UNIVERSIDADE NOVE DE JULHO-UNINOVE

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA E GESTÃO DO CONHECIMENTO

FERNANDO ANDRE ZEMUNER GARCIA

MODELAGEM E OTIMIZAÇÃO MULTIOBJETIVO DO PROBLEMA INTEGRADO DE ESCALONAMENTO DE TAREFAS E ALOCAÇÃO DE RECURSOS COM CURVA DE APRENDIZADO EM MÚLTIPLOS PROJETOS

São Paulo

2022

FERNANDO ANDRE ZEMUNER GARCIA

MODELAGEM E OTIMIZAÇÃO MULTIOBJETIVO DO PROBLEMA INTEGRADO DE ESCALONAMENTO DE TAREFAS E ALOCAÇÃO DE RECURSOS COM CURVA DE APRENDIZADO EM MÚLTIPLOS PROJETOS

Tese apresentada ao Programa de Pós-Graduação em Informática e Gestão do Conhecimento da Universidade Nove de Julho – UNINOVE como requisito parcial para a obtenção do título de Doutor em Informática e Gestão do Conhecimento.

Prof. Fabio Henrique Pereira, Dr. – Orientador

São Paulo

2022

Garcia, Fernando Andre Zemuner.
Modelagem e otimização multiobjetivo do problema integrado de escalonamento de tarefas e alocação de recursos com curva de aprendizado em múltiplos projetos. / Fernando Andre Zemuner Garcia. 2022.
163 f.
Tese (Doutorado) - Universidade Nove de Julho - UNINOVE, São Paulo, 2022.

Orientador (a): Prof. Dr. Fabio Henrique Pereira.
1. Algoritmo de colônia de formigas. 2. Múltiplas habilidades. 3. Múltiplos projetos. 4. Curva de aprendizado. 5. Esquecimento.
I. Pereira, Fabio Henrique. II. Título.

CDU 004

UNINOVE
Universidade Nove de Julho

PARECER – EXAME DE DEFESA

Parecer da Comissão Examinadora designada para o exame de defesa do Programa de Pós-Graduação em Informática e Gestão do Conhecimento, a qual se submeteu o aluno regularmente matriculado **Fernando André Zémuner Garcia.**

Tendo examinado o trabalho apresentado para obtenção do título de "Doutor em Informática e Gestão do Conhecimento", com Tese intitulada **"**Modelagem e otimização multiobjetivo do problema integrado de escalonamento de tarefas e alocação de recursos com curva de aprendizado em múltiplos projetos", a Comissão Examinadora considerou o trabalho:

(X) Aprovado **( ) Aprovado condicionalmente**
( ) Reprovado com direito a novo exame **( ) Reprovado**

Parecer:
Aprovado

EXAMINADORES

Prof(a). Dr(a). Fabio Henrique Pereira – UNINOVE (Orientador)

Prof(a). Dr(a). Gilberto Francisco Martha de Souza – USP (Membro Externo)

Prof(a). Dr(a). Marcio Cardoso Machado – UNIP (Membro Externo)

Prof(a). Dr(a). Alessandro Melo de Ana – PPGI/UNINOVE (Membro Interno)

Prof(a). Dr(a). Cleber Gustavo Dias – PPGI/UNINOVE (Membro Interno)

São paulo, 10 de março de 2022.

DEDICATÓRIA

A minha família

Helita Varalda de Souza Zemuner

Nathália Varalda de Souza Zemuner Garcia

Nickolas Varalda de Souza Zemuner Garcia

AGRADECIMENTOS

O aprendizado foi difícil, porém com muitas recompensas e pessoas que me apoiaram nesta caminhada para poder concluir mais esta etapa em minha vida:

Agradeço a minha **família** pela compreensão, carinho, incentivo e suporte neste processo árduo, com muitos períodos de afastamento, porém sempre próximos em pensamento.

Agradeço aos meus **pais e sogros** por sempre me incentivarem em todos os momentos, bem como suporte recebido e para com a minha família.

Agradeço as amizades conquistadas e as que mantive neste tempo. **Pessoas** que suportaram e ajudaram nesta caminhada.

Agradeço ao professor **Fabio Henrique Pereira** pela sua paciência, orientação e direcionamento durante este período de doutorado. Estes atributos possibilitaram um grande aprendizado não apenas acadêmico, mas também pessoal.

Agradeço aos professores do curso aos quais pude absorver um pouco de seu conhecimento e experiência, que serão muito importantes em meu futuro.

Agradeço aos professores **Alessandro Melo Deana, Cleber Gustavo Dias, Marcio Cardoso Machado** e **Gilberto Francisco Martha de Souza** pelo tempo e esforço dispendido para contribuir com a evolução e avaliação desta tese.

RESUMO

Em um cenário de transformação digital acelerada, um rígido controle dos processos produtivos é cada vez mais necessário. Nesse contexto, emerge o conceito de projeto caracterizado como um conjunto de tarefas temporárias para criação de um produto, serviço ou resultado único e exclusivo. Por envolver a execução de tarefas previamente determinadas e com recursos limitados, o desenvolvimento de projetos requer ações de planejamento e controle, especialmente no que se refere à otimização dos recursos empresariais e a produtividade. Além de alocar os recursos mais adequados às tarefas nos diferentes projetos, tais ações envolvem a definição de um escalonamento das tarefas designadas a cada um dos recursos alocados, o que consiste em determinar uma sequência para a execução das tarefas. Adicionalmente, os recursos podem possuir habilidades diversas que não apenas interferem na sua alocação como também no tempo para a execução de cada tarefa. Apesar da grande influência do escalonamento para a qualidade da alocação, os problemas são geralmente resolvidos de forma independente. Assim, esta tese propõe um novo algoritmo de otimização integrado com um modelo de simulação para solução unificada dos problemas de escalonamento de tarefas e alocação de recursos em múltiplos projetos, considerando a dependência entre tarefas e recursos com múltiplas habilidades e curva de aprendizado. O algoritmo proposto foi avaliado e posteriormente comparado com a literatura correlata em relação a alocação de recursos em múltiplos projetos, alocação de recursos com múltiplas habilidades em projeto, alocação de múltiplos recursos, com múltiplas habilidades, com ganho e perda destas habilidades em múltiplos projetos de forma unificada. O algoritmo se mostrou eficaz, pois consegue atender de forma unificada a todos os problemas avaliados podendo aumentar ou diminuir a complexidade da simulação conforme os parâmetros utilizados. Porém, o método proposto necessita de evoluções pois apresentou alto tempo de execução do algoritmo em determinados cenários simulados, em comparação com outros algoritmos avaliados.

Palavras-chave: Algoritmo de colônia de formigas, múltiplas habilidades, múltiplos projetos, curva de aprendizado e esquecimento.

ABSTRACT

In a scenario of accelerated digital transformation, strict control of production processes is increasingly necessary. In this context, emerges the concept of project characterized as a set of temporary tasks for creating a single and unique product, service, or result. As it involves the execution of previously determined tasks with limited resources, project development requires planning and control actions, especially with regard to the optimization of business resources and productivity. In addition to allocating the most appropriate resources to the tasks in the different projects, such actions involve defining a schedule of tasks assigned to each of the allocated resources, which consists of determining a sequence for the execution of the tasks. Additionally, resources can have different skills that not only interfere in their allocation but also in the time required to perform each task. Despite the great influence of scheduling for the quality of allocation, problems are usually solved independently. Thus, this thesis proposes a new optimization algorithm integrated with a simulation model for a unified solution of task scheduling and resource allocation problems in multiple projects, considering the dependency between tasks and resources with multiple skills and learning curve. The proposed algorithm was evaluated and then compared with the correlated literature in relation to resource allocation in multiple projects, allocation of resources with multiple project skills, allocation of multiple resources, with multiple skills, with gain and loss of these skills in multiple projects in a unified way. The algorithm proved to be effective, because it can respond in a unified way to all the problems evaluated and may increase or decrease the complexity of the simulation according to the parameters used. However, However, the proposed.method requires evolution because it presented high execution time of the algorithm in certain simulated scenarios, compared to other evaluated algorithms.

Keywords: Ant colony algorithm, multiple skills, multiple projects, learn and forget curve.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE APÊNDICES

LISTA DE ABREVIATURAS E SIGLAS

ACA	*Ant Colony Algorithm*
ACO	*Ant Colony Optimization*
ACO-BF	*Ant Colony Optimization Based on Best Fit*
ACODR	*Ant Colony Optimization Based on A Dominance Ranking*
ACO-DRC	*Based on a Dominance Ranking Procedure and Crowding Distance Comparison*
AG	Algoritmo Genético
AMWLC	*Approximate Modified Wright Learning Curve*
AS	*Simulated Annealing*
BF	*Backward-Forward*
BFHGA	*Backward-Forward Hybrid Genetic Algorithm*
CFRP	*Conflict-Free Routing Problem*
CPU	*Computational Processing Units*
DEM	*Differential Evolution Metaheuristic*
Disc	Distribuição de Probabilidade Discreta
DOMVO	*Discrete Oppositional Multi-Verse Optimization*
EBS	*Event-Based Scheduler*
FAP	*Frequency Assignment Problem*
FJSSP	*Flexible Job Shop Scheduling Problem*
GALib	*Genetic Algorithm Library*
GAP	*Generalized Assignment Problem*
GP-HH	*Genetic Programming Hyper-Heuristic*
GPU	*Graphical Processing Units*
GVNS	*General Variable Neighborhood Search Approach*
ICMPACO	*Multi-Population Co-Evolution Ant Colony Optimization*
iMOPSE	*Intelligent Multi Objective Project Scheduling Environment*
JSON	*JavaScript Object Notation*
JSP	*Job Shop Problem*
KBACO	*Knowledge-Based Ant Colony Optimization*
KBFOA	*Knowledge-Based Fruit Fly Optimization Algorithm*
LFCM	*Learn–Forget Curve Model*
LFS	*Learn And Forget Skill*
MACO	*Multi Ant Colony Optimization*
MDVRP	*Multi-Depot Vehicle Routing Problem*
MHACO	*Multi-Objective Hybrid Ant Colony Optimization*
MILP	*Mixed-Integer Linear Programming*
MINLP	*Mixed-Integer Nonlinear Programming*
MIN-SLK	*Minimum Activity Total Slack*
MMGAs	*Multi-Modal Genetic Algorithms*
MOACO	*Multi-Objective Ant Colony Optimization*
MOFA	*Multi-Objective Fibonacci-Based Algorithm*
MOGP-HH / D	*Decomposition-Based Multi-Objective Genetic Programming Hyper-Heuristic*
MPSSSL	*Multiple Project Selection, Scheduling and Staffing with Learning*
MSRCPSP	*Multiskill Resource-Constrained Project Scheduling Problem*
MS-RCPSP	*Multiskill Resource-Constrained Project Scheduling Problem*

MWLC	*Modified Wright Learning Curve*
NSGA-II	*Non-Dominated Sorting Genetic Algorithm II*
NTGA	*Non-Dominated Tournament Genetic Algorithm*
P-ACO	*Pareto Ant Colony Optimization*
P-GWO	*Pareto-Based Greywolf Optimizer*
PR	*Priority Rules*
P-SGS	*Parallel Schedule Generation Scheme*
PSO	*Particle Swarm Optimization*
PVRP	*Period Vehicle Routing Problem*
QAP	*Quadratic Assignment Problem*
RCMPSP	*Resource Constrained Multi-Project Scheduling Problem*
RCMPSPLIB	*Resource Constrained Multi-Project Scheduling Problem Library*
RCPSP	*Resource Constrained Project Scheduling Problem*
Real-RCPSP	*Real-Resource-Constrained Project Scheduling Problem*
SaGA	*Sampling GA*
SMT	*Satisfiability Modulo Theories*
SVRP	*Stochastic Vehicle Routing Problem*
TLBO	*Teaching–Learning-Based Optimization*
VLGA	*Variable-Length*
VRIF	*Variable Regression to Invariant Forgetting*
VRVF	*Variable Regression to Variable Forgetting*
WoS	*Web of Science*

SUMÁRIO

1 Sumário

1. Introdução

Em um cenário de transformação digital acelerada, um rígido controle dos processos produtivos é cada vez mais necessário. Nesse contexto, emerge o conceito de projeto caracterizado como um conjunto de tarefas temporárias para criação de um produto, serviço ou resultado único e exclusivo (PMI, 2018). A utilização dos projetos faz com que as empresas possam realizar suas entregas de forma mais rápida e eficiente, economizando recursos, sejam eles humanos, materiais ou econômicos (ÁLVAREZ; PATIÑO, 2015).

Kannimuthu *et al.* (2018) identificaram que nas empresas que adotam os projetos para suas entregas, é comum e existência de múltiplos projetos sendo executados de forma sequencial ou em paralelo. Os autores Chen e Zhang (2013) afirman que cada vez mais, os projetos são executados ao mesmo tempo, e que, com esta concorrência de projetos, os recursos precisam ser alocados para que todas as tarefas possam ser executadas, seja de forma sequencial, ou com dependência entre elas, para que se possa concluir os projetos no menor tempo possível e com o menor custo.

Nembhard (2000) acrescenta que as habilidades dos recursos devem ser avaliadas, pois os recursos podem ter múltiplas habilidades que influenciam em sua alocação nos projetos, e que a alocação pode promover a aquisição de novas habilidades com o aprendizado. Por outro lado, as habilidades também podem retroceder caso os recursos deixem de utilizá-la ou de desenvolvê-las. Nembhard (2000) salienta ainda que essa mudança no nível de habilidade dos recursos pode interferir no tempo que esse recurso gasta para executar as tarefas de um projeto, impactando tanto no prazo quanto no custo do projeto. Entretanto, esse impacto é pouco abordado na literatura e não é completamente conhecido e avaliado.

