcular_Beta(x);
  $\gamma^* \leftarrow$ Calcular_Gamma(x);
  Best_Insertion_2opt(i^*, m^*_1, p^*_1, j^*, m^*_2, p^*_2) $\leftarrow$ Ø;
  for each $m_1 \in M$ **do**
    for each i, tal que $m_i = m_1$ **do**
      for each $m_2 \in M$, and $m_1 \neq m_2$ **do**
        for each j, tal que $m_j = m_2$ **do**
          $\beta_{aux} \leftarrow$ Calcular_Beta(i, m_1, j, m_2);
          if ($\beta_{aux} < \beta^*$) **then**
            $\beta^* \leftarrow \beta aux$;
            Best_Insertion_2opt(i^*, m^*_1, p^*_1, j^*, m^*_2, p^*_2) $\leftarrow$ (i, m_1 j, m_2);
          end if
        end for
      end for
    end for
  end for
  for each $p_1 \in m^*_1$ **do**
    for each $p_2 \in m^*_2$ **do**
      $C_{aux} \leftarrow$ Calcular_Makespan(i^*, m^*_1, p_1, j^*, m^*_2, p_2);

```
23:            γ_aux ← Calcular_Gamma(i*, m1*, p1, j*, m2*, p2);
24:            if (γ_aux < γ*) and (C_aux ≤ C*_max) then
25:                γ* ← γ_aux;
26:                Best_Insertion_2opt(i*, m1*, p1*, j*, m2*, p2*) ← (p1, p2);
27:            end if
28:        end for
29:    end for
30:    x ← Best_Insertion_2opt(i*, m1*, p1*, j*, m2*, p2*);
31: end while
32: return x;
```

O parâmetro de entrada do *External Insertion 2-OPT* é a solução x (Linha 1). As linhas 3 - 5 calculam respectivamente o *makespan*, os coeficientes β^* e γ^* da solução atual. A linha 6 inicializa o armazenamento do melhor par de inserções. O coeficiente auxiliar β_{aux} é calculado para a troca de cada par de tarefas e suas respectivas máquinas na linha 11. A linha 12 verifica se houve melhoria na solução, em caso positivo são atualizados o coeficiente β^* e o armazenamento do melhor par de inserções (Linhas 13 - 14). Após encontrar o melhor par de tarefas, nas linhas 22 e 23, o algoritmo calcula o *makespan* auxiliar e o coeficiente auxiliar γ_{aux} para todas as possíveis posições de inserção das tarefas i^* e j^* determinadas previamente. Em caso de aprimoramento da solução, o coeficiente γ^* e o armazenamento do melhor par de inserções são respectivamente atualizados nas linhas 25 e 26. Por fim, se houve melhoria, a solução x é atualizada na linha 30. O *External Insertion 2-OPT* também é realizado enquanto houver melhoria da solução.

A terceira e última vizinhança, a ser aplicada a busca local, é o *External Swap*, $N_k(x) = swap$. O *External Swap* é definido pela troca entre uma tarefa i, designada na máquina com maior tempo de processamento m_1, com uma tarefa j, que está designada em alguma outra máquina m_2. A FIGURA 13 apresenta um exemplo de uma solução após a aplicação do *External Swap*. Neste exemplo é considerado que as máquinas em que as tarefas serão trocadas já estão definidas.

FIGURA 13 – EXEMPLO *EXTERNAL SWAP*.



FONTE: O autor (2020).

A realização destas trocas, impacta na quantidade total de recursos no sequenciamento. Portanto, para avaliar o impacto disto na solução é utilizado um coeficiente β (4.29), avaliando todas as trocas possíveis e realizando a troca que apresentar o menor valor para o coeficiente β, desde que seja inferior ao coeficiente da solução atual β^*. Os detalhes deste procedimento são descritos no Algoritmo 7:

Algoritmo 7 Pseudocódigo do External Swap

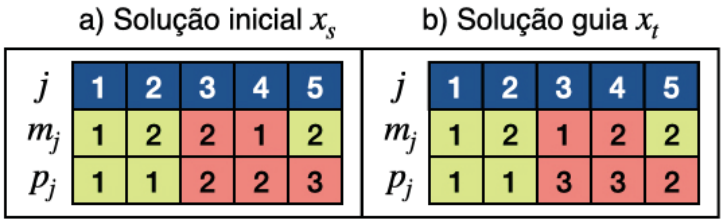
```
1: Input: x
2: while (Condição de Parada) do
3:     β* ← Calcular_Beta();
4:     C*max ← Calcular_Makespan();
5:     Best_Swap(i*,m*1 j*, m*2) ← Ø;
6:     m1 ← argmax_{m∈M}{Cm};
7:     for each i, tal que mi = m1 do
8:         for each m2 ∈ M and m1 ≠ m2 do
9:             for each j, tal que mj = m2 do
10:                βaux ← Calcular_Beta(i, m1, j, m2);
11:                Caux ← Calcular_Makespan(i, m1, j, m2);
12:                if (βaux < β*) and (Caux ≤ C*max) then
13:                    C*max ← Caux;
14:                    β* ← βaux;
15:                    Best_Swap(i*, m*1, j*, m*2) ← (i, m1 j, m2);
16:                end if
17:            end for
18:        end for
19:    end for
20:    x ← Best_Swap(i*, j*, m*);
21: end while
22: return x;
```

O parâmetro de entrada do *External Swap* é a solução x (Linha 1). O coeficiente β^* e o *makespan* da solução atual são calculados nas linhas 3 e 4. O armazenamento da melhor troca a ser realizada é realizado na linha 5. A linha 7 é responsável por identificar a máquina m_1^* em que o *makespan* ocorre. Na linha 10, o coeficiente auxiliar β_{aux} é calculado para cada troca possível e devidamente comparado ao coeficiente da solução atual β^* (Linha 12). Caso seja inferior, o *makespan*, o coeficiente β^* e o armazenamento da melhor troca são atualizados (Linhas 13 - 15). Finalmente, se necessária, a melhor troca é realizada na linha 22. Da mesma forma que os outros procedimentos, este é realizado até que a solução não seja aprimorada.

O GRASP possui uma natureza randômica, o que garante a geração de diversas soluções distintas, com isto é possível construir conjuntos de soluções distintas, tal como o conjunto E, de soluções elites, utilizado no *path-relinking*. Originalmente proposto por Glover (1997), o *path-relinking* é uma estratégia que conecta duas soluções com o propósito encontrar melhores soluções na trajetória desta conexão. Dada uma

solução inicial x_s gerada pelo GRASP, é selecionada uma solução guia do conjunto elite, $x_t \in E$. Estas soluções, x_s e x_t, devem ser distintas, ou seja, $\Delta(x_s, x_t) > 0$. Para calcular a distância entre as soluções são analisadas as características da soluções. Para os problemas UPMR e UPMSR, as características analisadas são a máquina m_j, em que a tarefa j foi designada e sua respectiva posição p_j. A FIGURA 14 exemplifica a solução inicial x_s, a solução guia x_t e suas respectivas características.

FIGURA 14 – EXEMPLO DE CARACTERÍSTICAS DA SOLUÇÃO INICIAL E DA SOLUÇÃO GUIA



FONTE: O autor (2020)

No exemplo acima, podemos ver que as características em comum entre as soluções, destacadas em verde, e as características diferentes, destacadas em vermelho. Para realizar o cálculo do valor de $\Delta(x_s, x_t)$, são analisadas as características de todas as tarefas j, das soluções, x_s e x_t. Para cada tarefa j, que esteja designada em máquinas distintas nas soluções, é acrescido 1 ponto ao valor de $\Delta(x_s, x_t)$. No entanto, caso a tarefa j esteja designada para a mesma máquina, mas com posições diferentes, é acrescido 0.5 ponto ao valor de $\Delta(x_s, x_t)$. Em caso, das características da tarefa j, designação de máquina e posição, sejam as mesmas, nenhum ponto é acrescido ao valor de $\Delta(x_s, x_t)$.

Calculando o valor de $\Delta(x_s, x_t)$ para o exemplo da apresentado na FIGURA 14, temos: Para ambas as soluções, as tarefas 1 e 2 possuem características iguais, ou seja, estão designadas para as mesmas máquinas nas mesmas posições. Portanto, para estas duas tarefas nenhum ponto é adicionado, $\Delta(x_s, x_t) = 0$. As tarefas 3 e 4 possuem designação de máquinas distintas, em relação as soluções, então são adicionados dois pontos, um para cada tarefa, $\Delta(x_s, x_t) = 2$. A tarefa 5 possui designação de máquina igual, porém a posição é diferente, por este motivo é adicionado 0.5 ponto, $\Delta(x_s, x_t) = 2.5$. Por fim, é calculado o percentual da relação entre $\Delta(x_s, x_t) = 2.5$ e a quantidade de tarefas $|J| = 5$. Assim, o valor resultante é de $\Delta(x_s, x_t) = 0.5$.

O cálculo do valor de $\Delta(x_s, x_t)$ é detalhado no Algoritmo 8. Para facilitar a compreensão deste algoritmo considere, a máquina m_j em que a tarefa j foi processada para a solução x_s, como $m_j(x_s)$. A posição em que a tarefa j foi processada na máquina m_j para a solução x_s, como $p_j(x_s)$. A máquina m_j em que a tarefa j foi processada para

a solução x_t, como $m_j(x_t)$. E a posição em que a tarefa j foi processada na máquina m_j para a solução x_t, como $p_j(x_t)$.

Algoritmo 8 Pseudocódigo do Cálculo de $\Delta(x_s, x_t)$

```
1: Input: x_s, x_t
2: Δ(x_s, x_t) = 0;
3: for each j ∈ J do
4:     if m_j(x_s) ≠ m_j(x_t) then
5:         Δ(x_s, x_t) = Δ(x_s, x_t) + 1;
6:     else if p_j(x_s) ≠ p_j(x_t) then
7:         Δ(x_s, x_t) = Δ(x_s, x_t) + 0.5;
8:     end if
9: end for
10: Δ(x_s, x_t) = Δ(x_s, x_t)/|J|;
11: return Δ(x_s, x_t);
```

Os parâmetros de entrada do procedimento para calcular o valor de $\Delta(x_s, x_t)$ são a solução inicial x_s e a solução guia x_t (Linha 1). A linha 2 é responsável por inicializar o valor de $\Delta(x_s, x_t)$. É verificado a diferença das soluções para todas as tarefas $j \in J$ (Loop 3 - 9). Na linha 4, é verificado se a tarefa j é designada para máquinas diferentes nas soluções ($m_j(x_s) \neq m_j(x_t)$). Caso positivo, é adicionado 1 ponto ao valor de $\Delta(x_s, x_t)$ (Linha 5). Em caso negativo, na linha 6, é verificado se a tarefa j está alocada em posições diferentes nas soluções ($p_j(x_s) \neq p_j(x_t)$). Em caso positivo para a última verificação, é adicionado 0.5 ponto ao valor de $\Delta(x_s, x_t)$ (Linha 7). Por fim, o valor de pontos acumulados de $\Delta(x_s, x_t)$ é divido pela quantidade de tarefas $|J|$ (Linha 11).

Ao analisar as soluções, suas características em comum se mantém intactas e o objetivo é inserir iterativamente as características da solução guia que não estejam presentes na solução inicial. Para que isto seja possível, existe um conjunto de movimentos possíveis para realizar esta conexão. Utilizando o exemplo apresentado na FIGURA 14, podemos criar a lista de movimentos possíveis para que x_s seja religada à x_t. Como existem três tarefas, que possuem características diferentes nas soluções x_s e x_t, a lista contém três movimentos possíveis para realizar a reconexão: (i) Inserir a tarefa 3 na posição 3 da máquina 1; (ii) Inserir a tarefa 4 na posição 2 da máquina 2; (iii) Inserir a tarefa 5 na posição 2 da máquina 1. Com isso, podemos definir que serão realizadas três iterações até que as soluções sejam reconectadas. Para fins de exemplo, vamos selecionar o primeiro movimento possível, inserir a tarefa 3 na posição 3 da máquina 1. Ao realizar este movimento, é possível obter a solução intermediária x'_s e o valor atualizado de $\Delta(x'_s, x_t) = 0.3$. Vale ressaltar que a cada movimento o valor de $\Delta(x_s, x_t)$ diminuí, pois a cada movimento as soluções se tornam cada vez mais similares. Os detalhes desta solução intermediária são apresentados na FIGURA 15.

FIGURA 15 – EXEMPLO DE CARACTERÍSTICAS DA SOLUÇÃO INTERMEDIÁRIA E DA SOLUÇÃO GUIA

a) Solução intermediária x'_s

j	1	2	3	4	5
m_j	1	2	1	1	2
p_j	1	1	3	2	3

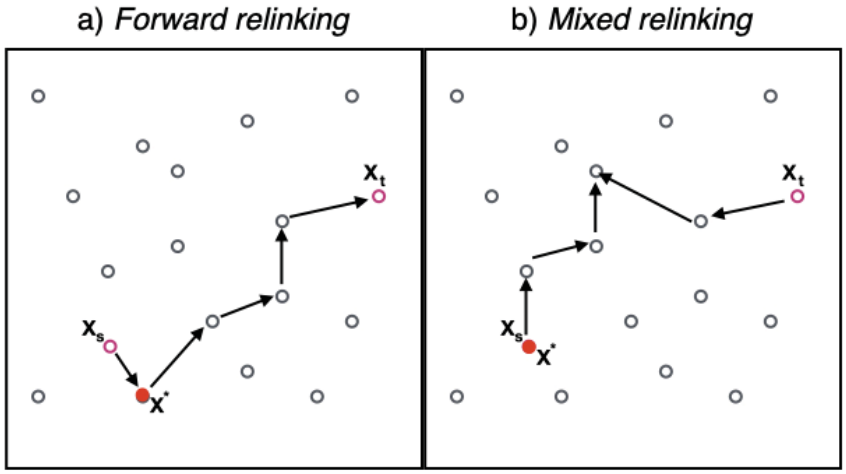
b) Solução guia x_t

j	1	2	3	4	5
m_j	1	2	1	2	2
p_j	1	1	3	3	2

FONTE: O autor (2020)

A cada iteração, todas as possibilidades de movimentos são analisadas e, usualmente, a melhor possibilidade é selecionada. Cada movimento representa um movimento no espaço de soluções, similar às movimentações que ocorrem nas buscas locais, como pode ser visto na FIGURA 16. Os movimentos são realizados até que as soluções sejam idênticas $(\Delta(x_s, x_t) = 0)$ e a cada movimento uma nova solução é encontrada. Por fim, a melhor solução x^* encontrada nesta trajetória é selecionada.

FIGURA 16 – EXEMPLO DE TRAJETÓRIAS DO *PATH-RELINKING*.



FONTE: O autor (2020)

Resende e Ribeiro (2005a) apontam que diversas estratégias de trajetórias já foram consideradas e até mesmo implementadas em conjunto, tais como:

- *Forward relinking*: A pior solução entre x_s e x_t é selecionado como solução inicial e a outra como guia;
- *Backward relinking*: A melhor solução entre x_s e x_t é selecionado como solução inicial e a outra como guia;
- *Back and forward relinking*: Duas trajetórias são exploradas, primeiro utilizando x_s como solução inicial e depois x_t;

- *Mixed relinking:* Os dois sentidos da trajetória são explorados intercaladamente, até que ambos se encontrem em uma solução intermediária equidistante;
- *Truncated relinking:* Apenas uma parte da trajetória entre as soluções x_s e x_t é examinada, determinada por um índice de truncamento τ.

A solução x^*, encontrada ao final de cada *path-relinking*, é considerada candidata para entrar no conjunto de soluções elite E, desde que seja suficientemente diferente de todas as soluções do conjunto, ou seja, a distância entre as soluções deve ser maior que o índice de diversidade δ, tal que, $\Delta(x^*, x_e) > \delta \; \forall x_e \in E$. Além do mais, caso o conjunto elite esteja completo, $|E| = E_{max}$, a solução candidata somente poderá ser inserida no conjunto se superar alguma solução do conjunto elite, devendo assim, ser realizada a seleção da solução a ser retirada. Para isso, as soluções inferiores à solução candidata são analisadas e a solução mais similar é selecionada, ou seja, $argmim_e\{\Delta(x^*, x_e)\}$, tal que, $\forall x_e \in E$, $C(x^*) < C(x_e)$. Caso o conjunto elite não esteja completo, $|E| < E_{max}$, a solução é simplesmente adicionada ao conjunto. Desta forma, este gerenciamento de inserção e remoção, garante tanto a qualidade quanto a diversidade nas soluções presentes no conjunto elite (RESENDE; WERNECK, 2004). Este processo é repetido até que o critério de parada seja satisfeito.

Conforme dito anteriormente, usualmente no *path-relinking*, o melhor movimento é selecionado até que a solução inicial alcance a solução guia. Desta forma, a trajetória sempre será a mesma de acordo com uma das estratégias apontadas por Resende e Ribeiro (2005a). Com o intuito de obter soluções diversificadas, Binato, Faria Jr. e Resende (2001) introduziram o *Greedy Randomized Adaptive Path-Relinking* (GRAPR), que aplica conceitos do GRASP, afim de randomizar os movimentos de uma trajetória do *path-relinking*. Para isso, uma RCL é criada com os movimentos possíveis e cada movimento tem seu impacto na função objetivo avaliado. Da mesma forma que o GRASP, o tamanho da RCL é determinada de acordo com o índice de gulosidade, aqui denominado como $\alpha_{GRAPR} \in [0, 1]$. A seleção do movimento é realizada aleatoriamente, com probabilidade de acordo com impacto gerado, ou seja, quanto melhor o impacto, maior a probabilidade desta solução ser selecionada. O valor de impacto de cada movimento na solução, através do coeficiente η dado por:

$$\eta = \frac{C_{max} - \beta}{C_{max}} \tag{4.32}$$

Resende e Werneck (2004) também introduziram o *Evolutionary Path-Relinking* (EPR), como um método de pós-otimização, com objetivo de aprimorar a solução obtida pelo GRAPR. O conjunto de soluções elites resultante do GRAPR, é utilizado como conjunto elite inicial E_0 no EPR. A cada iteração k, a reconexão de todos os pares de soluções do conjunto E_k é realizada e a solução de cada aplicação se torna candidata

do conjunto E_{k+1} da próxima iteração. Os critérios de inserção de uma solução no conjunto E_{k+1} são os mesmos utilizados no *path-relinking*. O processo é repetido até que nenhuma melhoria seja encontrada.

Para o presente trabalho foi desenvolvida uma abordagem do *Greedy Randomized Adaptive Path-Relinking*, que utiliza o *Evolutionary Path-Relinking* a cada ψ iterações e na fase de pós-otimização, que será intitulado como *Greedy Randomized Adaptive Evolutionary Path-Relinking* (GRAEPR). Nesta abordagem serão testadas duas estratégias de trajetórias, *forward relinking* e *mixed relinking*, juntamente com o *truncated relinking*, utilizando o índice de truncamento τ para determinar o tamanho das trajetórias. O pseudocódigo é descrito no Algoritmo 9:

Algoritmo 9 Pseudocódigo do GRAEPR

```
1:  Input: α_GRASP, α_GRAPR, δ, E_max, τ, Estrategia, ψ
2:  E ←Inicializar_Elite(δ, E_max);
3:  k = 0;
4:  while (Condição de Parada) do
5:      x_s ← GRASP(α_GRASP);
6:      x_t ← Seleciona_Solução_Guia(E, x_s);
7:      x_s ← GRAPR(x_s, x_t, α_GRAPR, τ, Estrategia);
8:      x_s ← Busca_Local();
9:      E ← Atualizar_Elite(δ, x_s);
10:     if (mod(k, ψ) = 0) then
11:         E ← EPR(α_GRAPR, τ, Estrategia, δ, E);
12:     end if
13:     k ← k + 1;
14: end while
15: E ← EPR(α_GRAPR, τ, Estrategia, δ, E);
16: return x*;
```

Os parâmetros de entrada para o GRAEPR são, o índice de gulosidade do GRASP α_{GRASP}, o índice de gulosidade do GRAPR α_{GRAPR}, o índice de diversidade, δ, o tamanho estabelecido para o conjunto elite E_{max}, o índice de truncamento τ, a estratégia de reconexão das soluções $Estrategia$ e quantidade de interações para aplicar o EPR ψ (Linha 1).