O problema de alocação de recursos em tarefas nos projetos é uma tema que se iniciou nos anos 50 como exemplo o estudo de Wagner (1959) para a alocação dos recursos em um projeto, devido sua grande complexidade computacional a qual cresce exponencialmente com o aumento no número de tarefas a serem desempenhadas e da quantidade de recursos a serem alocados (BEN ISSA; PATTERSON; TU, 2021). Trata-se, portanto, de um problema de difícil resolução e com muitas soluções possíveis não sendo uma atividade trivial (VILLAFÁÑEZ *et al.*,

2020), para o qual o tratamento de forma manual é, em geral, inviável tanto em termos de tempo quanto em qualidade da solução (JERIĆ; FIGUEIRA, 2012) e encontrar a melhor solução com os algoritmos clássicos propostos são inviáveis com os sistemas computacionais mais recentes (VILLAFÁÑEZ *et al.*, 2019a).

Em um contexto de múltiplos projetos, mesmo após a alocação dos recursos ainda existe o problema de estabelecer uma ordem para a execução das tarefas nos diferentes projetos. Em geral, existe uma ordem tecnológica definida para as tarefas de um mesmo projeto e que não pode ser modificada (KUHPFAHL; BIERWIRTH, 2016). Entretanto, se um recurso é alocado em mais de um projeto simultâneo, pode-se estabelecer uma sequência que o recurso deverá seguir para atender as tarefas entre os diferentes projetos. Trata-se do problema de escalonamento ou agendamento de tarefas (do inglês, *scheduling*) que possui grande importância acadêmica e industrial (YAZDANI *et al.*, 2017).

Uma questão importante é que o escalonamento das tarefas interfere na qualidade da solução de alocação, e vice-versa, de forma que uma solução inicialmente boa para a alocação dos recursos pode se tornar não tão boa quando o escalonamento é definido. Assim, idealmente os dois problemas devem ser resolvidos simultaneamente aumentando a complexidade computacional em relação aos problemas originais de escalonar as tarefas e alocar os recursos de forma independente (ATTIA; DUQUENNE; LE-LANN, 2014).

A literatura mostra a existência de algumas variações do problema de escalonamento de tarefas em projetos em suas diversas aplicações em indústrias, faculdades, empresas em geral e outros setores (ATTIA; DUQUENNE; LE-LANN, 2014). Destaca-se, no entanto, que apesar do grande número de abordagens existentes predomina o tratamento determinístico de versões independentes dos dois problemas. Nesse contexto, a aplicação de métodos exatos para a obtenção da solução ótima, pode não ser viável em função do elevado tempo de processamento, conforme destacam Fernandez-Viagas e Framinan (2014), por exemplo. A pesquisa bibliográfica realizada, conforme apresentada no capítulo 2, mostra que há um número reduzido de artigos que buscaram uma solução integrada para a alocação de recursos com múltiplas habilidades em projeto e o escalonamento de tarefas em múltiplos projetos, em especial se for levada em conta a aleatoriedade características em ambientes reais de produção.

Destaca-se que a busca de soluções viáveis para o problema integrado é fundamental em cenários de recursos humanos especialistas com múltiplas habilidades que atuam em múltiplos projetos simultâneos. Como exemplo, pode-se citar SHU *et al.* (2018) que estudaram o atendimento hospitalar no qual os recursos atuam simultaneamente em diversas especialidades, com escalas de plantões (alocações) e agendamento das tarefas com o objetivo de minimizar tempos de atendimento e custos.

Outro exemplo, é a alocação de recursos de TI que trabalham com diversas linguagens de programação em projetos de desenvolvimento de software. Os projetos são solicitados e os gestores de TI precisam estimar o custo e o prazo para completar todos os projetos utilizando os recursos disponíveis. Antes do início, é solicitado o tempo e o custo, para verificar se o produto a ser entregue será economicamente viável e se será atendido dentro do prazo necessário para o lançamento (BEN ISSA; PATTERSON; TU, 2021).

Por fim, vale destacar mais uma característica associada aos problemas de alocação de recursos e escalonamento de tarefas em projetos, especialmente em cenários reais. Em geral tratados de forma determinística, os tempos de processamento das tarefas pode apresentar características estocásticas com grandes variações, como é o caso, por exemplo, dos atendimentos hospitalares citados anteriormente (SHU *et al.*, 2018). Ademais, os tempos de processamento podem ser fortemente dependentes do nível de habilidade do recurso alocado. Assim, conforme passa o tempo e o recurso se dedica a aprender, ou passa a utilizar a habilidade para a qual foi alocado, o tempo de execução da tarefa tende a ficar menor. De forma inversa, conforme o tempo passa e o recurso deixa de utilizar a habilidade que possui, o tempo de execução de determinada tarefa que depende da habilidade, tende a ficar maior. Este estudo é conhecido como curva de aprendizado ou modelagem da curva de aprendizado e esquecimento (ATTIA; DUQUENNE; LE-LANN, 2014; HOEDT *et al.*, 2020).

Ao considerar às características de aprender e esquecer com curvas estocásticas para o tempo de execução das tarefas o problema integrado resultante torna-se um dos mais difíceis da classe de complexidade NP-completo, para o qual as heurísticas tradicionais não são capazes de encontrar solução em tempo hábil para as empresas. Assim, optou-se neste trabalho pela utilização de metaheurísticas para

tentar encontrar a solução ótima, ou uma subótima. Assim, o objetivo geral e os objetivos específicos foram definidos e serão apresentados na próxima seção.

1.1 Objetivos geral e específicos

De acordo com a proposta desta tese são apresentados a seguir o objetivo geral e os objetivos específicos.

1.1.1 Objetivo geral

Propor um novo algoritmo de otimização integrado com um modelo de simulação para solução unificada dos problemas de escalonamento de tarefas e alocação de recursos em múltiplos projetos, considerando a dependência entre tarefas e recursos com múltiplas habilidades e curva de aprendizado.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

a) Desenvolver um modelo de simulação e um algoritmo de otimização para a alocação de recursos e escalonamento de tarefas de forma integrada;
b) Propor uma versão multiobjetivo da metaheurística colônia de formigas para solução do problema integrado.

1.2 Justificativa da pesquisa

Apesar dos recentes avanços tecnológicos, as empresas ainda possuem dificuldades para otimização e redução de custos nas alocações dos recursos nos projetos, devido ao fato de ser um problema complexo, que demanda muito esforço humano ou computacional (HOFFMANN; KELLENBRINK; HELBER, 2020).

O tempo total do projeto afeta diretamente na entrega do produto, podendo impactar na estratégia competitiva da empresa que solicita o desenvolvimento (YASSINE *et al.*, 2017). A solução ideal, na maioria dos casos, é a que possui o menor tempo para finalizar todo o projeto. Porém, somente medir o tempo, sem considerar outros fatores, pode gerar custos mais altos para a conclusão do projeto. Assim, os pesquisadores buscam uma combinação de objetivos para a solução ideal, como avaliar o tempo mais o custo total do projeto como uma alternativa viável para maximizar o sucesso na entrega do produto final do projeto para o cliente (CHEN *et al.*, 2017; YASSINE *et al.*, 2017; BARTH; KOCH, 2019)..

Merece justificativa também o uso da modelagem e simulação em substituição aos algoritmos matemáticos frequentemente adotados na literatura. A modelagem e simulação, além de possibilitar uma visualização e análise temporal do funcionamento do sistema em estudo, permite um controle dos parâmetros do ambiente e repetição dos experimentos sob diferentes restrições (HUSSEIN; MOUSA; ALQARNI, 2019), como por exemplo o tratamento estocástico dos tempos de processamento das tarefas em função do grau de habilidade do recurso alocado. Ressalta-se que os poucos trabalhos que consideram essa aleatoriedade dos tempos de processamento das tarefas o fazem com base em números *Fuzzy*, que consideram valores mais altos, mais baixos e médios para modelar a aleatoriedade e tentar obter uma solução aceitável (HEMATIAN *et al.*, 2020). Os tempos de processamento Fuzzy podem não representar adequadamente os tempos de execução de cada tarefa em cenários que possuem grandes variações.

Tanto o desenvolvimento quanto a atenuação da habilidade podem apresentar impactos nos prazos e custos dos projetos (CHEN *et al.*, 2017), justificando-se assim o uso da curva de aprendizado e esquecimento.

Por fim, destaca-se que, dada a complexidade dos problemas, a utilização de métodos exatos nem sempre é computacionalmente viável, o que faz com as metaheurísticas sejam uma escolha natural por possibilitar soluções de boas qualidades em um tempo computacional aceitável em aplicações reais (Chen *et al.*, 2017). Ao pesquisar na literatura, diversas metaheurísticas são utilizadas para resolução do problema de escalonamento de tarefas e alocação de recursos, como Algoritmo Genético, Colônia de Formigas e outros. Ao aprofundar as pesquisas, foi identificado que o algoritmo de Colônia de Formigas continua sendo alterado para atender a novos problemas, como a de alocação de recursos que executam na nuvem, que caso fosse abstraída a solução adotada, este algoritmo, poderia ser alterado para, resolver de forma integrada a alocação de recursos em múltiplos projetos, com múltiplas habilidades, considerando a curva de aprendizado e ainda acoplado a um modelo de simulação, atendendo assim ao objetivo principal desta tese.

1.3 Estrutura da Tese

O restante desta tese está organizado nos seguintes tópicos: No capítulo 2 descreve a revisão da literatura; o capítulo 3 terá o detalhamento do referencial teórico; já o capítulo 4 possui os métodos e instrumentos previstos para este trabalho;

o capítulo 5 apresenta os resultados obtidos deste trabalho; por fim, no capítulo 6 são apresentadas as considerações finais e contribuições para a área, além da sugestão para trabalhos futuros.

2 Revisão da Literatura

A revisão da literatura realizada para este trabalho considerou como pilares os três problemas abordados: 1) escalonamento das tarefas em múltiplos projetos, 2) alocação de recursos com múltiplas habilidades em projetos, e 3) a curva de aprendizado dos recursos. Além destes problemas, foi realizada a revisão da literatura relacionada à metaheurística Colônia de Formigas.

Um primeiro problema envolvendo o escalonamento de tarefas em múltiplos projetos com recursos limitados é definido na literatura como *Resource Constrained multi-Project Scheduling Problem (*RCMPSP*).* Este é um tipo de problema de resolução complexa, que deriva de um problema conhecido como *Job Shop Problem (*JSP*).* O *RCMPSP* possui a finalidade de alocar *m* recursos disponíveis nas tarefas que precisam ser executadas em *p* projetos, gerando um número de combinações exponencial. Quanto maior o número de recursos e maior o número de projetos e tarefas, maior será o tempo para encontrar a solução ótima (ou mesmo subótimas), Em geral, a busca pela solução ótima é inviável em termos de tempo computacional (PINHA; AHLUWALIA, 2019).

O segundo problema, o qual trata da alocação de recursos limitados com múltiplas habilidades, é conhecido na literatura como *Multiskill Resource-Constrained Project Scheduling Problem* (MSRCPSP). Este problema também deriva do JSP, e de semelhante forma ao RCMPSP também é um problema com elevada complexidade computacional (LIN *et al.*, 2020).

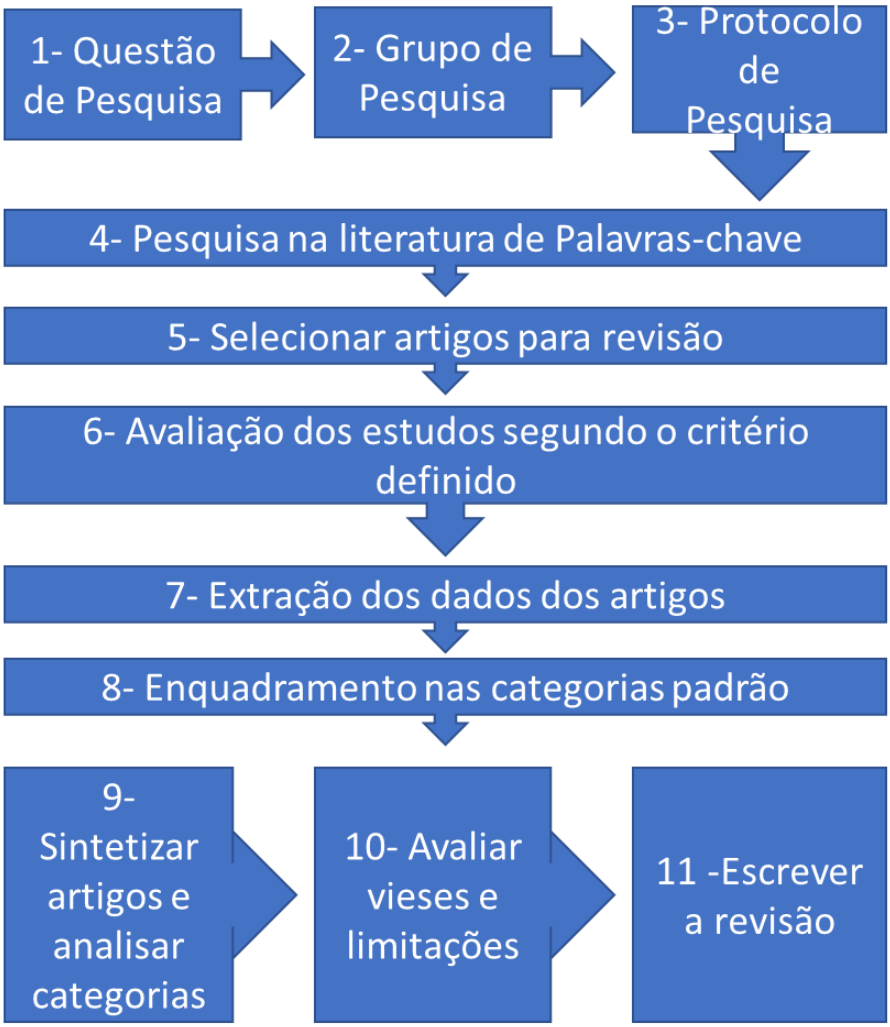
Por fim, é foi realizada uma revisão sobre a utilização da curva de aprendizado na composição do problema de alocação dos recursos, definido na literatura por *Learn and Forget Curve Model* (LFCM), e sobre a aplicação da metaheurística Colônia de Formigas (ACO, do inglês *Ant Colony Optimization*) na solução de problemas dessa classe

Foi utilizado um método de revisão sistemática da literatura para os problemas estudados nesta tese, e que será descrito no próximo tópico.