A linha 2 inicializa o conjunto de soluções elite E com tamanho E_{max}. Estas soluções são distintas e com valor de $\Delta(x_s, x_t) \geq \delta$. A linha 3 inicializa o contador k. De forma iterativa, os procedimentos são realizados até que a condição de parada seja satisfeita (Linha 4). Neste trabalho, o GRAEPR é realizado enquanto houver uma solução no conjunto elite que ainda não foi utilizada como guia. A linha 5 é responsável pela geração de uma solução inicial x_s utilizando o GRASP apresentado anteriormente (Algoritmo 2). A seleção da solução guia ocorre na linha 6, com probabilidade de acordo com a distância simétrica da solução x_s com as soluções do conjunto elite ($\Delta(x_s, x_e)$ $\forall x_e \in E$). Quanto mais distante de x_s, maior a probabilidade de ser selecionada como

solução guia. Na linha 7 é aplicado o *Greedy Randomized Adaptive Path-Relinking* nas soluções x_s e x_t, com índice de gulosidade α_{GRAPR}, com o índice de truncamento τ e de acordo com a estratégia definida. Após este procedimento, será retornada a melhor solução obtida na trajetória entre as duas soluções. Após obter a solução gerada pelo GRAPR, na linha 7, é aplicado uma busca local (Algoritmo 4) para aprimorar a solução (Linha 8). Na linha 14 o conjunto E é atualizado de acordo com critério de inserção e remoção do conjunto elite, analisando a solução candidata e o índice de diversidade δ. Na linha 10 é verificado a quantidade de iterações realizadas e a cada ψ iterações é aplicado o *Evolutionary Path-Relinking* (Linha 11). O contador k é incrementado na linha 13. Por fim, na linha 15 o EPR é aplicado em fase de pós-otimização e a melhor solução x^* é retornada na linha 16.

O Algoritmo 10 descreve as operações realizadas ao aplicar o *Greedy Randomized Adaptive Path-Relinking* nas soluções x_s e x_t. Para esta aplicação são necessários como parâmetros as duas soluções que serão conectadas, o índice de gulosidade do GRAPR α_{GRAPR}, o índice de truncamento τ, que determina a quantidade de movimentos realizados na trajetória e a estratégia determinada para percorrê-la.

Algoritmo 10 Pseudocódigo do GRAPR

```
1: Input: x_s, x_t, α_GRAPR, τ, Estrategia
2: x* ← x_t;
3: RCL ← Inicializar_RCL(α_GRAPR);
4: k = 0;
5: while (k < RCL.size() * τ) do
6:     x_s ← Remover_Ocioso();
7:     (j, m, p) ← Seleção_Aleatória(RCL);
8:     x_s ← Alterar tarefa j para posição p da máquina m;
9:     x_s ← Reparar_Solução();
10:    if (C_max < C*_max) then
11:        x* ← x_s;
12:    end if
13:    if (Estrategia = Mixed) then
14:        Inverter_Sentido_Trajetória(x_s, x_t);
15:    end if
16:    Atualizar_RCL(α_GRAPR);
17:    k ← k + 1;
18: end while
19: return x*;
```

Os parâmetros de entrada para o GRAPR são, a solução inicial x_s, a solução guia x_t, o índice de gulosidade do GRAPR α_{GRAPR}, o índice de truncamento τ e a estratégia de reconexão das soluções $Estrategia$ (Linha 1). A linha 2 atribui a solução alvo x_t à melhor solução. Na linha 3, a lista restrita de candidatos é inicializada, inserindo todos os movimentos possíveis para conectar a solução x_s à x_t, de acordo com o índice

de gulosidade α_{GRAPR}. Ao inicializar a RCL, também são calculados os valores de impacto de cada movimento na solução, através do coeficiente η (4.32).

Na linha 4 o contador k é iniciado. A quantidade de movimentos realizados para conectar as soluções é determinada pelo índice de truncamento τ (Linha 5). Da mesma forma que na busca local, os movimentos são avaliados nas soluções sem ociosidade, que é removida na linha 6. A linha 7 seleciona aleatoriamente um movimento, onde uma tarefa j será inserida na posição p da máquina m. A probabilidade de seleção de um movimento é de acordo com o valor do seu coeficiente η, quanto menor o valor de η, maior a probabilidade deste movimento ser selecionado. O movimento selecionado é realizado e a solução x_s é atualizada na linha 8. A solução é reparada na linha 9 e tem sua função objetivo avaliada (Linha 10). Caso a estratégia selecionada seja *mixed relinking*, o sentido da trajetória deve ser trocado intercaladamente (Linha 14). Então, a lista restrita de candidatos é atualizada, na linha 16, tendo o coeficiente η de cada movimento recalculado. Para respeitar a quantidade de movimentos a ser realizado, o contador k é atualizado na linha 17. Por fim, o método retorna a melhor solução obtida.

O Algoritmo 11 apresenta o as operações realizadas no *Evolutionary Path-Relinking*, utilizado para diversificar as soluções do conjunto elite E e na fase de pós-otimização do GRAEPR.

Algoritmo 11 Pseudocódigo do EPR

```
 1: Input: α_GRAPR, τ, Estrategia, δ, E
 2: E_0 ← E;
 3: k = 0;
 4: while (Condição de Parada) do
 5:     for each par x_s, x_t ∈ E_k do
 6:         x_s ← GRAPR(x_s, x_t, α_GRAPR, τ, Estrategia);
 7:         x_s ← Local_Search();
 8:         E_{k+1} ← Atualizar_Elite(δ, x_s);
 9:     end for
10:     k ← k + 1;
11: end while
12: return E_k;
```

Os parâmetros de entrada para o EPR são, o índice de gulosidade do GRAPR α_{GRAPR}, o índice de truncamento τ, a estratégia de reconexão das soluções $Estrategia$, o índice de diversidade δ e o conjunto elite E (Linha 1). Na Linha 2, o conjunto elite da primeira iteração E_0 recebe as soluções presentes no conjunto elite atual. O contador k é inicializado na linha 3. A condição de parada é verificada na linha 4. Para o EPR foi estabelecido que o algoritmo será repetido enquanto houver aprimoramento na melhor solução do conjunto. A cada iteração, é aplicado o GRAPR em cada par de soluções x_s e x_t presente no conjunto elite E_k (Linha 6) e a busca local é aplicada na solução obtida (Linha 7). Na linha 8, o conjunto elite da próxima iteração é atualizado de acordo

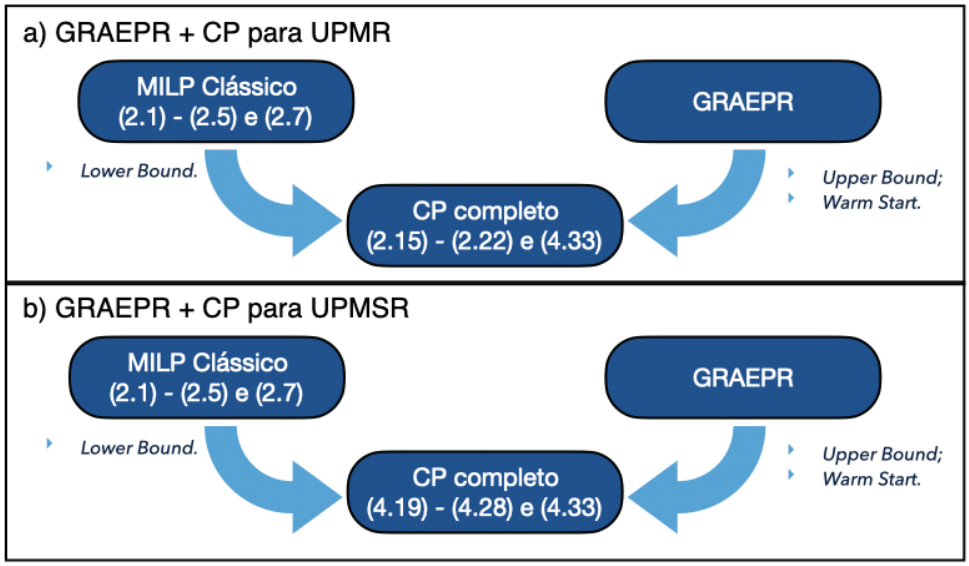
com as mesmas regras utilizadas no GRAEPR. Na linha 10 o contador de iterações é atualizado. Por fim, o último conjunto elite E_k é retornado.

4.4 ABORDAGEM GRAEPR COM PROGRAMAÇÃO POR RESTRIÇÃO

Conforme estabelecido na revisão de literatura, diversas pesquisas tem utilizado modelos de programação por restrição para resolver o problema de máquinas paralelas não relacionadas, no entanto, abordagens exatas tem custo computacional elevado. Inspirado pelas ferramentas já estabelecidas na literatura e visando utilizar as características positivas de métodos exatos e heurísticas é desenvolvido uma abordagem híbrida que utiliza o GRAEPR, apresentado na Seção 4.3 e um modelo de programação por restrição (GRAEPR + CP). O GRAEPR tem a capacidade de fornecer uma solução de alta qualidade em um tempo computacional consideravelmente pequeno, com isso, a solução obtida pode ser utilizada como *Upper Bound* e como *Warm Start* do CP.

Além disso, da mesma forma que Fleszar e Hindi (2018), será utilizado o resultado obtido do modelo (2.1) - (2.5) com a adição da restrição (2.7) como *Lower Bound* desta abordagem. As estruturas desta abordagem, para os problemas UPMR e UPMSR, são apresentadas na FIGURA 17.