2.1 Método empregado na Revisão da Literatura

Para a revisão de literatura desta tese, foi utilizado o método de Revisão Sistemática da Literatura para cada um dos itens tratados, seguindo os 11 passos propostos por Petticrew e Roberts (2006) conforme modelo que segue na Figura 1.

Figura 1. Modelo de Revisão Sistemática da Literatura utilizado



Fonte: Adaptado de Petticrew e Roberts (2006)

Para a revisão sistemática, o primeiro passo, foi definir a questão de pesquisa. A questão de pesquisa foi definida como: Quais os artigos seminais com mais de 15 citações ou os artigos mais recentes relacionados a cada item da revisão da literatura? No passo 2, foi definido que o autor desta tese que iria realizar a revisão e análise.

O protocolo, constante do passo 3, foi definido que as buscas seriam realizadas nas bases Scopus e *Web of Science* (WOS) nos campos Título, Resumo e Palavras-chave por todos os tipos de artigos (com revisão por pares ou apresentados em congressos), em língua Inglesa, Espanhola ou Portuguesa. Os dados dos passos anteriores são iguais para todos os itens, e não se repetem nos tópicos abordados.

A partir do passo 4, pode haver alguma diferença entre os tópicos abordados, e foram sintetizados no início de cada tópico. Após a pesquisa realizada no item 4,

foram selecionados os artigos com uma quantidade superior a 15 citações, ou artigos publicados com data igual ou superior a 2019, com a revisão dos resumos destes artigos no passo 5.

No passo 6, as duas bases foram unificadas, eliminando-se as duplicações. Artigos aos quais não foi possível a leitura na íntegra por não conseguir obter a publicação também foram eliminados. Outro fator de eliminação do artigo, é que ele não seja aderente ao assunto estudado.

No passo 7, os dados foram categorizados conforme o tema abordado e posteriormente enquadrados no passo 8 nas categorias padrão, ou com ajustes nas categorias originais, gerando tabelas com estes dados, com exceção para o ACO que os dados não foram tabelados.

Com estes dados, foi realizada a sintetização dos dados no passo 9, e a análise de viés e limitações dos métodos no passo 10. O objetivo final foi expor os dados na revisão, que são apresentados ao longo deste capítulo, em seus tópicos, conforme o passo 11.

Na sequência, são apresentados os resultados da revisão da literatura do RCMPSP.

2.2 RCMPSP

O problema de alocação de recursos em tarefas de forma integrado em múltiplos projetos possui uma vasta aplicação em muitas áreas do conhecimento distintas. Buscando pelo termo “RCMPSP” nos campos Título, Resumo e Palavras-chave na base Scopus em 15/08/2021, foram encontrados 54 artigos diferentes, com aproximadamente 51 concentradas nas áreas de Engenharia, Ciências da Computação, Administração, Ciência de Dados e Matemática (Um artigo pode estar em mais de uma categoria). Na base WoS, na mesma data, foram encontrados 38 artigos diferentes, com a maioria concentrada na área de Engenharia e Ciências da Computação (Aproximadamente 35, tendo o mesmo comportamento que a base Scopus).

Dos estudos listados na base Scopus, 28 são artigos publicados em revistas com revisão por pares e 25 foram apresentados em conferências ou congressos. Na base WoS foram encontrados 27 artigos com revisão por pares e 11 foram apresentados em conferências ou congressos. Unificando o resultado das duas

bases, existe sobreposição dos artigos, porém, com a maioria sendo únicos, com uma lista total de 61 artigos únicos. Para a análise dos artigos listados, foram utilizados dois critérios: 1) artigos possuem mais de 15 citações, ou 2) artigos publicados nos últimos 3 anos, com data igual ou superior a 2019. Com estes critérios, foram listados 28 artigos.

Dos artigos selecionados, 3 não foram encontrados os artigos originais para leitura, e 1 estava numa língua fora do escopo do método empregado, sendo descartados da análise. Durante a leitura e classificação dos artigos, 2 foram descartados pois não eram uma aplicação direta de algoritmo para resolução do problema RCMPSP conforme Quadro 1.

A aplicação do RCMPSP é variada, sendo a primeira publicação encontrada nas bases, é de Chiu e Tsai (2002). Neste artigo, os autores utilizaram o RCMPSP para resolver o problema fluxo de caixa descontado na análise financeira, introduzindo penalidade por atraso e bônus se concluir antecipadamente no algoritmo determinístico de Programação não Linear Inteiro-Misto (*Mixed-Integer Nonlinear Programming* - MINLP) que encontra a solução ótima de forma mais eficiente que os algoritmos anteriormente propostos para uma solução multimodo para minimizar o atraso médio e o tempo dos projetos.

Os autores Zhuang e Yassine (2004) foram os primeiros que encontramos a propor um Algoritmo Genético (AG) para a solução do problema RCMPSP. Com a finalidade de reduzir o tempo total dos projetos, propuseram um AG com ponto único de cruzamento com um operador de atualização. Com esta solução, provê uma rápida convergência para soluções ótimas globais e detecta o paralelismo desejado nas soluções.

Para o lançamento de produtos em uma indústria farmacêutica, os autores Zapata *et al.* (2008) propuseram 3 algoritmos de *Mixed-Integer Linear Programming* (MILP) que foram adaptados considerando a grande incerteza existente neste tipo de projeto, com a finalidade de maximizar o lucro da empresa na alocação dos recursos. As 3 formulações propostas foram de: tempo discreto, tempo contínuo baseado em precedência e tempo contínuo baseado em eventos. Porém não obtiveram resultados satisfatórios com as formulações criadas, assim como as soluções com mais de um objetivo, conhecidas como multimodo, possuem.

Os autores Browning e Yassine (2010) utilizaram os algoritmos de regras de priorização para minimizar o tempo e o atraso médio das tarefas, tendo realizado a análise de 20 tipos de regras de prioridade (*Priority Rules* - PR) diferentes e mais de 12.000 testes para concluir que as regras de priorização possuem soluções pobres em alguns casos e que em outros, é preciso utilizar a regra adequada para chegar na solução deseja, produzindo tabelas de apoio de qual regra utilizar para o problema a resolver.

O MILP foi o algoritmo utilizado por Krüger e Scholl (2010) para minimizar atraso médio e minimizar tempo dos projetos para minimizar os custos do projeto. Com a finalidade de se aproximar das situações do cotidiano, os autores consideraram que existe um custo de tempo e financeiro de transferência entre o término de uma tarefa e o início de outra, como por exemplo de deslocamentos entre sites da empresa, além de um custo inicial para o projeto na alocação dos recursos, concluindo que é vital para soluções especializadas utilizar esta abordagem.

Palencia e Delgadillo (2012) conduziram um estudo de caso da automatização do processo de decisão via AG em uma indústria de montagem de ônibus, comprovando a redução do tempo total de entrega. Este é um dos poucos estudos com aplicação prática do RCMPSP.

Para melhorar a eficiência da solução do RCMPSP com multimodo, ou multiobjetivo de forma a ter melhores resultados na minimização do tempo e custo em tempos menores de processamento, os autores Dalfard e Ranjbar (2012) propuseram um algoritmo híbrido de PR com *Simulated Annealing (*SA*)*, alinhando um AS com a melhor PR para obter a solução ótima com uma metaheurística para diminuir o tempo total de buscas locais.

As incertezas, limitação dos recursos e a estrutura de rede da solução podem impactar na resolução do problema RCMPSP, ainda quando considerado que os projetos possuem prioridades diferentes e as tarefas não possuem um tempo definido para sua solução. Devido a estes fatos, os autores Zheng *et al.* (2013) utilizaram o algoritmo *Non-Dominated Sorting Genetic Algorithm II* (NSGA-II) para verificar o impacto no tempo e robustez da solução quando definidos parâmetros para cada um dos problemas iniciais citados, encontrando que projetos grandes possuem soluções mais robustas, principalmente quando utilizam um grande número de recursos, que

impactam na incerteza e robustez com relação linear entre ambas. Além disto, concluíram que as incertezas não produzem um impacto óbvio no agendamento.

Na artigo de Dong *et al.* (2012) na área de engenharia civil, os autores buscaram a minimização dos tempos de projeto na fase de finalização da construção utilizando um AG puro para a alocação dos recursos nos projetos. Este trabalho foi um estudo de caso direcionado, e encontrou bons resultados para a indústria aplicada.

Vázquez *et al.* (2013) criaram um algoritmo simples e rápido de aprendizado para determinar qual PR utilizar para cada instância do problema RCMPSP dentre as 34 PRs mais populares através de 26 problemas de simulação.

Sonmez e Uysal (2015) propuseram um novo algoritmo com o nome Algoritmo Genético Híbrido *Backward-Forward* (*Backward-Forward Hybrid Genetic Algorithm* - BFHGA) que combinou a parte robusta do algoritmo *Backward-Forward* (BF) de agendamento com AG e SA para reduzir o tempo total do projeto. Que obteve, segundo os autores, soluções ótimas para projetos simples e acima da média quando comparado com outras 5 metaheurísticas consideradas o “estado da arte” para a solução de múltiplos projetos.

Com a finalidade de propor alternativa para a tomada de decisão dos gerentes de projetos, Pérez *et al.* (2016) propuseram uma algoritmo chamado de Algoritmo Genético MultiModal (*Multi-Modal Genetic Algorithms* - MMGAs). Este algoritmo permite obter não somente a convergência para os ótimos globais, mas passa a verificar os ótimos locais, que possuem soluções igualmente boas para minimizar o tempo e o percentual do tempo médio de atraso. Os autores compararam o algoritmo MMGA com o AG puro e PR, concluindo que este é mais eficaz para encontrar a melhor solução ótima e encontrar múltiplos ótimos, bem como para reduzir o percentual do tempo médio de atraso, porém, PR possui um código melhor em termos de eficácia na minimização do tempo.

Yassine *et al.* (2017) propuseram 2 novas variações de AG para alocação de recursos em portifólio de projetos de pesquisa e desenvolvimento para reduzir o tempo total dos projetos, demonstrando que estes novos algoritmos geravam uma convergência rápida para a solução ótima global em relação a 31 regras de prioridade publicadas, produzindo uma tabela de direcionamento de qual a melhor escolha AG ou PR para o problema a ser tratado. OS AGs propostos receberam o nome de

Sampling GA (SaGA) e *Variable-Length GA* (VLGA), com a criação de diversas instâncias de solução do problema RCMPSP. Na sequência, utilizam o AG para avalição simultânea das instâncias propostas. A segunda solução trabalha com tamanhos variáveis do cromossomo, permitindo assim criação de instâncias diferentes para o problema.

Com uma finalidade diferente das até aqui discutidas, os autores Wang *et al.* (2017) buscam maximizar a qualidade e robustez da solução para o problema RCMPSP através de PR estocásticas, produzindo uma lista de recomendações de qual PR utilizar para resolução deste problemas, com o envolvimento de incertezas e produzindo soluções adequadas. As incertezas são importantes, pois os recursos podem possuir variação nos tempos de execução das, e devido a este fato, é necessário avaliar, através de algoritmos de otimização e podendo utilizar a simulação para identificar se os cenários apresentados se aproximam dos tempos e custos de execução das tarefas quando forem executadas.

Villafáñez *et al.* (2019) propuseram um novo algoritmo híbrido centralizado de PR, com Esquema de geração de cronograma paralelo (*Parallel Schedule Generation Scheme* - P-SGS) e Tarefa mínima com folga total (*Minimum Activity Total Slack* - MIN-SLK) para resolução do problema, e que quando comparado a outras PR, possui 42% de soluções boas em relação a estas.

As PRs como citado por Browning e Yassine (2010) e Wang *et al.* (2017) possuem soluções ótimas para diversos problemas, porém não são ótimas para todos os problemas, e devido a este fato, existem diversos trabalhos propostos para identificar a melhor PR para cada caso. Para melhorar este cenário, os autores Chen *et al.* (2019) propuseram um algoritmo híbrido decomposto em duas etapas e com o tempo de tarefas estocásticos, além de permitir a entrada de novos projetos ao longo do tempo, tendo uma aproximação do mundo real, no qual as tarefas são escalonadas nos projetos e posteriormente os recursos são alocados. Na primeira etapa, utiliza-se uma PR identificada dentre as propostas qual será a melhor para aquele problema, permitindo o recalculo com novos projetos. Após esta etapa, o algoritmo verifica qual a PR mais adequada naquele cenário e conclui a resolução do problema. Com este algoritmo híbrido, existe uma aproximação de problemas reais, e com um tempo menor de solução.

Com a finalidade de reduzir o impacto no tempo tardio e no custo, Tian *et al.* (2019) utilizaram o AG com a técnica da cadeia crítica para o sequenciamento de tarefas dos projetos. Esta é uma técnica na qual o gerente de projetos pode se utilizar para verificar quais os recursos ou tarefas que gerarão maior impacto no tempo e custo total do projeto, e permite a utilização de recursos adicionais que podem ser utilizados para minimizar o impacto dos recursos críticos no projeto, além de aumentar a estabilidade do agendamento dos projetos.

As metaheurísticas produzem soluções ótimas ou subótimas em tempos relativamente baixos de processamento, porém, como nem sempre produzem a solução ótima, os estudos buscam algoritmos híbridos para aumentar essa eficiência. Um dos estudos é de Sanchez *et al.* (2019) que utilizaram um algoritmo híbrido de Colônia de formigas (*Ant Colony Optimization* - ACO) com o algoritmo *Satisfiability Modulo Theories* (SMT) de busca local para minimizar o tempo total e o tempo total de atraso do projeto, encontrando soluções melhores que outros algoritmos com base de comparação. Este trabalho utilizou de dois estágios de buscas, que gerou a diferença para encontrar as melhores soluções, algoritmo este que consegue tornar o algoritmo mais eficiente, e que não é empregado nos AGs. Além deste fato, o ACO foi criado para resolução do problema do caixeiro viajante, quando se precisa passar em todos os pontos da solução, que é muito semelhante ao problema de alocação de recursos em projetos, sendo um algoritmo muito eficiente neste tipo de solução.

A gestão de múltiplos projetos flexíveis (que permitem a escolha de qual projeto executar e quando) com a finalidade de reduzir o tempo de pagamento do projeto inteiro é o objetivo dos autores Hoffmann *et al.* (2020) ao propor um novo algoritmo de AG para este tipo de problema. Os projetos flexíveis podem ter estrutura e tipo do projeto para atender as necessidades da empresa, e o quanto antes terminarem, quanto antes os pagamentos podem ser efetivados. Estes tipos de projetos são empregados em áreas de inovação, e com grande incerteza, que é um cenário diferente do proposto nesta tese.

O Quadro 1 apresenta a compilação de todos os trabalhos abordados até aqui, com a categorização deles.

Quadro 1. Artigos RCMPSP analisados

Estudo	Citações	Tarefa				Algoritmo			
		Recursos Alocados	Duração	Ordenação	Área	Objetivo	Algoritmo	Dados de teste	Tempo
Chiu e Tsai (2002)	26	Lim	Fix	Sim	Contábil	Minimizar tempo e Maximizar lucro	Programação não Linear Inteiro-misto - MINLP	Empresa	D
Zhuang e Yassine (2004)	23	Lim	Amb	Sim	Geral	Minimizar tempo	Algoritmo Genético – AG	Gerado	E
Zapata *et al.* (2008)	32	Lim	Fix	Sim	Farmacêutica	Maximizar lucro	Programação Linear Inteiro-misto - MILP	Empresa	D
Browning e Yassine (2010)	159	Lim	Fix	Sim	Geral	Minimizar tempo e atraso médio	Regra de Priorização - PR	Gerado	D
Krüger e Scholl (2010)	41	Lim	Fix	Sim	Geral	Minimizar custo	Programação Linear Inteiro-misto - MILP	Gerado	D
Palencia e Delgadillo (2012)	16	Lim	Fix	Sim	Montagem de Ônibus	Minimizar tempo e Maximizar entrega	Algoritmo Genético - AG	Empresa	E
Dalfard e Ranjbar (2012)	15	Lim	Fix	Sim	Geral	Minimizar tempo, soma do atraso total e o número de atividades atrasadas	Algoritmo híbrido – *Simulated Annealing* com Regras de Priorização	Gerado	E
Dong *et al.* (2012)	15	Lim	Fix	Sim	Construção	Minimizar tempo	Algoritmo Genético - AG	Empresa	E
Zheng *et al.* (2013)	17	Lim	Var	Sim	Geral	Minimizar tempo e Maximizar robustez	Algoritmo Genético de Classificação Não Dominado II - NSGA-II	Gerado	E
Vázquez *et al.* (2013)	16	Lim	Amb	Sim	Geral	Minimizar tempo e média de tempo de espera	Regra de Priorização	Gerado	D
Sonmez e Uysal (2015)	24	Lim	Fix	Sim	Geral	Minimizar tempo	Algoritmo Genético Híbrido *Backward-Forward* – BFHGA	Ambos	E
Pérez *et al.* (2016)	20	Lim	Amb	Sim	Geral	Minimizar tempo e média de tempo de espera	Algoritmo Genético MultiModal – MMGAs	Gerado	E
Yassine *et al.* (2017)	31	Lim	Amb	Sim	Geral	Minimizar tempo	Algoritmo Genético	Gerado	E

Quadro 1. Artigos RCMPSP analisados

Estudo	Citações	Tarefa				Algoritmo			
		Recursos Alocados	Duração	Ordenação	Área	Objetivo	Algoritmo	Dados de teste	Tempo
Wang *et al.* (2017)	21	Var	Var	Sim	Geral	Maximizar qualidade e robustez	Regra de priorização	Gerado	E
Villafáñez *et al.* (2019)	15	Lim	Amb	Sim	Geral	Minimizar tempo	Algoritmo híbrido de Regra de Priorização - P-SGS/MINSLK	Gerado	E
Chen *et al.* (2019)	8	Lim	Var	Sim	Geral	Maximizar qualidade e robustez	Algoritmo híbrido de Regras de Priorização	Gerado	E
Tian *et al.* (2019)	1	Lim	Var	Sim	Geral	Minimizar tempo e custo tardio da cadeia crítica	Algoritmo Genético - AG	Gerado	E
Sanchez *et al.* (2019)	1	Lim	Fix	Sim	Geral	Minimizar tempo e atraso	Algoritmo híbrido – Colônia de formigas - ACO e Procedimento SMT de busca local	Gerado	E
Hoffmann *et al.* (2020)	0	Var	Var	Não	Geral	Minimizar custo	Algoritmo Genético - AG	Gerado	E
Villafanez *et al.* (2020)	0	Lim	Fix	Sim	Geral	Minimizar tempo	Regra de Priorização	Gerado	D
Issa *et al.* (2021)	3	Amb	Amb	Amb	Geral	Minimizar tempo	Programação Linear Inteiro-misto - MILP	Gerado	D
Uysal *et al.* (2021)	0	Lim	Fix	Não	Geral	Minimizar tempo	Algoritmo Genético - AG	Gerado	E

Lim: Limitados; Amb: Ambos; Var: Variável; Fix: Fixo; D: Determinístico; E: Estocástico

Fonte: Autor

Como as empresas possuem diferenças de priorização dentro da organização, os autores Villafanez *et al.* (2020) propuseram um algoritmo baseado em PR publicadas para determinar a priorização das tarefas com base na estrutura da empresa reduzindo o tempo total dos projetos. Os algoritmos de PR são eficientes em encontrar ao menos uma solução viável, porém precisam ser adequados a cada tipo de projeto / solução necessária, com a necessidade de muitas PRs para a resolução de muitos problemas diferentes, tendo que construir um algoritmo para determinar a melhor solução para cada problema, o que dificulta uma generalização das soluções, e não será utilizada nesta tese.

Os autores Issa *et al.* (2021) utilizam um algoritmo de MILP para resolução do RCMPSP com três categorias, chamados de ABD. A categoria A se refere a tarefas de projetos que possuem duração fixa e número fixo de recursos. A categoria B se refere a tarefas que podem ser desempenhadas por recursos da categoria A mas que as não preempções das tarefas são relaxadas e pôr fim a categoria D, na qual as tarefas dos projetos podem utilizar de recursos flexíveis com durações variadas e podem ou não ser interrompidas. O algoritmo reduziu substancialmente o tempo total do projeto quando permitiu a liberação do recurso para utilização em outros projetos. Este é um outro tipo de problema existente na literatura, e que não será abordado.

Por fim, os autores Uysal *et al.* (2021) realizaram um estudo da implementação do AG para resolver o problema RCMPSP utilizando o processamento da GPU (*Graphical Processing Units*), que é a unidade de processamento gráfica do computador com o processamento via CPU unidade que controla todo o processamento do computador, demonstrando grande eficiência da GPU em relação a CPU principalmente em projetos grandes, indicando um caminho para melhorar o tempo de processamento de algoritmos. Porém, por se tratar de um algoritmo de processamento diferenciado, é de difícil programação e exige equipamentos específicos, não conseguindo ser generalizado para qualquer situação, e por este motivo, não será considerado nesta Tese.

Na compilação de todos os artigos analisados, o algoritmo determinístico é utilizado em 7 dos estudos analisados (BROWNING; YASSINE, 2010; CHIU; TSAI, 2002; KRÜGER; SCHOLL, 2010; VÁZQUEZ *et al.*, 2013; VILLAFANEZ *et al.*, 2020; ISSA *et al.*, 2021; ZAPATA *et al.*, 2008) para obter a solução ótima do problema RCMPSP e com dois algoritmos predominantes, sendo baseados em regras de

priorização ou programação linear mista, conforme pode-se verificar no Quadro 1. Os estudos analisados indicam que os algoritmos determinísticos ainda estão sendo estudados e permitem evolução, pois ainda são publicados estudos atualmente com estes algoritmos.

Os algoritmos determinísticos geram a solução ótima, porém, possuem um tempo de processamento elevado para obter esta solução (WANG *et al.*, 2017). Devido a este fato, muitos autores optam por metaheurísticas para resolução do problema. A principal metaheurística utilizada é o Algoritmo Genético (AG) e suas variações que foram utilizadas em 10 dos estudos analisados (DONG *et al.*, 2012; HOFFMANN *et al.*, 2020; PALENCIA; DELGADILLO, 2012; PÉREZ *et al.*, 2016; SONMEZ; UYSAL, 2015; TIAN *et al.*, 2019; UYSAL *et al.*, 2021; YASSINE *et al.*, 2017; ZHENG *et al.*, 2013; ZHUANG; YASSINE, 2004). O AG é amplamente utilizado devido a suas características, produzindo soluções ótimas ou subótimas com um tempo de processamento baixo (UYSAL; SONMEZ; ISLEYEN, 2021) e tanto nos estudos mais citados, quanto nos estudos atuais, continua sendo amplamente utilizado.

Wang *et al.* (2017) analisaram que as regras determinísticas são as melhores soluções, porém, apresentam tempo elevado para a resolução, ao contrário das metaheurísticas que apresentam baixo tempo para resolução, porém nem sempre encontrando a melhor solução para o problema. Sendo assim, os autores propuseram um algoritmo alternativo híbrido, que utiliza dados estocásticos que permitem soluções mais rápidas porém utilizando um algoritmo de Regra de Priorização, que conforme os autores Wang *et al.*, (2017) são amplamente utilizados em *softwares* comerciais devido a produzir as soluções ótimas.

Além deste estudo, mais outros 5 estudos adotaram algoritmos híbridos (CHEN *et al.*, 2019; DALFARD; RANJBAR, 2012; SANCHEZ *et al.*, 2019; SONMEZ; UYSAL, 2015; VILLAFÁÑEZ *et al.*, 2019) devido à complexidade de resolução do problema e a deficiência que os algoritmos, sejam eles determinísticos ou estocásticos, possuem para determinados problemas. O algoritmo ACO de forma híbrida foi utilizado em apenas 1 estudo, porém, conforme os autores, foi mais adequado, e produziu soluções melhores que outros algoritmos híbridos, principalmente em comparação com o AG.

Um fator importante nos estudos é quanto ao objetivo, que pode conter um objetivo único ou ser multimodo. O objetivo único possui análise de apenas 1 fator na função objetivo, seja o tempo, custo, qualidade ou outro, e este foi adotado em 11 estudos (DONG *et al.*, 2012; HOFFMANN *et al.*, 2020; ISSA *et al.*, 2021; KRÜGER; SCHOLL, 2010; SONMEZ; UYSAL, 2015; UYSAL *et al.*, 2021; VILLAFÁÑEZ *et al.*, 2019; VILLAFANEZ *et al.*, 2020; ZAPATA *et al.*, 2008; ZHUANG; YASSINE, 2004), ou seja, metade. A outra metade que adota a solução de um modo múltiplo, 9 possuem no mínimo o fator de redução do tempo, e somente dois estudos que adotam o estudo da qualidade e robustez do projeto (CHEN *et al.*, 2019; WANG *et al.*, 2017).

Na análise do algoritmo, e da área de tarefa do estudo, nota-se uma dificuldade grande dos pesquisadores para validar os algoritmos propostos e analisar o real impacto deles em uma empresa, com apenas 5 estudos utilizando de dados reais para esta validação e em diversas áreas do conhecimento (CHIU; TSAI, 2002; DONG *et al.*, 2012; PALENCIA; DELGADILLO, 2012; SONMEZ; UYSAL, 2015; ZAPATA *et al.*, 2008;). Além deste fato, um outro fator importante é poder realizar o comparativo do algoritmo proposto com outras soluções anteriormente propostas, e verificar a eficiência e eficácia da solução, tendo algumas bases disponibilizadas, sendo que as mais utilizadas nos estudos é o de Browning e Yassine (2010) e Pérez *et al.* (2016).

Pelas características do problema estudado quase todos os estudos utilizam uma quantidade limitada de recursos e com a ordenação da tarefa sendo gerada, com dominância em praticamente todos os estudos. Uma maior variação no item Tarefas, com 12 estudos apresentando esta variação, se refere ao tempo de duração da tarefa com a utilização do tempo variável ou um algoritmo misto, podendo variar entre fixo e variável.

Por fim, o RCMPSP é de difícil solução e ainda apresenta uma dificuldade grande para encontrar a melhor solução em um tempo computacional baixo, de forma que possa ser utilizado para encontrar soluções adequadas para os problemas nas empresas, por este motivo continua sendo estudado e possui uma relevância em termos acadêmicos e profissionais, buscando-se o equilíbrio entre solução adequada (podendo ser a ótima) e tempo de processamento.

No próximo item, seguem os resultados da revisão da literatura para o MSRCPSP.

2.3 MSRCPSP

O problema de alocação de recursos em tarefas de forma integrado com múltiplas habilidades possui uma vasta aplicação em muitas áreas do conhecimento distintas. Buscando pelo termo “MSRCPSP” ou “MS-RCPSP” nos campos Título, Resumo e Palavras-chave na base Scopus no dia 15/08/2021, foram encontrados 33 artigos diferentes. Na base WoS foram encontrados 28 artigos diferentes na mesma data, com a maioria dos artigos em ambas as bases concentrada na área de ciências da computação (Acima de 25 artigos nas duas bases).