FIGURA 17 – ABORDAGEM GRAPR COM CP



FONTE: O autor (2020)

Para o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR), na fase do modelo de programação por restrição, será utilizado o modelo (2.15) - (2.22), proposto por Fleszar e Hindi (2018). No entanto, este modelo não considera a utilização de *Upper Bound*, com isto, para auxiliar em

sua convergência, será adicionada a restrição (4.33), onde o valor de UB é o valor da solução utilizada como *Warm Start*.

$$C_{max} \leq UB \tag{4.33}$$

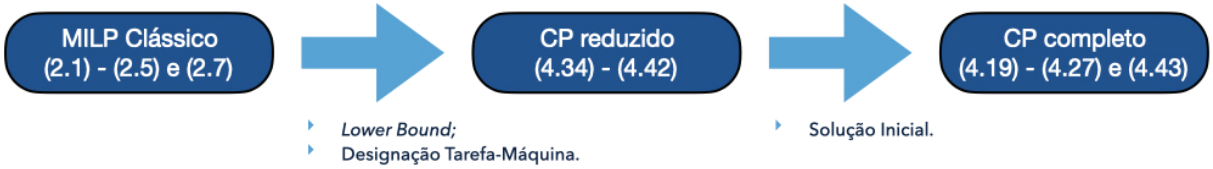
Agora, para o problema de máquinas paralelas não relacionadas com *setup* dependente e com restrição de recursos renováveis (UPMSR), será utilizado o modelo desenvolvido nesta pesquisa (4.19) - (4.28) também com a adição desta restrição de *Upper Bound* (4.33).

4.5 ADAPTAÇÃO DA ABORDAGEM HÍBRIDA DE FLESZAR E HINDI

De acordo com Lenstra, Rinnooy Kan e Brucker (1977), o UPM e suas variantes são consideradas NP-*Hard*, sendo assim, devido a maior complexidade do UPMSR, este também pode ser considerado pertencente a esta classe. Como já mencionado, problemas da classe NP-*Hard* são essencialmente difíceis de resolver e requerem tempo computacional elevado. Em vista disso, também é proposto uma adaptação da abordagem híbrida elaborada por Fleszar e Hindi (2018) (AFH), a qual foi comprovada como uma forte ferramenta para este tipo de solução.

Conforme apresentada na Seção 2.3, esta abordagem consiste três estágios. Primeiramente é estabelecido um *Lower Bound* utilizando o modelo apresentado em (2.1) - (2.5) e (2.7). O segundo estágio, consiste em um modelo que aborda uma versão reduzida do problema. O objetivo deste modelo reduzido é definir o sequenciamento das tarefas, de acordo com a designação tarefa-máquina obtida no estagio anterior e utilizar esta solução como ponto inicial da próxima fase. A última fase é um modelo de *constraint programming* que aborda o problema como um todo. Portanto, para utilizar esta abordagem no problema UPMSR, se faz necessário desenvolver os modelos de ambas as fases. Conforme apresentado na FIGURA 18

FIGURA 18 – ABORDAGEM HÍBRIDA DE FLESZAR E HINDI (2018)



FONTE: O autor (2020).

Para a segunda fase, o modelo foi desenvolvido baseado no modelo proposto por Fleszar e Hindi (2018). As notações utilizadas para a modelagem em *constraint programming* são apresentadas na TABELA 8.

TABELA 8 – NOTAÇÕES UTILIZADAS NO MODELO DE *CONSTRAINT PROGRAMMING* PARA O UPMSR REDUZIDO

Conjuntos e índices	
J	Tarefas; $i, j \in J = 1...\|J\|$;
M	Máquinas; $m \in M = 1...\|M\|$;
Parâmetros	
m_j	máquina m em que a tarefa j é pré-designada;
P_{jm}	Tempo de processamento da tarefa j na máquina m;
S_{ijm}	Tempo de *Setup* entre as tarefas i e j (sendo a tarefa i imediatamente antecessora da tarefa j) na máquina m;
Res_{jm}	Recursos necessários para realizar a tarefa j na máquina m;
Res_{max}	Limite de recurso disponível para a realização das tarefas;
LB	*Lower Bound*;
Variáveis de Decisão	
C_{max}	Instante máximo de conclusão das tarefas ou *makespan*;
Y_j	Variável de intervalo relacionada à tarefa j;
W_m	Variável de sequência de intervalos relacionada ao sequenciamento que será executado na máquina m;
Z_m	Variável de transição que representa os tempos de *setup* executados no sequenciamento da máquina m;
$Resource$	Função Acumulativa;

FONTE: O autor (2020).

Min C_{max} (4.34)

s.a.

$$Y_j = size(P_{jm_j}) \qquad \forall j = 1..|J|; \forall m = 1..|M|; \tag{4.35}$$

$$noOverlap(Y_{jm_j}, \forall j, \text{ tal que} m = m_j) \qquad \forall m = 1..|M|; \tag{4.36}$$

$$Z_{m_j} = S_{ijm_j} \qquad \forall m = 1..|M|; \tag{4.37}$$

$$noOverlap(W_{m_j}, Z_{m_j}) \qquad \forall m = 1..|M|; \tag{4.38}$$

$$Resource = \sum_{j=1}^{|J|} pulse(Y_j, R_{jm_j}) \tag{4.39}$$

$$Resource \leq R_{max} \tag{4.40}$$

$$C_{max} \geq endOF(Y_j) \qquad \forall j = 1..|J|; \tag{4.41}$$

$$C_{max} \geq LB \tag{4.42}$$

A função objetivo (2.8) é minimizar o *makespan*. O conjunto de restrição (2.9) atribui o tempo de processamento P_{jm_j} à variável de intervalo relacionada à tarefa j. A não sobreposição das tarefas é garantida na restrição (2.10). O conjunto de restrição (4.37) atribui os valores da matriz de *setup*, relacionada à máquina m, à

variável de transição Z_m. A não sobreposição dos *setups* é garantida pelas restrições (4.38). A restrição (4.39) define a função acumulativa $Resource$, como a soma dos recursos utilizados pela tarefa j na máquina pré-designada m_j. A garantia que os recursos respeitam o limite determinado é realizada pela restrição (4.40). O conjunto de restrições (4.41) garante que o *makespan* é o maior instante de conclusão. Por fim, a restrição (4.42) garante que o *makespan* respeite o valor de *Lower Bound*, afim de acelerar a convergência da solução.

Para a terceira fase, será utilizado o modelo proposto na 4.2 (4.19) - (4.28). Para esta abordagem, existe a necessidade de uma restrição que respeite o *Lower Bound* obtido na primeira fase, portanto é removida a restrição (4.28) e adicionada a restrição (4.43).

$$C_{max} \geq LB \tag{4.43}$$

5 RESULTADOS COMPUTACIONAIS

O objetivo deste capítulo é analisar a performance das abordagens propostas avaliando sua eficiência. Para realizar a implementação computacional foi utilizada a linguagem de programação C++ e utilizado o solver CPLEX v.12.8, com número de *threads* igual à 1 e os outros parâmetros *default*. Os testes foram realizados no cluster Graham da compute Canada, utilizando 2 processadores Intel Xeon Gold 5120 de 2.2GHz e 4Gb de memória RAM, com tempo limite estipulado de 3600 segundos. As instâncias, os códigos e os resultados detalhados estão disponíveis *online* [1].

5.1 RESULTADOS COMPUTACIONAIS PARA O UPMR

Para a realização dos testes, para o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR), foram utilizados dois conjuntos de instâncias presentes na literatura, o primeiro proposto por Fanjul-Peyro, Perea e Ruiz (2017), o qual contém instâncias de pequeno e médio porte, e o segundo proposto por Fleszar e Hindi (2018), com instâncias de grande porte. Ambos os conjuntos foram gerados utilizando a ferramenta desenvolvida por Fanjul-Peyro, Perea e Ruiz (2017).

Em relação aos tempos de processamento P_{jm} foram gerados de 5 maneiras diferentes: três envolvendo distribuições uniformes, $U(1, 100)$, $U(10, 100)$ e $U(100, 200)$ e as outras duas envolvendo correlações, uma entre tarefas e a outra entre máquinas. Os recursos R_{im} foram gerados por duas maneiras: relacionados aos tempos de processamento e por uma distribuição uniforme, podendo receber valores inteiros entre 1 e 9. O limite de recursos foi definido como 5 vezes o número de máquinas, $R_{max} = 5m$. Além disso, para cada configuração de parâmetros, de acordo com as quantidades de tarefas e máquinas, foram geradas cinco replicações. Portanto, tomando todos os valores em conta, para cada configuração existem $5x2x5 = 50$ instâncias.

Para instâncias de pequeno (8, 12 e 16 tarefas) e médio porte (20, 25 e 30 tarefas), foram considerados apenas três quantidades máquinas (2, 4 e 6 máquinas), resultando em 150 instâncias para cada quantidade de tarefa determinada. Para as grandes instâncias (50, 100, 500 e 1000 tarefas) são consideradas cinco quantidades de máquinas (2, 4, 6, 10 e 15 máquinas), resultando em 250 instâncias para cada quantidade de tarefa. Estas são as configurações resultam nas 1900 instâncias utilizadas, 450 pequenas (8,12 e 16 tarefas), 450 médias e 1000 de tamanho grande. As configurações para a geração das instâncias são apresentadas na TABELA 9. Vale ressaltar que as instâncias, utilizadas neste trabalho, consideram que as tarefas necessitam

[1] Acesse *https://1drv.ms/u/s!AonCIGhquVnsiZp2IXzdl4ONL8gL3w?e=OmgfnY*.

apenas um tipo de recurso para o seu processamento.

TABELA 9 – PARÂMETROS PARA GERAÇÃO DE INSTÂNCIAS

Tarefas	**Máquinas**	**# Instâncias**
8	{2, 4, 6 }	150
12	{2, 4, 6 }	150
16	{2, 4, 6 }	150
20	{2, 4, 6 }	150
25	{2, 4, 6 }	150
30	{2, 4, 6 }	150
50	{2, 4, 6, 10, 15 }	250
100	{2, 4, 6, 10, 15 }	250
500	{2, 4, 6, 10, 15 }	250
1000	{2, 4, 6, 10, 15 }	250

FONTE: O autor (2020).

Como descrito, para utilização do GRAEPR são requeridos alguns parâmetros, os quais devem ser configurados para obter a melhor performance possível. Métodos como o IRACE, apresentado por López-Ibánez et al. (2016), são muito utilizados para automatizar esta difícil tarefa. Este método realiza testes e busca de forma iterativa encontrar a configuração ideal, de acordo com os intervalos estabelecidos. Os parâmetros que necessitam de configuração, seus respectivos intervalos e o valores resultantes obtidos pelo IRACE são apresentados na TABELA 10:

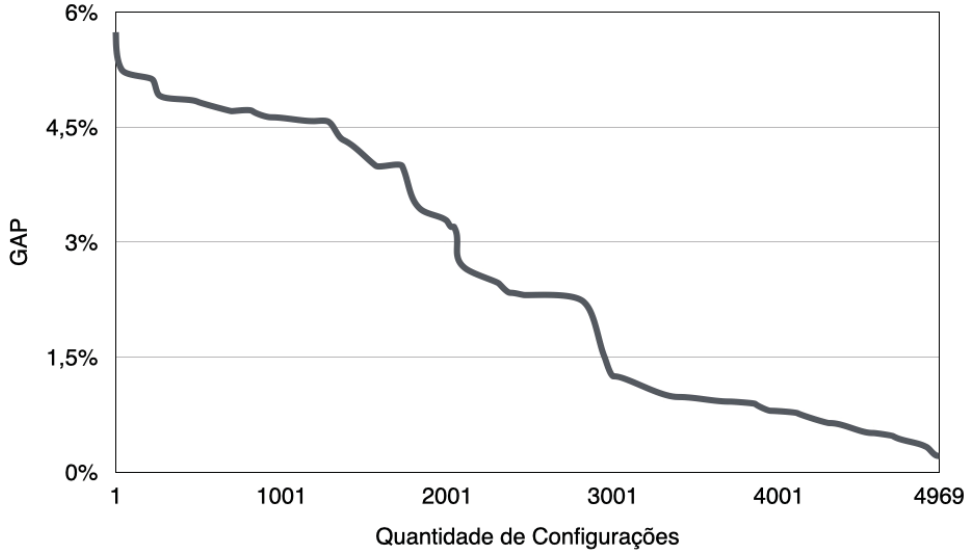
TABELA 10 – PARÂMETROS PARA O UPMR

Parâmetro	**Intervalo**	**Valor**
Estratégia	*forward*, *mixed*	*mixed*
α_{GRASP}	$[0.3, 1.0]$	0.5730
α_{GRAPR}	$[0.0, 1.0]$	0.8950
E_{max}	$[10, 20]$	20
τ	$[0.3, 1.0]$	0.5636
δ	$[0.2, 0.5]$	0.2641
ψ	$[2 * E_{max}, 10 * E_{max}]$	$2 * E_{max}$

FONTE: O autor (2020).

Para validação do método, foi estabelecido um limite de 50000 iterações e 50 instâncias foram selecionadas aleatoriamente para a realização dos testes. Com estas definições, o IRACE estabeleceu e testou 4969 combinações de configurações.

FIGURA 19 – EVOLUÇÃO DAS SOLUÇÕES DO IRACE PARA O UPMR



FONTE: O autor (2020)

A FIGURA 19 apresenta a evolução da qualidade das soluções encontradas pelo IRACE para as 4969 configurações estabelecidas. Mostrando a variação da média do GAP, calculado entre a melhor solução obtida e a solução da configuração. O valor máximo de GAP obtido foi de 5,74% e o valor mínimo de 0,21%.

Lembrando que conforme a tabela TABELA 5, os métodos de resolução utilizados, para o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR), são o *Greedy Randomized Adaptive Path-Relinking* (GRAEPR) e a abordagem híbrida do GRAEPR com um modelo de programação por restrição.

Afim de expor os dados obtidos da melhor maneira, os resultados são apresentados em conjuntos de instâncias, classificados pela quantidade de tarefas. A TABELA 11 apresenta os resultados obtidos ao resolver todas as instâncias para o problema UPMR. A TABELA 11 exibe a média e a porcentagem máxima dos resultados obtidos pelos dois métodos apresentados neste trabalho e os resultados apresentados no trabalho de Fleszar e Hindi (2018). O percentual do GAP apresentado nesta tabela é calculado utilizando os valores de *Lower* e *Upper Bounds* obtido em cada método. Por questões de comparação, dado que o método GRAEPR não fornece um *Lower Bound*, para este método é utilizado o melhor *Lower Bound* encontrado por um dos outros dois métodos. O cálculo do percentual do GAP é apresentado na equação (5.1).

$$\%GAP = \frac{(UB - LB)}{UB} \tag{5.1}$$

TABELA 11 – PERCENTUAL DO GAP DOS RESULTADOS DO UPMR

Tarefas	GRAEPR		GRAEPR + CP		Fleszar e Hindi (2018)	
	Média	Máximo	Média	Máximo	Média	Máximo
8	1,10%	21,82%	0,00%	0,00%	0,00%	0,00%
12	1,81%	14,08%	0,00%	0,00%	0,00%	0,00%
16	2,51%	19,75%	0,23%	20,72%	0,00%	0,00%
20	2,96%	16,82%	0,64%	16,15%	0,09%	2,33%
25	3,13%	13,41%	1,70%	21,69%	0,09%	2,55%
30	3,64%	15,15%	0,96%	11,13%	0,09%	2,02%
50	8,58%	34,69%	0,89%	11,36%	0,19%	2,70%
100	10,50%	39,78%	1,73%	25,49%	0,25%	2,55%
500	13,66%	39,84%	2,28%	9,52%	0,85%	3,56%
1000	15,74%	38,03%	2,13%	9,77%	0,95%	3,38%
	7,57%	39,84%	1,20%	25,49%	0,32%	3,56%

FONTE: O autor (2020).

É possível notar que, utilizando apenas o GRAEPR é possível encontrar soluções ótimas locais, no entanto, conforme o tamanho da instância aumenta, as soluções se distanciam mais do *Lower Bound*, mas ainda permanecem com resultados competitivos. Também é comprovado que ao adicionar o método exato é possível fortalecer o GRAEPR. Apesar de, em média, não superar os resultados apresentados por Fleszar e Hindi (2018), em média 1,20% contra 0,32% de GAP. Mesmo assim, o método proposto nesta pesquisa, se prova eficiente e competitivo, pois ao utilizar apenas o GRAEPR é possível obter soluções aproximadas. Além disso, a abordagem híbrida também se provou consistente e com soluções aproximadas aos resultados presentes na literatura.

TABELA 12 – TEMPO DE RESOLUÇÃO, EM SEGUNDOS, DO UPMR

Tarefas	GRAEPR	GRAEPR + CP	Fleszar e Hindi (2018)
8	0	0	0
12	1	2	0
16	2	130	4
20	3	1076	56
25	5	1611	55
30	8	1655	67
50	22	2211	118
100	132	3469	183
500	1127	3598	373
1000	1261	3598	394
	336	2047	167

FONTE: O autor (2020).

Os tempos computacionais são apresentados na TABELA 12. É importante ressaltar que Fleszar e Hindi (2018) determinaram como tempo limite de seus experimentos apenas 600 segundos. A abordagem GRAEPR apresentou uma velocidade de resolução superior as demais para instâncias de porte pequeno e médio, e para instâncias com 50 e 100 tarefas. No entanto, para as instâncias maiores, o método proposto por Fleszar e Hindi (2018) é superior. Ademais, pode-se concluir que, o método heurístico se sobressai, pois consegue fornecer soluções de qualidade em um pequeno tempo computacional. Em virtude, da abordagem híbrida obter resultados consistentes e soluções aproximadas à abordagem de Fleszar e Hindi (2018), os próximos resultados apresentarão comparação entre estes dois métodos.

TABELA 13 – PERCENTUAL DE SOLUÇÕES ÓTIMAS ENCONTRADAS NO UPMR

Tarefas	**Empate**		**GRAEPR + CP**	**Fleszar e Hindi (2018)**
	Sol. Ótima	**Sol. Não Ótima**		
8	100,00%	0,00%	0,00%	0,00%
12	100,00%	0,00%	0,00%	0,00%
16	98,00%	0,00%	0,00%	2,00%
20	71,33%	8,00%	2,67%	18,00%
25	56,67%	5,33%	2,00%	36,00%
30	57,33%	10,00%	0,00%	32,67%
50	43,60%	18,80%	0,40%	37,20%
100	5,20%	29,60%	0,00%	65,20%
500	0,00%	58,80%	0,00%	41,20%
1000	0,00%	61,20%	0,00%	38,80%
	44,58%	24,00%	0,42%	31,00%

FONTE: O autor (2020).

A TABELA 13 apresenta o percentual de soluções ótimas obtidas em ambos os métodos. Os resultados mostram que ambos os métodos, o GRAEPR com *constraint programming* e o método proposto por Fleszar e Hindi (2018), comprovaram otimalidade em 44,58% (847 instâncias) e 24,00% (456 instâncias) dos resultados ainda permanecem sem solução ótima encontrada. Em 31,00% (589 instâncias), não foi possível alcançar os resultados obtido por Fleszar e Hindi (2018), os quais apresentaram solução ótima. No entanto, o GRAEPR com *constraint programming* comprovou a otimalidade exclusivamente de 0,42% (oito instâncias), ou seja, estas instâncias não possuíam otimalidade comprovada nas pesquisas anteriores. Estas instâncias são explicitadas na TABELA 14.

TABELA 14 – INSTANCIAS EM QUE SOLUÇÕES ÓTIMAS ENCONTRADAS NO UPMR

Instância	GRAEPR + CP			Fleszar e Hindi (2018)		
	LB	F.O.	GAP	LB	F.O.	GAP
20x4_1_JobCorre_R_uni_	292	292	0%	291	292	0,34%
20x4_5_JobCorre_R_inter_	282	282	0%	281	282	0,35%
20x6_4_JobCorre_R_inter_	178	178	0%	177	178	0,56%
20x6_4_JobCorre_R_uni_	194	194	0%	193	194	0,52%
25x6_2_JobCorre_R_uni_	186	186	0%	185	186	0,54%
25x4_4_U_10_100__R_uni_	195	195	0%	191	196	2,55%
25x6_4_JobCorre_R_inter_	215	215	0%	214	217	1,38%
50x10_3_U_100_200__R_uni_	563	563	0%	563	565	0,35%

FONTE: O autor (2020).