Dos estudos listados na Scopus, 21 são artigos publicados em revistas com revisão por pares e 12 foram apresentados em conferências ou congressos. Na base WoS foram encontrados 20 artigos com revisão por pares e 8 foram apresentados em conferências ou congressos. Unificando o resultado das duas bases, existe sobreposição dos artigos, porém, alguns únicos, com uma lista total de 34 artigos únicos. Para a análise dos artigos, foram utilizados dois critérios, que foram os que possuem mais de 15 citações, ou artigos dos últimos 3 anos, com data igual ou superior a 2019, com uma lista total de 28 artigos selecionados.

Dos artigos selecionados, não foi possível recuperar o texto original de 2 deles, sendo descartados das análises. Durante a leitura e classificação dos artigos, 3 foram descartados pois não eram uma aplicação direta de algoritmo para resolução do problema RCMPSP.

O primeiro artigo encontrado nas duas bases, é o artigo de Skowroński *et al.* (2013) no qual os autores buscaram uma solução para minimizar o tempo total do projeto, bem como o custo alocando recursos com múltiplas habilidades em projeto na indústria automobilística utilizando um algoritmo de busca Tabu para atingir o objetivo. Assim como este artigo, praticamente todos se utilizaram de algoritmos estocásticos em suas buscas, com exceção de Myszkowski *et al.* (2015b) que utilizaram um algoritmo determinístico de estatística de agendamento, conforme pode-se verificar no Quadro 2.

Quadro 2. Artigos MSRCPSP analisados

Estudo	Citações	Tarefa						Algoritmo			
		Recursos Alocados	Variação Proficiência Habilidade	Aprendizado	Duração	Ordenação	Área	Objetivo	Algoritmo	Dados de teste	Tempo
Skowroński *et al.* (2013)	15	Lim	Não	Não	Fix	Sim	Auto motivo	Minimizar tempo e custo	Busca Tabu	Gerados	E
Myszkowski *et al.* (2015)	68	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo e custo	Algoritmo híbrido de Colônia de formigas com Regras de Priorização	iMOPSE	E
Myszkowski *et al.* (2015b)	20	Var	Não	Não	Var	Amb	Geral	Minimizar tempo e custo	Estimativas de Agendamento de Dificuldade do Projeto	Gerados	D
Almeida *et al.* (2016)	43	Lim	Não	Não	Var	Sim	Geral	Minimizar tempo e Maximizar Qualidade	Esquema de Programação Paralela	Gerados	E
Zheng *et al.* (2017)	55	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Algoritmo de Otimização baseado em Ensino-Aprendizagem (TLBO)	iMOPSE	E
Wang e Zheng (2018)	71	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Algoritmo híbrido de otimização de mosca-das-frutas baseado em conhecimento (KBFOA)	Gerados	E
Myszkowski *et al.* (2018)	41	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Algoritmo híbrido de Evolução Diferencial e Algoritmo Guloso	iMOPSE	E
Laszczyk e Myszkowski (2019)	21	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo e custo	Algoritmo híbrido Genético de Classificação Não Dominado II (NSGA-II)	iMOPSE	E
Hosseinian *et al.* (2019)	11	Var	Sim	Sim	Fix	Sim	Geral	Minimizar tempo e tempo total com recursos com menor habilidade	Algoritmo Genético melhorado de Classificação Não Dominado II (IM-NSGA-II)	Gerados	E

Quadro 2. Artigos MSRCPSP analisados

Estudo	Citações	Tarefa						Algoritmo			
		Recursos Alocados	Variação Proficiência Habilidade	Aprendizado	Duração	Ordenação	Área	Objetivo	Algoritmo	Dados de teste	Tempo
Zhu *et al.* (2019)	8	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo e custo	Algoritmo híbrido de Algoritmo Guloso (*Greed*) e Algoritmo Genético (AG) (DOMVO)	iMOPSE	E
Joshi *et al.* (2019)	2	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Algoritmo de Otimização baseado em Ensino-Aprendizagem (TLBO)	Gerados	E
Quoc *et al.* (2019)	0	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Baseado na busca Cuckoo (R-CSM)	iMOPSE	E
Lin *et al.* (2020)	21	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Algoritmo híbrido de Hyper-Heuristica com Programação Genética (GP-HH)	iMOPSE	E
Hosseinian e Baradaran (2020)	6	Lim	Não	Não	Fix	Sim	Gás	Minimizar tempo e custo	Algoritmo híbrido de Otimização de GreyWolf com base em Pareto (P-GWO) e Algoritmo Multi-Objetivo baseado em Fibonacci (MOFA)	Ambos	E
Dai *et al.* (2020)	2	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Abordagem Geral de Pesquisa de Vizinhança Variável (GVNS)	iMOPSE	E
Quoc *et al.* (2020)	1	Lim	Não	Não	Fix	Sim	Têxtil	Minimizar tempo	Baseado na busca Cuckoo (R-CSM)	Ambos	E
Quoc *et al.* (2020)	1	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Baseado na busca Cuckoo (R-CSM)	iMOPSE	E
Quoc *et al.* (2020)	0	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Baseado na Evolução Diferencial	iMOPSE	E
Quoc *et al.* (2020)	0	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo	Baseado na Evolução Diferencial	iMOPSE	E
Joshi *et al.* (2020)	0	Lim	Sim	Não	Fix	Sim	Geral	Minimizar tempo e tempo total com	Algoritmo de Otimização baseado em Ensino-Aprendizagem (TLBO)	Gerados	E

Quadro 2. Artigos MSRCPSP analisados

Estudo	Citações	Tarefa						Algoritmo			
		Recursos Alocados	Variação Proficiência Habilidade	Aprendizado	Duração	Ordenação	Área	Objetivo	Algoritmo	Dados de teste	Tempo
								recursos com menor habilidade			
Zhu *et al.* (2021)	2	Lim	Não	Não	Fix	Sim	Geral	Minimizar tempo e custo	Programação Genética Multi-Objetivo baseada em Decomposição Hyper-Heuristica (MOGP-HH / D)	iMOPSE	E
Snauwaert e Vanhoucke (2021)	1	Lim	Sim	Não	Var	Sim	Geral	Minimizar tempo	Algoritmo Genético (AG)	Gerados	E
Myszkowski e Laszczyk (2021)	1	Lim	Sim	Não	Var	Sim	Geral	Minimizar tempo e custo	Algoritmo híbrido de Genético de Torneio não Dominado (NTGA2)	iMOPSE	E

Lim: Limitados; Amb: Ambos; Var: Variável; Fix: Fixo; D: Determinístico; E: Estocástico

Fonte: Autor

Os artigos de Myszkowski *et al.* (2015) e Myszkowski *et al.* (2015b) no qual os autores utilizaram um algoritmo híbrido de colônia de formigas com Regras de Priorização e Estimativas de Agendamento de Dificuldade do Projeto respectivamente para resolver o problema MS-RCPSP é um marco para este problema, visto que utilizou e disponibilizou a base de dados iMOPSE (*Intelligent Multi Objective Project Scheduling Environment)* de forma on-line para outros autores, além de disponibilizar uma área para que os pesquisadores registrem seus algoritmos e tempos de processamento, permitindo a comparação em termos de eficiência e eficácia, com registro de 13 artigos que utilizaram esta base em nossa pesquisa (DAI *et al.*, 2020; LASZCZYK; MYSZKOWSKI, 2019; LIN *et al.* 2020; MYSZKOWSKI *et al.*, 2015; MYSZKOWSKI *et al.* 2018; MYSZKOWSKI; LASZCZYK, 2021; QUOC *et al.*, 2019; QUOC *et al.*, 2020; QUOC *et al.*, 2020; QUOC *et al.*, 2020; ZHENG *et al.* 2017; ZHU *et al.*, 2019; ZHU *et al.*, 2021). Este fator é importante, pois assim como para o RCMPSP, o MSRCPSP é difícil de ter dados coletados das empresas, tendo apenas dois estudos com dados reais (HOSSEINIAN; BARADARAN, 2020; QUOC *et al.*, 2020), e que os autores Quoc *et al.* (2020) ainda propõe um novo tipo de problema para dados reais, denominado Real-RCPSP.

Almeida *et al.* (2016) utilizaram um Esquema de Programação Paralela cujo objetivo do algoritmo proposto é minimizar o tempo e aumentar a qualidade da entrega dos projetos através de soluções subótimas para diferentes tamanhos de projeto, tendo uma performance superior a sistema comerciais em tempo de processamento e na qualidade da solução. Para encontrar estas soluções, os autores acrescentaram um peso para cada recurso além de realizar o agrupamento das tarefas para facilitar a resolução.

Com a finalidade de minimizar o tempo total do projetos, os autores Zheng *et al.* (2017) propuseram o Algoritmo de Otimização baseado em Ensino-Aprendizagem (*Teaching–Learning-Based Optimization* - TLBO), utilizando de duas listas, uma com as habilidades dos recursos e outra com os recursos disponíveis, cuja combinação gera as soluções factíveis com um balanceamento entre as listas de exploração global e local, além do acréscimo de uma fase de reforço com a permutação das duas listas. A solução proposta, ao se comprar pelas bases de comparação existentes, se mostrou mais eficiente e eficaz com os algoritmos atém então utilizados.

Os autores Wang e Zheng (2018) utilizaram um algoritmo híbrido de otimização de mosca-das-frutas baseado em conhecimento *(Knowledge-Based Fruit Fly Optimization Algorithm* - KBFOA*)* para minimizar o tempo total do projeto. Foi a primeira vez que o algoritmo da mosca da fruta foi utilizado em problemas do tipo MSRCSP, e para tanto, os autores desenharam um operador de busca baseada na vizinhança baseado na pesquisa por cheiro e além deste fato, acrescentaram um multe enxame para evitar a armadilha dos ótimos locais demonstrando por comparativo de execução com as bases de comparação ser mais eficiente e eficaz que outros algoritmos existentes.

Os autores Myszkowski *et al.* (2018) propuseram o algoritmo híbrido de Evolução Diferencial e Algoritmo Guloso para minimizar o tempo total do projeto. Os autores transformaram o espaço de solução de discreto para continuo com uma representação indireta dele, além de realizar diversos testes para encontrar a melhor parametrização para o algoritmo, e dos elementos do algoritmo, como inicialização, eliminação da clonagem, mutação etc. Um dos autores – Myszkowski – é um dos especialistas no problema MSRCPSP, tendo 5 artigos analisados e diversos algoritmos propostos para resolução do problema.

Além dos citados anteriormente, ainda podemos destacar Laszczyk e Myszkowski (2019) utilizaram um Algoritmo híbrido de Genético de Torneio não Dominado (*Non-dominated Tournament Genetic Algorithm* - NTGA), baseado no NSGA-II para resolução do problema. Como o próprio nome do algoritmo nos diz, a principal modificação está em permitir valores altos de torneio, retirando-se a classificação não dominada e substituição de todos os filhos que foram substituídos na geração atual. Em comparativos com outros algoritmos propostos pelos autores e outros, se mostrou mais eficiente e eficaz e mais recentemente Myszkowski e Laszczyk (2021) com a evolução deste algoritmo para o NTGA2 que permite uma solução multimodo e trata algumas inconsistências do primeiro algoritmo.

Um algoritmo de otimização multiverso de oposição discreta (*Discrete Oppositional Multi-Verse Optimization* – DOMVO) multimodo para minimizar tempo e custo foi proposto por Zhu *et al.* (2019). Se trata de um algoritmo híbrido entre o algoritmo guloso (Greed) e o algoritmo AG, que demonstrou ser mais eficiente e eficaz em relação aos algoritmos considerados estado da arte pela base de comparação iMOPSE. Zhu *et al.* (2021) propuseram um segundo algoritmo denominado

Programação Genética Multiobjetivo baseada em Decomposição Hiper Heurística (*Decomposition-Based Multi-Objective Genetic Programming Hyper-Heuristic* - MOGP-HH / D), que assim como o primeiro é baseado no AG e se mostrou eficiente na sua solução.

Um algoritmo de TLBO foi proposto para a resolução do problema por Joshi *et al.* (2019) com a finalidade de reduzir o tempo e o custo do projeto. Com o acréscimo do conceito de alto-aprendizado e exame no processo de busca os autores tentaram evitar a armadilha do ótimo local e encontraram uma solução mais rápida que o AG. Os autores Joshi *et al.* (2020) evoluíram a solução para um multimodo, que além de minimizar o tempo, busca uma minimização do tempo com recursos menos capacitados, ou seja, buscando redução do custo do projeto.

Quoc *et al.* (2019), Quoc *et al.* (2020) e Quoc *et al.* (2020) apresentaram um algoritmo Baseado no busca Cuckoo (R-CSM) com a finalidade de redução do tempo total do projeto em uma indústria têxtil, utilizando uma função de realocação para a convergência para a solução ótima global. A grande contribuição dos autores, é cunhar um novo termo, para quando utilizado de dados reais para o problema, denominado Real-RCPSP (*Real-Resource-Constrained Project Scheduling Problem*). Além dos dados reais, os autores compararam o algoritmo utilizando a base iMOPSE, concluindo que o algoritmo possui uma qualidade de solução superior.

O algoritmo híbrido de Hyper-Heuristica com Programação Genética *(Genetic Programming Hyper-Heuristic* - GP-HH*)* foi proposto por Lin *et al.* (2020) para resolução do problema com a finalidade de minimizar o tempo total. Neste algoritmo, as tarefas são encadeadas, e aplica-se um algoritmo baseado em reparação para geração das soluções factíveis. Após, aplicam-se 10 regras de heurísticas simples que foram projetadas para atuar em conjunto como uma camada de heurística de baixo nível. Por fim, aplicam a programação genética baseado no AG como uma camada de alto nível para gerenciar as camadas de baixo nível.

A proposta do algoritmo de Hosseinian e Baradaran (2020) é utilizar um Algoritmo híbrido de Otimização de GreyWolf com base em Pareto (*Pareto-based GreyWolf Optimizer* - P-GWO) para a solução principal com efeito de deterioração e recursos financeiros limitados em conjunto com um Algoritmo Multiobjetivo baseado em Fibonacci (*Multi-Objective Fibonacci-based Algorithm* - MOFA) para as solução

não dominadas, concluindo que este algoritmo possui performance superior ao NSGA-II e outros aos quais foi comparado.