Na TABELA 14 é possível notar que, nas cinco primeiras instâncias, Fleszar e Hindi (2018) encontraram o valor ótimo da função objetivo (F.O.), no entanto, não conseguiram comprovar a otimalidade, mas encontraram *Lower Bounds* próximos à solução. Nas outras instâncias com 25 tarefas, o GRAEPR aprimorou tanto o valor de LB, quanto o valor da solução da função objetivo. Para a instância com 50 tarefas, Fleszar e Hindi (2018) encontraram o valor de *Lower Bound*, mas não o valor da solução.

Além da otimalidade comprovada nestas novas instâncias, foi possível aprimorar alguns valores de *Lower* e *Upper Bound*. Este aprimoramento é devidamente calculado em relação aos valores obtidos anteriormente na literatura.

$$\%LB = \frac{(LB_{new} - LB_{old})}{LB_{new}} \qquad \%UB = \frac{(UB_{old} - UB_{new})}{UB_{new}} \tag{5.2}$$

A TABELA 15 apresenta os índices de melhora e piora dos *Lower Bounds* encontrados. De acordo a TABELA 15, o método apresentado nesta pesquisa estabeleceu 11 novos valores de *Lower Bound*, com um aprimoramento médio de 0,45% em relação aos valores estabelecidos previamente. Em 1584 testes (83,37% dos testes) foi possível obter os mesmos valores já definidos. Os resultados inferiores, representam 16,05% das instâncias e tem uma diferença percentual média de -2,56%, com o pior resultado apresentando um *gap* de -34,22%.

TABELA 15 – *LOWER BOUNDS* OBTIDOS PARA O UPMR

Tarefas	#Empate	Melhores			Piores		
		#	Média	Máximo	#	Média	Mínimo
8	150	0	0,00%	0,00%	0	0,00%	0,00%
12	150	0	0,00%	0,00%	0	0,00%	0,00%
16	147	0	0,00%	0,00%	3	-13,65%	-26,13%
20	123	4	0,44%	0,56%	23	-3,76%	-19,27%
25	106	3	1,02%	2,05%	41	-6,68%	-27,69%
30	117	0	0,00%	0,00%	33	-3,90%	-12,52%
50	206	0	0,00%	0,00%	44	-2,14%	-12,65%
100	199	0	0,00%	0,00%	51	-2,73%	-34,22%
500	191	2	0,03%	0,03%	57	-0,27%	-4,37%
1000	195	2	0,03%	0,03%	53	-0,07%	-1,50%
	1584	11	0,45%	2,05%	305	-2,56%	-34,22%

FONTE: O autor (2020).

De forma análoga, a TABELA 16 exibe o índice de aprimoramento dos valores de *Upper Bounds* obtidos. A abordagem proposta estabeleceu um percentual médio de 0,32% de melhoria, em 88 soluções (4,63%) já conhecidas. No entanto, em 43,05% as soluções apresentaram resultados inferiores, com um percentual médio de -0,71% e máximo de -7,73% de diferença. Nos outros 52,58% dos testes realizados a solução encontrada foi a mesma que os resuldos obtidos previamente.

TABELA 16 – *UPPER BOUNDS* OBTIDOS PARA O UPMR

Tarefas	#Empate	Melhores			Piores		
		#	Média	Máximo	#	Média	Mínimo
8	150	0	0,00%	0,00%	0	0,00%	0,00%
12	150	0	0,00%	0,00%	0	0,00%	0,00%
16	150	0	0,00%	0,00%	0	0,00%	0,00%
20	141	2	0,59%	0,68%	7	-0,68%	-1,17%
25	130	3	0,64%	0,93%	17	-0,53%	-1,25%
30	122	3	0,66%	1,14%	25	-0,54%	-1,49%
50	128	3	0,30%	0,40%	119	-0,75%	-3,90%
100	20	3	0,21%	0,27%	227	-1,11%	-4,24%
500	1	28	0,30%	1,25%	221	-1,61%	-8,78%
1000	2	46	0,28%	1,21%	202	-1,51%	-8,38%
	994	88	0,32%	1,25%	818	-1,26%	-8,78%

FONTE: O autor (2020).

Nesta análise foi apresentado que, a abordagem, aqui proposta, se aproxima dos resultados presentes na literatura, sendo possível comprovar a otimalidade de algumas soluções que permaneciam em aberto e estabelecer novos valores de *Lower* e *Upper Bound*. Os resultados também comprovam que a utilização de modelos de programação por restrição são eficazes em conjunto com outros métodos.

5.2 RESULTADOS COMPUTACIONAIS PARA O UPMSR

Devido à inexistência de instâncias para o UPMSR na literatura, foram devidamente adaptadas as 900 instâncias apresentadas por Fanjul-Peyro, Perea e Ruiz (2017). Estas instâncias são consideradas de pequeno e médio porte, maiores detalhes da geração destes dados se encontram na Seção 5.1. A única informação ausente nestas instâncias é o tempo de *setup* entre as tarefas para cada máquina. Estes valores foram calculados através da distribuição uniforme do percentual do tempo de processamento da tarefa subsequente ao *setup* executado. A expressão (5.3) apresenta o cálculo realizado:

$$S_{ijm} = U(0.10, 0.20) * P_{jm} \tag{5.3}$$

As duas abordagens desenvolvidas neste trabalho, GRAEPR e GRAEPR+CP, também serão utilizadas para resolver o problema UPMSR. No entanto, como este é um problema diferente do tratado anteriormente, os parâmetros devem ser reconfigurados com a configuração ideal para este problema. Os parâmetros e seus intervalos utilizados permanecem os mesmos. Os valores resultantes são apresentados na TABELA 17.

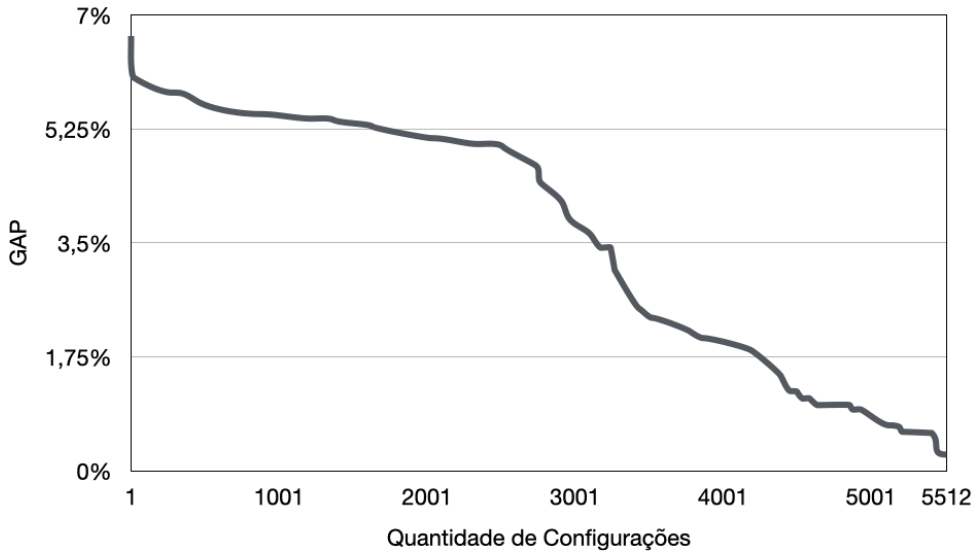
TABELA 17 – PARÂMETROS PARA O UPMSR

Parâmetro	**Intervalo**	**Valor**
Estratégia	(*forward*, *mixed*)	*mixed*
α_{GRASP}	$[0.3, 1.0]$	0.4611
α_{GRAPR}	$[0.0, 1.0]$	0.9656
E_{max}	$[10, 20]$	17
τ	$[0.3, 1.0]$	0.4865
δ	$[0.2, 0.5]$	0.2041
ψ	$[2 * E_{max}, 10 * E_{max}]$	$2 * E_{max}$

FONTE: O autor (2020).

Da mesma forma que para o UPMR, foram selecionadas 50 instâncias aleatoriamente e determinado 50000 iterações como limite. Para o UPMSR o IRACE determinou 5512 configurações, obtendo uma variação do valor máximo de 6,68% e valor mínimo de 0,25%. A evolução das soluções é apresentada na FIGURA 20.

FIGURA 20 – EVOLUÇÃO DAS SOLUÇÕES DO IRACE PARA O UPMSR



FONTE: O autor (2020)

Para este problema, foram avaliadas as performances dos 5 métodos apresentados: modelo de programação inteira mista (MILP), modelo de programação por restrição (CP), o *Greedy Randomized Adaptive Evolutionary Path Relinking* (GRAEPR), a abordagem híbrida do GRAEPR com o *constraint programming* (GRAPR + CP) e a abordagem híbrida de Fleszar e Hindi (2018) adaptada para este problema (AFH). Para cada método, o percentual do GAP é calculado utilizando a equação (5.1) e são utilizados os seus próprios valores de *Lower* e *Upper Bounds*. No entanto, para o GRAEPR, por ser um método heurístico, não é possível obter um *Lower Bound*, neste caso é utilizado o melhor *Lower Bound* encontrado por um dos outros quatro métodos.

TABELA 18 – PERCENTUAL MÉDIO DO GAP DOS RESULTADOS DO UPMSR

Tarefas	MILP	CP	GRAEPR	GRAEPR + CP	AFH
8	11,59%	**0,00%**	0,55%	**0,00%**	**0,00%**
12	50,15%	0,87%	1,52%	**0,00%**	**0,00%**
16	72,51%	27,05%	3,72%	**1,44%**	1,52%
20	90,03%	52,90%	8,71%	5,29%	**5,22%**
25	95,40%	61,99%	10,46%	7,05%	**6,24%**
30	93,79%	70,88%	11,09%	7,26%	**7,07%**
	60,06%	35,62%	6,16%	3,51%	**3,34%**

FONTE: O autor (2020).