Os autores Dai *et al.* (2020) apresentaram um algoritmo diferente dos empregados até então, utilizando um algoritmo com Abordagem Geral de Pesquisa de Vizinhança Variável (*General Variable Neighborhood Search approach* - GVNS) com um passo de deterioração visando a minimização do tempo total do projeto encontrando soluções de qualidade.

Quoc *et al.* (2020) e Quoc *et al.* (2020) baseado no algoritmo de Evolução Diferencial (*Differential Evolution Metaheuristic* - DEM) com o objetivo de minimizar o tempo total do projeto com a convergência mais rápida para o ótimo global.

Por fim, Snauwaert e Vanhoucke (2021) propuseram um novo algoritmo baseado no AG, com uma nova abordagem para os cruzamentos e dois problemas novos de busca local, utilizando tarefas com tempos variáveis atacando o problema de profundidade e amplitude. Os autores definem como o algoritmo como o novo estado da arte em comparação com os atuais via base de comparação.

Para resolução deste problema, não existe um algoritmo que seja amplamente utilizado pelos autores. O AG com suas variações e em conjunto com outros algoritmos é empregado em 7 dos artigos analisados (HOSSEINIAN *et al.*, 2019; LASZCZYK; MYSZKOWSKI, 2019; LIN *et al.*, 2020; MYSZKOWSKI; LASZCZYK, 2021; SNAUWAERT; VANHOUCKE, 2021; ZHU *et al.*, 2019; ZHU *et al.*, 2021). Além de destacar o AG, é possível verificar em dados do Quadro 2 que alguns autores se utilizam de algoritmos híbridos na busca da melhor solução do problema, verificando-se que 9 dos artigos utilizaram combinação de algoritmos (HOSSEINIAN; BARADARAN, 2020; LASZCZYK; MYSZKOWSKI, 2019; LIN *et al.*, 2020; MYSZKOWSKI *et al.*, 2015; MYSZKOWSKI *et al.*, 2018; MYSZKOWSKI; LASZCZYK, 2021; WANG; ZHENG, 2018; ZHU *et al.*, 2019; ZHU *et al.*, 2021). Os autores que utilizaram os algoritmos híbridos indicam esta abordagem para diminuir o tempo de processamento do algoritmo e evitar falhas que ocorrem com frequência nos algoritmos quando utilizados de forma isolada. De semelhante forma ao encontrado no problema RCMSP, temos apenas um autor utilizando um algoritmo híbrido de ACO para a solução do problema, e que foi desenvolvido por um dos autores que mais publicaram sobre MSRCPSP.

A revisão da literatura do *Learn and Forget* é verificada na sequência.

2.4 Learn and Forget

O problema de curva de aprendizado e esquecimento é bem antigo, com o primeiro estudo iniciado em 1936 por Wright. Deste estudo, existem diversas derivações e milhares de artigos nas mais diversas áreas do conhecimento[1] Para esta tese, foram realizadas buscas nas bases Scopus e WoS no dia 21/08/2021 direcionadas ao tema ao qual associamos a curva de aprendizado, que são as habilidades dos recursos. Devido a este fato, utilizamos os termos "*learn**" e "*forget**" e "*curve model**" nos campos Título, Resumo e Palavras-chave e por *skill** em todos os campos dos artigos nas bases citadas anteriormente para recuperar os artigos desejados para análise.

Na base Scopus, foram encontrados 55 artigos diferentes. Na base WoS foram encontrados 15 artigos diferentes, com a maioria dos artigos em ambas as bases concentrada na área de Ciências da Computação e Engenharia (Acima de 50 artigos na Scopus e 12 na WoS, com sobreposição das áreas nas duas bases).

Dos estudos listados na Scopus, 34 são artigos publicados em revistas com revisão por pares e 21 foram apresentados em conferências ou congressos. Na base WoS foram encontrados 12 artigos com revisão por pares e 3 foram apresentados em conferências ou congressos. Unificando o resultado das duas bases, existe sobreposição dos artigos, porém com uma lista total de 46 artigos únicos. Para a análise dos artigos, foram utilizados dois critérios, que foram os que possuem mais de 30 citações, ou artigos dos últimos 3 anos, com data igual ou superior a 2019, com uma lista total de 41 artigos selecionados.

Dos artigos selecionados, não foi possível recuperar o texto original de 1, e foi descartado. Durante a leitura e classificação dos artigos, 15 foram descartados pois não eram uma aplicação direta de algoritmo para a área de estudo atual, compilando os dados conforme o Quadro 3.

[1] Encontrados mais de 6.000 artigos na base Scopus pelos termos *learn* and forget**.

Quadro 3. Artigos Learn e Forget analisados

Estudo	Citações	Algoritmo	Geral	Dificuldade					Aprendizado		Esquecimento				Habilidade	
		Objetivo	Algoritmo	Dados de teste	Tempo	Habilidade Usuário	Nível Item	Nível Habilidade	Ganha	Perde	Intervalo Tempo do item	Intervalo Tempo da Habilidade	Intervalo Tempo de Interação	Uso da Habilidade	Aprende Sobre Seq. ordenada	Habilidade Prévia
Shtub *et al.* (1993)	41	Análise de impcato em linha de produção e na Qualidade	Bailey e Globerson	Manufatura Tecnológica	E	N	N	N	S	S	S	S	N	N	S	N
Li e Cheng (1994)	38	Análise Tempo e Custo	Nadler and Smith	Gerados	E	N	N	N	S	S	S	S	N	N	S	N
Badiru (1995)	44	Análise Linha de Produção e Qualidade	Womer e Badiru	Indústria	E	N	N	N	S	S	S	S	N	N	S	N
Dar-El *et al.* (1995)	43	Impacto das Curvas de Aprendizado e Esquecimento	Aprendizado -Sabag, 1988 e Esquecimento - Hence	Empresa	D	N	N	N	S	S	S	S	N	N	S	N
Jaber e Bonney (1996)	138	Maximizar produção e reduzir custo	Wright, Carlson e Rowel (Esquecimento) - Adaptação - Learn–	Bases Educacionais	E	N	N	N	S	S	S	S	N	N	S	N

Quadro 3. Artigos Learn e Forget analisados

Estudo	Citações	Algoritmo	Geral	Dificuldade					Aprendizado		Esquecimento				Habilidade	
		Objetivo	Algoritmo	Dados de teste	Tempo	Habilidade Usuário	Nível Item	Nível Habilidade	Ganha	Perde	Intervalo Tempo do item	Intervalo Tempo da Habilidade	Intervalo Tempo de Interação	Uso da Habilidade	Aprende Sobre Seq. ordenada	Habilidade Prévia
			Forget Curve Model (LFCM)													
Turner (1996)	46	Comparativo de 5 Algoritmos	Bush and Mosteller	Gerados	D	N	N	N	S	S	N	N	N	N	N	N
Jaber e Bonney (1997)	91	Comparativo de Algoritmos	Variable Regression to Invariant Forgetting (VRIF), Variable Regression to Variable Forgetting (VRVF), and Learn-Forget Curve Model (LFCM)	Gerados	E	N	N	N	S	S	S	S	N	N	N	N
Nembhard e Uzumeri (2000)	99	Alocação de Recurso	Extensão do algoritmo de Atividades Envolvidas (Mazur and Hastie, 1978; Uzumeri and Nembhard, 1998).	Ambos	E	S	N	N	S	S	S	S	S	S	S	N
Shafer *et al.* (2001)	138	Simular Linha de Produção	Uzumeri and Nembhard - Algoritmo hiperbólico com 3 parâmetros - Recency model RC	Indústria de Rádios	E	S	N	S	S	S	S	S	S	N	S	S
Lam *et al.* (2001)	48	Avaliar Impacto no Tempo e Produtividade	LFCM	Indústria Construção	E	N	N	N	S	S	S	S	S	N	S	N

Quadro 3. Artigos Learn e Forget analisados

Estudo	Citações	Algoritmo	Geral	Dificuldade					Aprendizado		Esquecimento				Habilidade	
		Objetivo	Algoritmo	Dados de teste	Tempo	Habilidade Usuário	Nível Item	Nível Habilidade	Ganha	Perde	Intervalo Tempo do item	Intervalo Tempo da Habilidade	Intervalo Tempo de Interação	Uso da Habilidade	Aprende Sobre Seq. ordenada	Habilidade Prévia
Jaber e Bonney (2003)	102	Maximizar Qualidade e impacto no custo	Wright e Loftus (esquecimento) - Learn–Forget Curve Model (LFCM)	Gerados	E	N	N	N	S	S	S	S	N	N	N	N
Jaber *et al.* (2003)	59	Minimizar Tempo de Esqueci Mento	LFCM	Gerados	E	N	N	S	S	S	S	S	S	N	S	N
Jaber e Sikström (2004)	64	Minimizar Tempo	Learn–Forget Curve Model (LFCM), The Recency Model (RC), and The Power Integration Diffusion (PID)	Gerados	E	N	N	S	S	S	S	S	S	N	S	N
Stratman *ET AL.* (2004)	41	Análise de Custo	Bailey e Globerson	Manufatura	D	N	N	N	S	S	N	N	N	N	N	N
Allwood e Lee (2004)	37	Melhoria Linha Produção	Aprendizado Yelle (1979) e esquecimento McCreery and Krajewski (1999)	Gerados	E	N	N	N	S	S	S	S	N	N	S	N
Sayin e Karabati (2007)	55	Maximizar Alocação de Usuários	Curva de aprendizado e esquecimento baseado em Nembhard and Uzumeri (2000b)	Gerados	E	N	N	N	S	S	S	S	S	N	N	S
Heimerl e Kolisch (2010)	61	Minimizar Tempo	Adapatação de Nembhard and Uzumeri (2000), Pendharkar and Subramanian (2007)	Gerados	E	S	S	N	S	S	S	S	S	S	S	N

Quadro 3. Artigos Learn e Forget analisados

Estudo	Citações	Algoritmo	Geral	Dificuldade					Aprendizado		Esquecimento				Habilidade	
		Objetivo	Algoritmo	Dados de teste	Tempo	Habilidade Usuário	Nível Item	Nível Habilidade	Ganha	Perde	Intervalo Tempo do item	Intervalo Tempo da Habilidade	Intervalo Tempo de Interação	Uso da Habilidade	Aprende Sobre Seq. ordenada	Habilidade Prévia
			para curva de aprendizado e Gutjahr *et al.* (2008) para alocação de recursos													
Choffin *et al.* (2019)	7	Aumentar Aprendizado	DAS3H	Bases Educacionais	E	S	S	S	S	S	S	S	S	S	S	S
Guo *et al.* (2019)	5	Minimizar Tempo e Custo	Firework Algorithm baseado em WLC learning (Wright 1936).) and VRVF forgetting models, Carlson and Rowe (1976)	Gerados	E	S	N	N	S	S	S	S	N	S	S	N
Kim *et al.* (2019)	4	Padronizar Avaliação da Curva de Aprendizado	Learning (log-linear, accumulation, and replacement functions.)	Gerados	E	S	N	S	S	S	S	S	S	N	S	S
Cavagnini *et al.* (2020)	3	Reduzir Custos	Adapatado de Hewitt *et al.*, 2015)	Gerados	E	N	N	N	S	S	N	S	S	N	N	N
Gan *et al.* (2020)	1	Avaliar Nível de Aprendizado	Learning baseado no estudo 38 Forgetting Chen Y, Liu Q, Huang Z,Wu L, Chen E,Wu R, Su Y, Hu G (2017) e	Bases educacionais	E	N	N	N	S	S	S	N	N	N	N	N

Quadro 3. Artigos Learn e Forget analisados

Estudo	Citações	Algoritmo	Geral	Dificuldade					Aprendizado		Esquecimento				Habilidade	
		Objetivo	Algoritmo	Dados de teste	Tempo	Habilidade Usuário	Nível Item	Nível Habilidade	Ganha	Perde	Intervalo Tempo do item	Intervalo Tempo da Habilidade	Intervalo Tempo de Interação	Uso da Habilidade	Aprende Sobre Seq. ordenada	Habilidade Prévia
			Settles B, Meeder B (2016)													
Hoedt *et al.* (2020)	0	Predizer a produção em Tempo Real com Impacto de Interrupções	MLFCM (modified learn-forget curve model)	Fábrica De Cadeirinha de Criança	E	N	N	N	S	S	S	N	N	N	S	N
Jaber *et al.* (2021)	0	Criar Novo Algoritmo para Curva de Aprendizado	MWLC e AMWLC	Gerados	E	S	N	S	S	S	S	S	S	N	S	S
Gräßler *et al.* (2021)	0	Predizer Performance Linha Baseado em fatores Humanos	Inclusão de fatores humanos adaptando o algoritmo de Jaber e Bonney	Gerados	E	S	S	S	S	S	S	S	S	S	N	S

Fonte: Autor

Shtub *et al.* (1993) foi o primeiro artigo analisado. Os autores utilizaram as curvas de aprendizado e esquecimento baseados no algoritmo de Bailey e Globerson para avaliar o impacto das curvas em uma linha de produção de uma indústria altamente tecnológica controlada por computadores considerando os impactos de interrupção da produção.

As curvas de aprendizado e esquecimento foram utilizadas por Li e Cheng, (1994); Badiru (1995); e Dar-El *et al.* (1995) para avaliar os impactos da interrupção em uma linha de produção, seja de custo, qualidade dos produtos produzidos ou melhorar a predição da produção. Foram os primeiros trabalhos a analisar as variações que ocorrem na linha de produção utilizando a alternância entre o aprendizado e o esquecimento, comparando com as interrupções. Cada um utilizou uma abordagem diferente, como curvas multivariadas ao invés de curvas invariáveis utilizadas na literatura até então, aprimorando os algoritmos existentes até então.