A TABELA 18 apresenta os resultados gerais dos métodos avaliados, comprovando que os métodos exatos necessitam de tempos computacionais elevados para comprovar a otimalidade, pois obtiveram altos valores de GAP ao atingir o tempo limite. Além disso, é possível notar que o GRAEPR supera os métodos mencionados

acima, apresentando resultados competitivos, com GAP médio de 6,16%. Os métodos, GRAEPR + CP e AFH, se sobressaíram em relação aos demais, pois nestes métodos a otimalidade foi obtida nas instâncias com número de tarefas menor e igual à 12. Para as instâncias maiores, este valor aumenta de acordo com o tamanho da instância. Apesar da abordagem AFH apresentar melhores resultados, os resultados obtidos pelo GRAEPR + CP são extremamente similares, com uma diferença média de 0,17%.

TABELA 19 – TEMPO DE RESOLUÇÃO, EM SEGUNDOS, DO UPMSR

Tarefas	MILP	CP	GRAEPR	GRAEPR + CP	FLE
8	2061	7	**0**	**0**	**0**
12	3388	176	**0**	7	11
16	3587	1437	**1**	699	760
20	3578	2182	**2**	2277	2303
25	3570	2542	**3**	2760	2685
30	3578	2870	**5**	3079	3061
	3178	1536	**2**	1470	1470

FONTE: O autor (2020).

Os tempos computacionais são apresentados na TABELA 19. As abordagens que utilizam métodos exatos para resolução dos problemas apresentam tempos computacionais elevados. A abordagem GRAEPR apresentou uma velocidade de resolução extremamente superior as demais, mostrando o impacto que a velocidade de resolução teve no método GRAEPR + CP, que utiliza o GRAEPR na sua primeira fase. Ademais, pode-se concluir que, na relação entre qualidade da solução e tempo computacional, o método heurístico se sobressai, pois apresenta boas soluções com tempo computacional mínimo. Além disso, é possível ver o impacto que a inserção de uma solução inicial gera ao utilizar métodos exatos de forma híbrida, apontado na redução do tempo e na qualidade das soluções obtidas pelos métodos GRAEPR + CP e AFH.

TABELA 20 – PERCENTUAL DE SOLUÇÕES ÓTIMAS ENCONTRADAS NO UPMSR

Tarefas	Empate		GRAEPR + CP	AFH
	Sol. Ótima	Sol. Não Ótima		
8	100,00%	0,00%	0,00%	0,00%
12	100,00%	0,00%	0,00%	0,00%
16	82,67%	14,00%	2,00%	1,33%
20	36,00%	58,67%	1,33%	4,00%
25	24,00%	68,67%	0,00%	7,33%
30	16,00%	82,67%	0,00%	1,33%
	59,78%	37,33%	0,56%	2,33%

FONTE: O autor (2020).

A TABELA 20 apresenta a quantidade de soluções ótimas obtidas pelos 2 métodos com melhor performance. Ambos comprovaram otimalidade de 538 instâncias, aproximadamente 59,78% e em 336 instâncias (37,33%) não foi possível esta comprovação. Em 0,56% (cinco instâncias), somente o GRAEPR + CP obteve soluções ótimas, enquanto o AFH comprovou em 2,33% (21 instâncias). Estes números comprovam mais uma vez similaridade de ambos os métodos.

TABELA 21 – *LOWER* E *UPPER BOUNDS* OBTIDOS PARA O UPMSR

Tarefas	***Lower Bound***			***Upper Bound***		
	#Empate	**GRAEPR + CP**	**AFH**	**#Empate**	**GRAEPR + CP**	**AFH**
8	100,00%	0,00%	0,00%	100,00%	0,00%	0,00%
12	100,00%	0,00%	0,00%	100,00%	0,00%	0,00%
16	96,67%	2,00%	1,33%	98,67%	1,33%	0,00%
20	94,00%	2,00%	4,00%	77,33%	10,00%	12,67%
25	92,67%	0,00%	7,33%	64,67%	12,00%	23,33%
30	98,00%	0,67%	1,33%	46,67%	18,00%	35,33%
	96,89%	0,78%	2,33%	81,22%	6,89%	11,89%

FONTE: O autor (2020).

Uma análise em relação à quantidade de valores de *Lower* e *Upper Bound*, obtidos pelos métodos GRAEPR + CP e AFH, é apresentada na TABELA 21. Na maioria das instâncias, ambos os métodos obtiveram os mesmos valores de LB (96,89% das instâncias) e UB (81,22% das instâncias), demonstrando a eficiência semelhante de ambos os métodos. O GRAEPR + CP estabeleceu exclusivamente seis (0,78%) valores de *Lower Bounds* e 62 (6,89%) melhores soluções, enquanto o AFH estabeleceu 21 (2,33%) valores de *Lower Bound* e 107 valores de *Upper Bound*.

6 CONCLUSÃO

Neste trabalho são abordados dois problemas, o problema de máquinas paralelas não relacionadas com restrição de recursos renováveis (UPMR) e o problema de máquinas paralelas não relacionadas com *setup* dependente e restrição de recursos renováveis. Para resolução de ambos os problemas é apresentado o *Greedy Randomized Adaptive Evolutionary Path Relinking* (GRAEPR), que provou ser eficiente em obter soluções com pequenos valores de tempo computacional. Também para ambos os problemas foi apresentado uma abordagem híbrida utilizando o GRAEPR e um modelo de programação por restrição.

Para o UPMR, problema diversamente abordado na literatura, foram realizadas avaliações de performance em comparação com os melhores resultados já apresentados. Os resultados obtidos são similares aos estabelecidos anteriormente, e foi possível realizar uma contribuição ao obter algumas soluções ótimas, não provadas anteriormente, e estabelecer novos valores de *Lower* e *Upper Bound*. Com isto, foi possível evidenciar a eficiência e o potencial do método apresentado.

Buscando expandir o problema abordado, as características de *setup* dependente foram inseridas no UPMR, definindo assim uma nova variante deste problema, o UPMSR. Devido a falta de pesquisas relacionadas a este problema, as instâncias já presentes na literatura foram adaptadas e também foram desenvolvidos dois novos modelos, um modelo de programação inteira mista e um modelo de programação por restrição. Além disso, devido a sua alta performance, a abordagem apresentada por Fleszar e Hindi (2018) foi adaptada para este problema. As 5 abordagens de resolução foram avaliadas e mais uma vez os resultados obtidos pelor GRAEPR + CP e pela abordagem de Fleszar e Hindi (2018) apresentaram resultados ótimos e similares. Além disso, foi possível demonstrar a eficiência do GRAEPR em pequenos tempos computacionais, evidenciando o potencial ainda a ser explorado deste método.

Este trabalho pode ser motivador para as seguintes pesquisas futuras: *i*) Explorar diferentes versões do GRAEPR, avaliando outras estratégias de trajetória, diferentes abordagens de construção do conjuntos elite e a utilização do *Evolutionary Path-relinking* durante o processo e não apenas na fase de pós-otimização; *ii*) Definir mais *Lower Bounds* robustos para fase de pré-otimização das abordagens híbridas; *iii*) Expandir o problema para utilização dos recursos durante o *setup* e o tempo de processamento das tarefas; *iv*) testar combinações diferentes de parâmetros para o solver CPLEX, dado que neste trabalho optou-se apenas por parâmetros *default*; *v*) utilizar estas abordagens para resolver problemas baseados em casos reais.

REFERÊNCIAS

AFZALIRAD, M.; REZAEIAN, J. Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. **Computers & Industrial Engineering**, v. 98, p. 40–52, 2016.

AFZALIRAD, M.; SHAFIPOUR, M. Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. **Journal of Intelligent Manufacturing**, v. 29, p. 423–437, 2018.

AKBAR, M.; IROHARA, T. Dual resource constrained scheduling considering operator working modes and moving in identical parallel machines using a permutation-based genetic algorithm. **Advances in Production Management Systems. Production Management for Data-Driven, Intelligent, Collaborative, and Sustainable Manufacturing. IFIP Advances in Information and Communication Technology**, v. 535, p. 464–472, 2018.

ARBAOUI, T.; YALAOUI, F. Solving the Unrelated Parallel Machine Scheduling Problem with Additional Resources Using Constraint Programming. **Intelligent Information and Database Systems**, p. 716–725, 2018.

BELKAID, F.; SARI, Z.; YALAOUI, F. A hybrid genetic algorithm for parallel machine scheduling problem with consumable resources. **International Journal of applied Metaheuristic Computing**, v. 4, p. 143–148, 2013.

BINATO, S.; FARIA JR., H.; RESENDE, M. G. C. Greedy randomized adaptive path relinking. In: edição: J.P. Sousa. [S.l.: s.n.], 2001. P. 393–398.

BŁAŻEWICZ, J.; BRAUNER, N.; FINKE, G. Scheduling with discrete resource constraints. In: **Handbook of Scheduling: Algorithms, Models, and Performance Analysis**. Edição: J. Y. -T. Leung. Boston, MA: CRC Press, USA, 2004. v. 3 cap. 23, p. 1493–1641. ISBN 978-1-4613-0303-9.

BŁAŻEWICZ, J.; ECKER, K. H. et al. From Theory to Applications. In: **Handbook on Scheduling**. Edição: J. Błażewicz, J. Węglarz e E. Pesch. Berlin, Heidelberg: Springer, 2007. cap. 12. ISBN 978-3-540-28046-0.

CHEN; POTTS, C. N.; WOEGINGER, G. J. A Review of Machine Scheduling: Complexity, Algorithms and Approximability. In: **Handbook of Combinatorial Optimization**. Edição: D. Du e P. M. Pardalos. Boston, MA: Springer US, 1998. v. 3, p. 1493–1641. ISBN 978-1-4613-0303-9.

CHENG, T. C. E.; DING, Q. The time dependent machine makespan problema is strongly NP-complete. **Computers & Operations Research**, v. 26, p. 749–754, 1999.

CHENG, T. C. E.; SIN, C. C. S. A state-of-the-art review of parallel-machine scheduling research. **European Journal of Operational Research**, v. 47, p. 147–153, 1990.

DANIELS, R. L.; HOOPES, B. J.; MAZZOLA, J. B. Scheduling parallels manufacturing cells with resource flexibility. **Management Science**, v. 42, n. 9, p. 1260–1276, 1996.

DANIELS, R. L.; HUA, S. Y.; WEBSTER, S. Heuristics for parallel-machine flexible-resource scheduling problems with unspecified job assignment. **Computers & Operations Research**, v. 26, n. 2, p. 143–155, 1999.

DAVIS, E.; JEFFREY, J. M. Algorithms for Scheduling Tasks on Unrelated Processors. **Journal Association for Computing Machinery**, v. 28, n. 4, p. 721–736, 1981.

DE, P.; MORTON, T. E. Scheduling to minimize makespan on unequal parallel processors. **Decision Sciences**, v. 11, n. 4, p. 586–602, 1980.

EDIS, E. B.; ARAZ, C.; OZKARAHAN, I. Lagrangian-based solution approaches for a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. **New Frontiers in Applied Artificial Intelligence**, p. 337–346, 2008.

EDIS, E. B.; OGUZ, C. Parallel machine scheduling with additional resources: A lagrangian-based constraint programming approach. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 6697, p. 92–98, 2011.

EDIS, E. B.; OGUZ, C. Parallel machine scheduling with flexible resources. **Computers & Industrial Engineering**, v. 63, n. 2, p. 433–447, 2012.

EDIS, E. B.; OGUZ, C.; OZKARAHAN, I. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. **European Journal of Operational Research**, v. 20, n. 3, p. 449–463, 2013.

EDIS, E. B.; OZKARAHAN, I. A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. **Engineering Optimization**, v. 43, n. 2, p. 135–157, 2011.

EDIS, E. B.; OZKARAHAN, I. Solution approaches for a real-life resource-constrained parallel machine scheduling problem. **International Journal of Advanced Manufacturing Technology**, v. 58, p. 1141–1153, 2012.

FANJUL-PEYRO, L.; PEREA, F.; RUIZ, R. Algorithms for the unspecified unrelated parallel machine scheduling problem with additional resources. **2015 International Conference on Industrial Engineering and Systems Management (IESM)**, p. 69–73, 2015.

FANJUL-PEYRO, L.; PEREA, F.; RUIZ, R. Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. **European Journal of Operational Research**, v. 260, n. 2, p. 482–493, 2017.

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

FEO, T.; SARATHY, K.; MCGAHAN, J. A grasp for single machine scheduling with sequence dependent setup costs and linear delay penalties. **Computers & Operations Research**, v. 23, n. 9, p. 881–895, 1996.

FEO, T.; VENKATRAMAN, K.; BARD, J. A GRASP for a difficult single machine scheduling problem. **Computers & Operations Research**, v. 18, n. 8, p. 635–643, 1991.

FESTA, P.; RESENDE, M. G. C. Grasp: An Annotated Bibliography. In: **Essays and Surveys in Metaheuristics. Operations Research/Computer Science Interfaces Series**. Edição: C. C. Ribeiro e P. Hansen. [S.l.]: Springer, 2002. v. 15, p. 325–367. ISBN 978-1-4615-1507-4.

FLESZAR, K.; HINDI, K. S. Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. **European Journal of Operational Research**, v. 271, n. 3, p. 839–848, 2018.

FU, Y. et al. Parallel machine scheduling with dynamic resource allocation via a master–slave genetic algorithm. **IEEJ Transactions on Electrical and Electronic Engineering**, v. 13, n. 5, p. 748–756, 2018.

GEDIK, R. et al. A constraint programming approach for the team orienteering problem with time windows. **Computers & Industrial Engineering**, v. 107, p. 178–195, 2017.

GLASS, C. A.; POTTS, C. N.; SHADE, P. Unrelated parallel machine scheduling using local search. **Mathematical and Computer Modelling**, v. 20, n. 2, p. 41–52, 1994.

GLOVER, F. Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges. In: **Interfaces in Computer Science and Operations Research. Operations Research/Computer Science Interfaces Series**. Edição: R. S. Barr, R. V. Helgason e J. L. Kennington. Boston, MA: Springer, 1997. v. 7, p. 1–75. ISBN 978-1-4613-6837-3.

GRAHAM, R. L. et al. Optimization and approximation in deterministic sequencing and scheduling: a survey. **Annals of Discrete Mathematics**, v. 5, n. 3, p. 287–326, 1979.

GRIGORIEV, A.; SVIRIDENKO, M.; UETZ, M. Unrelated parallel machine scheduling with resource dependent processing times. **Integer Programming and Combinatorial Optimization**, p. 182–195, 2005.

GUMPENBERGER, M. W. C.; GORRAIZ, J. Bibliometric practices and activities at the University of Vienna. **Library Management**, v. 33, n. 3, p. 174–183, 2012.

HARIRI, A. M. A.; POTTS, C. N. Heuristics for scheduling unrelated parallel machines. **Computers & Operations Research**, v. 18, n. 3, p. 323–331, 1991.

HERNÁNDEZ-PÉREZ, H.; RODRÍGUEZ-MARTÍN, I.; SALAZAR-GONZÁLEZ, J. J. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. **Computers & Operations Research**, v. 36, n. 5, p. 1639–1645, 2009.

HOROWITZ, E.; SAHNI, S. Exact and Approximate Algorithms for Scheduling Nonidentical Processors. **Journal Association for Computing Machinery**, v. 23, n. 2, p. 317–327, 1976.

IBM. **IBM ILOG CPLEX Optimization Studio CP Optimizer User's Manual**. 2017. Disponível em: <https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcpoptimizer.pdf>. (Acessado em: 01/01/2020).

KELLERER, H. An approximation algorithm for identical parallel machine scheduling with resource dependent processing times. **Operations Research Letters**, v. 36, n. 2, p. 157–159, 2008.

LENSTRA, J. K.; RINNOOY KAN, A. H. G.; BRUCKER, P. Complexity of machine scheduing problems. **Annals of Operations Research**, v. 1, p. 343–362, 1977.

LENSTRA, J. K.; SHMOYS, D. B.; TARDOS, É. Approximation algorithms for scheduling unrelated parallel machines. **Mathematical Programming**, v. 46, n. 1, p. 259–271, 1990.

LÓPEZ-IBÁNEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016.

MARTELLO, S.; SOUMIS, F.; TOTH, P. Exact and approximation algorithms for makespan minimization on unrelated parallel machines. **Discrete Applied Mathematics**, v. 75, n. 2, p. 169–188, 1997.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, v. 24, n. 11, p. 1097–1100, 1997.

MUSIER, R. F. H.; EVANS, L. B. An approximate method for the production scheduling of industrial batch processes with parallel units. **Computers & Chemical Engineering**, v. 13, n. 1, p. 229–238, 1989.

PFUND, M.; FOWLER, J. W.; GUPTA, J. N. D. A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. **Journal of the Chinese Institute of Industrial Engineers**, v. 21, n. 3, p. 230–241, 2004.

POTTS, C. N. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. **Discrete Applied Mathematics**, v. 10, n. 2, p. 155–164, 1985.

POWELL, W. B.; CHEN. Solving Parallel Machine Scheduling Problems by Column Generation. **Informs Journal on Computing**, v. 11, July 1996, p. 78–94, 1995.

QIONG, Z. et al. An ant colony optimization model for parallel machine scheduling with human resource constraints. **Advances in Intelligent and Soft Computing**, v. 66, p. 917–926, 2010.

RESENDE, M. G. C.; BINATO, S. et al. A GRASP for job shop scheduling. In: **Essays and Surveys in Metaheuristics. Operations Research/Computer Science Interfaces Series**. Edição: C. C. Ribeiro e P. Hansen. [S.l.]: Springer, 2002. v. 15, p. 59–80. ISBN 978-1-4615-1507-4.

RESENDE, M. G. C.; RIBEIRO, C. C. GRASP with Path-Relinking: Recent Advances and Applications. In: **Metaheuristics: Progress as Real Problem Solvers. Operations Research/Computer Science Interfaces Series**. Edição: T. Ibaraki, K. Nonobe e M. Yagiura. Boston, MA: Springer, 2005. v. 32, p. 29–63. ISBN 978-0-387-25382-4.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy Randomized Adaptive Search Procedures: Advances and Extensions. In: **Handbook of Metaheuristics. International Series in Operations Research & Management Science**. Edição: M. Gendreau e JY. Potvin. Cham: Springer, 2005. v. 272, p. 169–220. ISBN 978-3-319-91085-7.

RESENDE, M. G. C.; RIBEIRO, C. C. **Optimization by GRASP**. [S.l.]: Springer-Verlag New York, 2016. ISBN 978-1-4939-6530-4.

RESENDE, M. G. C.; WERNECK, R. F. A Hybrid Heuristic for the p-Median Problem. **Journal of Heuristics**, v. 10, n. 1, p. 59–88, 2004.

SUN, H. F.; LIU, X. D.; HOU, W. Research of parallel machine scheduling with flexible resources based on nested partition method. **Applied Mechanics and Materials**, v. 459, p. 488–493, 2013.

TANDON, M.; CUMMINGS, P. T.; LE VAN, M. D. Scheduling of multiple products on parallel units with tardiness penalties using simulated annealing. **Computers & Chemical Engineering**, v. 19, n. 10, p. 1069–1076, 1995.

VALLADA, E.; VILLA, F.; FANJUL-PEYRO, L. Enriched metaheuristics for the resource constrained unrelated parallel machine scheduling problem. **Computers & Operations Research**, v. 111, p. 415–424, 2019.

VENTURA, J. A.; KIM, D. Parallel machine scheduling about an unrestricted due date and additional resource constraints. **IIE Transactions (Institute of Industrial Engineers)**, v. 32, n. 2, p. 147–153, 2000.

VENTURA, J. A.; KIM, D. Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints. **Computers & Operations Research**, v. 30, n. 13, p. 1945–1958, 2003.

VILLA, F.; VALLADA, E.; FANJUL-PEYRO, L. Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. **Expert Systems with Applications**, v. 93, n. 1, p. 28–38, 2018.

YEPES-BORRERO, J. C. et al. GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. **Expert Systems With Applications**, v. 141, n. 7, p. 112959, 2020.

ZHENG, X. -L.; WANG, L. A Collaborative Multiobjective Fruit Fly Optimization Algorithm for the Resource Constrained Unrelated Parallel Machine Green Scheduling Problem. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 48, n. 5, p. 790–800, 2018.

ZHENG, X. -L.; WANG, L. A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. **Expert Systems with Applications**, v. 65, p. 28–39, 2016.