Para encontrar algoritmos que simulem o comportamento de predadores na natureza, Turner (1996) realizaram um comparativo de 5 algoritmos baseados em algoritmos da curva de aprendizado e esquecimento, utilizando uma tabela de comparativo entre 5 curvas de aprendizado e 7 de esquecimento encontrando diferenças significativas entre o aprendizado de longa duração e o de curta duração.

Com o intuito de minimizar os custos e maximizar a produção, Jaber e Bonney (1996), adaptaram a curva de aprendizado e esquecimento (*Learn–Forget Curve Model* - LFCM) e utilizaram bases educacionais para validação do algoritmo proposto. Este é um dos primeiros estudos acrescentando a curva de esquecimento ao algoritmo, com uma adaptação das curvas existentes, demonstrando o impacto no aprendizado com o acréscimo das quebras ao longo do tempo. Concluíram que o número de pessoas impacta diretamente no resultado.

Jaber é um autor com diversos estudos publicados na área entre os mais citados e que ainda permanece publicando, com os estudos: Jaber e Bonney (2003); Jaber e Bonney (1997); Jaber e Sikström (2004); Jaber *et al.* (2003); Jaber *et al.* (2021). Este autor propôs o algoritmo LFCM e em seus estudos, propôs novos algoritmos para a curva de aprendizado e esquecimento como os algoritmos: Regressão variável para Esquecimento Invariante (*Variable Regression to Invariant Forgetting* – VRIF*)*, Regressão variável para Esquecimento Variável (*Variable*

Regression to Variable Forgetting - VRVF); Curva de aprendizado de Wright modificada (*Modified Wright Learning Curve* – MWLC) e Curva aproximada de aprendizado de Wright modificada (*Approximate Modified Wright Learning Curve* - AMWLC*),* com os mais variados objetivos, como maximizar qualidade da produção e minimizar o impacto do esquecimento.

Os autores Nembhard e Uzumeri (2000) propuseram um algoritmo de curva de aprendizado e esquecimento que é utilizado como algoritmo base para diversos outros autores em seus estudos. Neste estudo em particular, o objetivo é alocação de recursos na indústria têxtil para tarefa repetitivas e excepcionais, com o algoritmo baseado em outros autores e no deles. Concluíram que o recurso que aprende rápido, esquecem igualmente rápido.

Os autores Shafer *et al.* (2001) tinham por objetivo simular uma linha de produção utilizando um algoritmo hiperbólico baseado em Uzumeri e Nembhard (2000). A base do estudo deles foi uma indústria de montagem de rádios com um levantamento dos padrões de aprendizado dos recursos, ao invés de se utilizar uma curva padrão entre todo, que demonstrou que a curva com dados reais tende a ser melhor que a padrão para estimativa do tempo de produção. Um outro dado encontrado pelos autores, é que quanto maior o nível do recurso, maior a produtividade dele, e menor influência do fator de aprendizado neste recurso.

Lam *et al.* (2001) utilizaram a LFCM para avaliar o tempo e produtividade numa indústria de construção, encontrando curvas mais realistas quando verificam que o conceito de esquecimento pode ser aplicado como interrupção na construção gerando perdas.

Stratman *et al.* (2004) analisaram o impacto no custo na produção considerando os impactos das interrupções entre as repetições na modelagem da curva de esquecimento. Comprovando uma teoria que, é possível ter uma correlação entre os períodos de interrupção e a curva de esquecimento.

Allwood e Lee (2004) analisaram a linha de produção com viés ao Lean verificando habilidades gerais e específicas, utilizando as curvas de aprendizado e esquecimento para reduzir o impacto na linha de produção. Também analisaram o impacto da rotação dos funcionários dentro da empresa comparando com o impacto

na especialização dentro linha de produção. Concluíram que, quando a linha de aprendizado é mais acelerada, a curva de esquecimento é menor.

Um outro estudo baseado no algoritmo de Nembhard e Uzumeri é o de Sayin e Karabati (2007) que busca a maximização da alocação de recursos em treinamentos com vistas a maximizar o retorno sobre o investimento para a empresa. O algoritmo usa dois estágios, uma para a curva de aprendizado e outro para alocação dos recursos em treinamentos.

Heimerl e Kolisch (2010) foram autores que utilizaram o algoritmo de Nembhard e Uzumeri e outros para minimizar o tempo total de projeto no problema MSRCPSP. Encontraram que as empresas ampliam seus conhecimentos com o aprendizado dos recursos, principalmente quando este aprendizado se dá de forma rápida.

Um dos estudos mais citados em nosso levantamento é o de Anzanello e Fogliatto (2011) que realizou uma revisão da literatura sobre curva de aprendizado. Porém, como não se trata de uma aplicação direta das curvas, não foi listado no Quadro 3.

Dentre os artigos mais novos, destacam-se alguns estudos voltados para a área de psicologia e ganho de conhecimento através do estudos como os trabalhos de Choffin *et al.* (2019) e Gan *et al.* (2020). Estes trabalhos propõem novos algoritmos para a curva de aprendizado e esquecimento para a área de ensino, mas que também podem ser utilizados em linhas de produção ou em projetos.

Nos trabalhos analisados, as curvas de aprendizado e esquecimento são amplamente utilizadas em linhas de produção e projetos, por permitirem uma modelagem dos impactos ou mesmo, possibilitar uma previsibilidade de como será o desenvolvimento do novo produto. Devido a esta características, os autores continuam propondo novos algoritmos para aumentar a acurácia e previsibilidade, como os trabalhos de Guo *et al.* (2019); Kim *et al.* (2019); Cavagnini *et al.* (2020); e Hoedt *et al.* (2020).

Um algoritmo diferente foi proposto por Gräßler *et al.* (2021) ao acrescentar fatores humanos e experiência nas curvas de aprendizado e esquecimento, tentando se aproximar ainda mais de um ambiente real e que, segundo os autores, ficou próximo ao ambiente fabril analisado.

Grande parte dos algoritmos utilizaram tempos estocásticos para avaliação das curvas, com somente 3 utilizando tempos determinísticos. Apenas 8 estudos levaram em conta as habilidades dos usuários na criação do algoritmo, sendo que 7 ainda acrescentaram o nível destas habilidades e apenas 3 o nível do item a ser produzido ou a tarefa executada.

Pela característica deste estudo, todos os artigos apresentaram as curvas de aprendizado e esquecimento. Na análise da curva de esquecimento, todos consideraram o impacto no intervalo de tempo entre cada tarefa, e praticamente todos avaliaram o impacto do intervalo de tempo na habilidade e 12 dos 25 artigos avaliaram o intervalo de tempo entre cada interação dos ciclos.

Por fim, na avaliação das habilidades, 5 artigos abordaram o uso da habilidade, somente 8 não verificaram o impacto do aprendizado sobre a sequência na ordenação das tarefas e somente 6 verificaram qual o impacto do recurso possuir uma habilidade prévia.

Na sequência da revisão, será abordado o algoritmo *Ant Colony Optimization* que será utilizado para resolução dos problemas integrados discutidos anteriormente, que foi identificado na literatura dos tópicos abordados anteriormente como uma solução factível e com produção de bons resultados.

2.5 Ant Colony Optimization

A heurística de Otimização Colônia de Formigas (*Ant Colony Optimization - ACO*) foi proposta em 1992 por Marco Dorigo em sua tese de doutorado. A primeira publicação foi realizada por Dorigo *et al.* (1996) com o descritivo da formulação matemática, um comparativo entre o ACO com as metaheurísticas Busca Tabú e SA, demonstrando uma melhor performance para solução de problemas do tipo Caixeiro Viajante, Problema de Atribuição Quadrático e JSP. O algoritmo foi proposto através do mimetismo de como as formigas utilizam um algoritmo para sair do ponto A e chegar até um ponto B tendo algumas opções de caminho, e acabam por escolher os menores caminhos para a rota com a utilização dos feromônios que são intensificados nos menores caminhos, e que vão se reduzindo ao longo do tempo quando deixam de ser utilizados.

É um algoritmo que possibilita amplas aplicações e utilizações, como as já citadas para problemas de agendamento como: JSP; RCMPSP; MSRCPSP e outros.

Problemas de atribuição como: Atribuição quadrática (*Quadratic Assignment Problem - QAP*); Atribuição Generalizada (*Generalized Assignment Problem* - GAP); Atribuição de Frequência (*Frequency Assignment Problem* - FAP) e outros. Problema de roteamento de veículos como: Roteamento Estocástico de Veículos (*Stochastic Vehicle Routing Problem* - SVRP); Roteamento de Veículos Multi Depósito (*Multi-Depot Vehicle Routing Problem* - MDVRP); Roteamento de Veículos por Período (*Period Vehicle Routing Problem* - PVRP) e outros. É utilizada ainda em processamento de imagens, otimização e síntese de antenas, otimização de tamanho de nano eletrônicos e muitas outras.

Foi realizado um levantamento nas bases *Scopus* e WoS nos dias 08/09/2021 e 09/09/2021 respectivamente para os termos "ACO" ou "*Ant"* e "*Colony"* e "*Optimization"* em Título, Resumo ou Palavras-Chave e adicionando os termos "*Model"* e "*Schedul*" e "*Problem*" também em Título, Resumo ou Palavras-Chave, encontrando 818 artigos na base Scopus e 641 na WoS. Ao filtrar pelo número de citações, foram encontrados 17 com mais de 100 citações. Filtrando por ano de publicação, considerando apenas os artigos com ano de publicação superior ou igual a 2019, foram encontrados 236 artigos, ou seja, é um tema ainda atual e com grande relevância. Para análise dos artigos mais novos, optou-se por verificar os artigos com mais de 15 citações neste período, com um total de 21 artigos.

Nos primeiros artigos publicados, os autores começaram a explorar as possibilidades de utilização do algoritmo em aplicações, como pode-se verificar em Gagné *et al.* (2002); Shyu *et al.* (2004); Xing *et al.* (2009); e Ferrandi *et al.* (2010) que exploraram a utilização do ACO para o agendamento das tarefas e comunicações, com linhas simples ou compostas. Outros estudos evoluíram para a solução de roteamento de veículos como Rossi e Dini (2007); Lin *et al.* (2014); e Huang e Lin (2010) Publicações mais recentes, exploram evoluções do algoritmo original para melhoria de performance do algoritmo para redução de tempo e aumento do número de soluções ótimas, conforme podemos ver nos próximos artigos.

Para resolução do problema de agendamento flexível JSP (*Flexible Job Shop Scheduling Problem* - FJSSP) por Xing *et al.* (2010) os autores propuseram um algoritmo híbrido do ACO integrado de forma efetiva com um algoritmo baseado em conhecimento (*Knowledge-Based Ant Colony Optimization* - KBACO). O algoritmo de aprende com o conhecimento gerado pelo algoritmo ACO e então aplica este

conhecimento com a finalidade de guiar a busca da heurística, atuando de forma iterativa. Nos testes, o KBACO demonstrou ser melhor sobre algumas abordagens em termos de qualidade do agendamento.

O algoritmo baseado na otimização multiobjetivo de colônia de formigas (*Multi-Objective Ant Colony Optimization* – MOACO) foi proposto para aumentar a qualidade das soluções para a produção e manutenção de máquinas em sistemas de produção. Segundo os autores Berrichi *et al.* (2010), os testes comparativos demonstraram a performance sobre dois AGs específicos para este tipo de solução pois utilizam múltiplas matrizes de feromônios e matrizes de heurísticas. Cada formiga é alocada com pesos diferentes para agregar os diferentes feromônios para as soluções não dominadas.

Chen e Zhang (2013) propuseram um algoritmo híbrido para a alocação de recursos em projetos de software. O algoritmo utiliza o agendamento baseado em eventos (*Event-Based Scheduler* - EBS) para realizar a alocação dos recursos evitando os conflitos de alocação, a preempção entre as tarefas e preserva a alocação flexível dos recursos e utilizo o ACO para o problema de agendamento. Conforme os autores, o algoritmo híbrido se mostrou muito promissor para a solução deste problema.

Uma solução em dois estágios para minimizar o tempo total do algoritmo de colônia de formigas (*Ant Colony Algorithm* - ACA) foi proposto para resolução do problema integrado do JSSP e do problema de conflito de rotas livres (*Conflict-Free Routing Problem - CFRP*) por (LIN *et al.*, 2014). O primeiro estágio foi utilizado para definir as rotas possíveis de solução e o segundo estágio otimiza essas rotas.

A implementação na nuvem de um algoritmo ACO melhorado para resolução do problema de agendamento de tarefas foi o algoritmo proposto pelos autores Zuo *et al.* (2015). O algoritmo apresenta dois objetivos, que são minimizar o tempo e o custo, com a reutilização de recursos e com taxas de violação do tempo limite com a avaliação da qualidade da solução e com a provisão de retroalimentação do sistema, demonstrando através de testes ser melhor que outros algoritmos similares, especialmente com um acréscimo de 56% de cenários ótimos.

Deng *et al.* (2019) propuseram uma série de melhorias no algoritmo ACO denominado *Multi-Population Co-Evolution Ant Colony Optimization* (ICMPACO).

Neste novo algoritmo, os autores fizeram uma decomposição do problema principal em vários subproblemas, além da divisão das formigas em duas categorias diferentes, com as formigas de elite e as formigas comuns, evitando-se assim os ótimos locais com a melhora das taxas de convergência. Para evitar de os feromônios serem liberados em pontos adjacentes por uma série de regiões, foi implantado um mecanismo de coevolução para a troca de informações entre as diferentes subpopulações geradas melhorando o desempenho da otimização, com a verificação via problema do caixeiro viajante e o problema de atribuição de portão.

Artigos mais recentes estão propondo a utilização do ACO para solução de problemas da nuvem, devido a este ambiente ser cobrado pela utilização de recursos. Autores como Chen *et al.* (2019); Li *et al.* (2019); Lin *et al.* (2019); Wei (2020); e Jia *et al.* (2021), utilizando algoritmos de múltiplas colônias (MACO) que utilizam múltiplas matrizes para controlar as colônias e formigas para resolução do problema, utilizando a implementação em nuvem ou não.

As múltiplas colônias de formigas, junto com uma nova regra para atualização dos feromônios baseado em um conjunto de soluções não dominadas para guiar cada colônia, com balanceamento entre os múltiplos objetivos e a utilização de formigas para melhorar a qualidade da solução global foi o algoritmo proposto por Chen *et al.* (2019) para a solução da nuvem. Li *et al.* (2019) propuseram dois níveis de ACO para a resolução do problema, de forma que o nível mais baixo, realiza o agendamento das tarefas de forma unitária, e o nível superior, controla cada um dos níveis mais baixos. Li e Zhang *et al.* (2019) compararam os algoritmos MACO e NSGA-II, comprovando sua eficiência.

O algoritmo híbrido do ACO com *Fuzzy* foi o algoritmo proposto por Jia *et al.* (2019) para máquinas em paralelo com diferentes capacidades. O algoritmo prevê uma lista de tarefas com dois candidatos de um espaço de soluções ainda não ocupados para guiar as formigas. O algoritmo *fuzzy* é utilizado para aumentar a qualidade das soluções locais. Ganji *et al.* (2020) utilizaram três metaheurísticas para resolução do problema, com o PSO (*Particle Swarm Optimization*), NSGA-II e MOACO, sendo que o NSGA-II foi usado no nível mais alto para controlar as demais metaheurísticas. O PSO é utilizado para cálculo das distâncias e pôr fim a MOACO para cálculo da camada mais baixa do algoritmo, com os cálculos individuais. Por fim, Zheng *et al.* (2020) evoluíram o MOACO utilizando um algoritmo híbrido denominado

MHACO (*Multi-Objective Hybrid Ant Colony Optimization*), com um algoritmo de regras de restrição de feromônio máximo–mínimo e as regras de busca local para evitar a armadilha do ótimo local.

Alguns estudos novos estão propondo algoritmos baseados em ACO para solução de problemas específicos. Rubaiee e Yildirim (2019) propuseram um ACO baseado na dominância por ranking ou por dominância por ranking e comparação com aglomeração de distância (Ant Colony Optimization based on a Dominance Ranking - ACODR ou *Based on a Dominance Ranking procedure and Crowding Distance Comparison* - ACO-DRC). Hussein *et al.* (2019) propuseram um algoritmo ACO alteraram o algoritmo para ser baseado no fitness da função (*Ant Colony Optimization based on Best Fit* - ACO-BF), considerando o melhor fitness e o maior fitness da função sempre avaliando os recursos e máquinas restantes.

Como pode ser verificado, o ACO é utilizado de forma ampla na resolução dos mais variados problemas da literatura. Nos artigos mais recentes, pode-se verificar que o algoritmo tem evoluído para algoritmos híbridos de solução para resolução dos problemas, indicando um caminho para obter a melhor solução para os mais diversos problemas, e em específico, a junção do RCMPSP e MSRCPSP.

Na próxima seção será apresentado o referencial teórico para os problemas abordados na revisão da literatura.

3 Referencial Teórico

Neste capítulo é apresentado o referencial teórico relativo aos temas abordados nesta tese e que são de fundamental importância para a compreensão do trabalho desenvolvido.

3.1 Problema de escalonamento de tarefas

O problema de alocação de recursos em tarefas é antigo, tendo uma série de estudos a este respeito, com origem no problema de escalonamento de tarefas, ou mais comumente conhecido como *Job Shop Problem* (JSP), sendo que este problema possui diversas variações.

Segundo Morales e Ronconi (2015), para resolução do problema JSP, as principais suposições das tarefas são:

- Cada tarefa é determinada no início do processo de escalonamento e deve estar disponível a todo momento;
- Existe uma relação de precedência entre um par de tarefas, porém não existe entre tarefas sem relação direta;
- O roteiro das tarefas é definido pela sequência das operações e recursos necessários;
- Cada tarefa possui um tempo determinístico e contínuo, incluindo os tempos de transporte e preparação;
- Não há data limite para o término;
- Todas as tarefas devem ser realizadas por todos os recursos.

Para a formulação matemática, conforme Morales e Ronconi (2015), as suposições sobre os recursos, que podem ser máquinas, recursos humanos ou qualquer outro recurso utilizado nos processos, são:

- Os recursos estão disponíveis durante todo o processo de forma contínua;
- Cada recurso é capaz de realizar uma tarefa por vez;
- Ao iniciar um processo ou uma operação, ela não será interrompida;
- A fila antes do início do processo, ou após o término não possui limite.

3.2 Problema de alocação de recursos limitados em projeto

Sprecher, Kolisch e Drexl (1995) afirmam que uma generalização do JSP, é quando existe uma restrição de recursos para resolução do problema de alocação de recursos em um projeto ou *Resource Constrained Project Scheduling Problem* (RCPSP). Conforme Hosseinian e Baradaran (2020), o RCPSP é um problema que consiste em escalonar tarefas de um projeto que possuem vínculos por relação de precedência entre si e que requer recursos específicos e disponíveis em quantidades finitas. Possui o objetivo de encontrar uma solução que seja o menor tempo possível de execução total (Solução ótima) do projeto respeitando as restrições de precedência e dos recursos. A solução ótima pode ser obtida por algoritmos exatos ou heurísticos, enquanto os algoritmos heurísticos podem também encontrar as soluções subótimas também. O RCPSP pode ser representado através de um grafo ou pelo gráfico de Gantt.

Assim como o JSP, o RCPSP possui outras variações ou generalizações, como para a alocação dos recursos considerando suas habilidades, quantidade de projetos simultâneos, quantidade de recursos disponíveis no sistema e outros. Houve evoluções ao longo do tempo do JSP, e com o avanço tecnológico dos recursos de TI disponíveis, possibilitando os cálculos necessários para atender estas variações.

3.3 Problema de programação para alocação de recursos limitados em múltiplos projetos

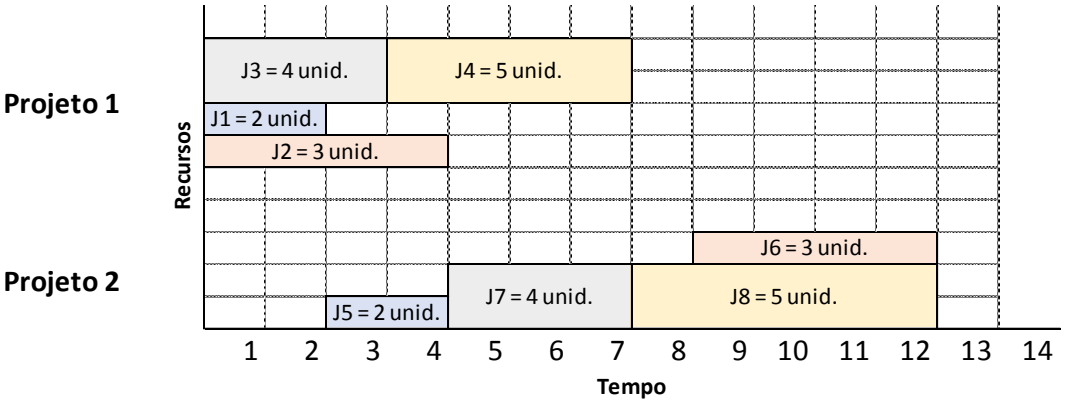
O problema para alocação de recursos limitados em múltiplos projetos ou *Resource Constraint Multi Project Scheduling Problem* (RCMPSP) é uma das generalizações do problema RCPSP com foco em alocar recursos limitados em múltiplos projetos Villafáñez *et al.* (2019b), sendo que os múltiplos projetos precisam ser escalonados, pois possui influência na alocação dos recursos e no escalonamento das tarefas no projeto, com impacto no tempo total de execução do projeto. Nos subtópicos a seguir, serão detalhados a representação gráfica e matemática do RCMPSP.

3.3.1 Representação gráfica do RCMPSP

O RCMPSP pode ser representado pelo gráfico de Gantt. No gráfico de Gantt representado na Figura 2 são representados dois projetos que são executados de simultaneamente, com as suas tarefas representadas pela letra J seguida do número da tarefa, com as tarefas de 1 a 4 sendo do projeto 1, e de 5 a 8 do projeto 1. As

premissas do tópico anterior foram respeitadas, não havendo concorrência entre os recursos, sem prioridade das tarefas individuais, não havendo interrupção de um processo para início de outra tarefa e não tendo limite na fila antes ou após o término do processo.

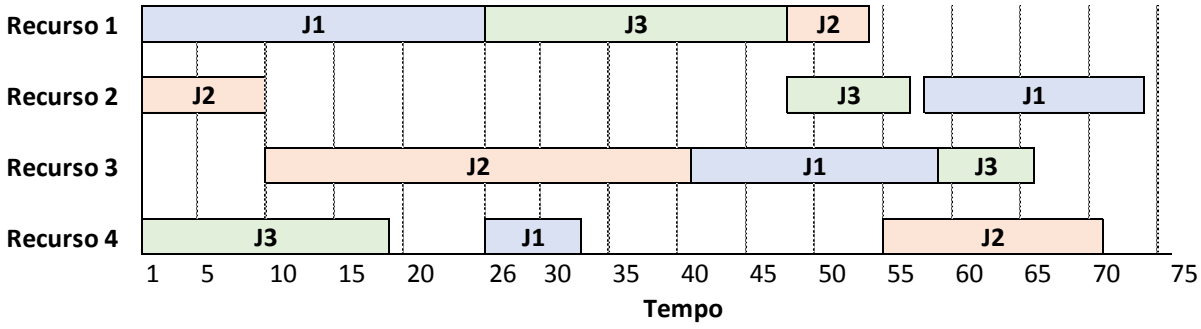
Figura 2. Representação gráfica do RCMPSP



Fonte: Zheng *et al.* (2013)

Com uma quantidade pequena de recursos, e um número baixo de projetos simultâneos, é possível ainda alocar e otimizar a alocação dos recursos de forma manual, porém, conforme se aumenta o número de recursos, e a quantidade de projetos simultâneos, a complexidade da alocação dos recursos aumenta, conforme pode-se verificar na Figura 3, que com uma quantidade maior de recursos disponíveis (4 recursos), mesmo com uma diminuição da quantidade de projetos (3 projetos, representados de J1 a J3) e com cada projeto possuindo diversas tarefas (cada quadrado representa uma tarefa), as possibilidades de alocação dos recursos são muito maiores, o que torna cada vez mais difícil a alocação dos recursos de forma manual, com a otimização destes recursos na alocação dos projetos, de forma otimizada, com tempo total dos projetos de forma reduzida.

Figura 3. Representação gráfica do RCMPSP com aumento de recursos



Fonte: Morales (2012)

Como uma medida de tempo e eficiência do projeto, uma das métricas mais utilizadas é o tempo total necessário para a realização de todas as suas tarefas, denominado de *Makespan*. Segundo Morales e Ronconi (2015), o *makespan* representa a diferença de tempo entre o início da primeira tarefa do projeto e o término da última tarefa do projeto e é obtido através do caminho crítico do projeto. No caso de múltiplos projetos, considera-se para o *makespan* o início da tarefa do primeiro projeto, até o término da última tarefa do último projeto a ser executado. Por exemplo, na Figura 2 o *makespan* é de 12 unidades de tempo e, na Figura 3 de 73 unidades de tempo.

3.3.2 Formulação matemática do RCMPSP

Com a finalidade de analisar o problema RCMPSP, que é um problema combinatório, Morales e Ronconi (2015) fizeram um estudo para avaliar diversas formulações que podem resolver o problema. Os autores concluíram que associando os algoritmos com estratégias de melhoramento apresentam melhores resultados, de forma que encontram um número significativo de soluções ótimas.

Morales (2012) identificou que existem dois grandes grupos dentre as alternativas de formulações matemáticas quando considerado o horizonte do tempo, que são as formulações com horizonte de tempo discreto e contínuo, que respectivamente consideram o tempo em períodos ou de forma contínua.

A formulação de tempo contínua apresenta dois grandes grupos, segundo Morales (2012), que são as formulações de precedência ou do tipo designação. Ambas formulações possuem expoentes da década de 60, sendo que a de precedência engloba os algoritmo de Manne (1960) e uma adaptação do mesmo por Liao e You (1992), se caracterizando por restrições disjuntivas que indicam a precedência entre as tarefas, enquanto que a de designação engloba o algoritmo de Wagner (1959) com adaptação de Wilson (1989), que as máquinas ou tarefas são divididas em posições e cada operação é estabelecida em uma única posição.

No APÊNDICE I apresenta-se a formulação matemática do RCMPSP nas tabelas: 5, 6, 7 e 8, e que foi adaptado de Morales (2012).

Ainda segundo Morales (2012), a formulação do problema apresenta quatro particularidades que podem ser consideradas como:

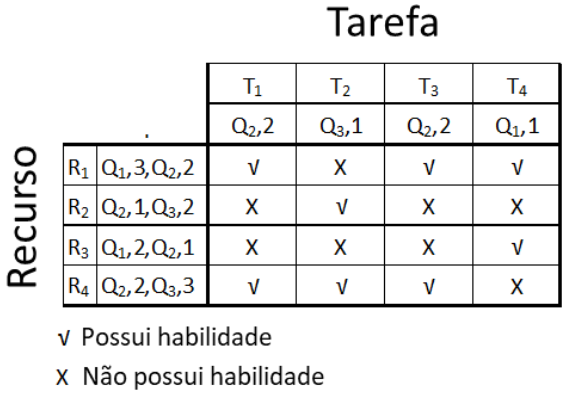
1. Designação das tarefas (cada tarefa desenvolvida por cada recurso) nas posições (representadas pelos recursos).
2. Definição do instante de início do projeto.
3. Existência de tempo ocioso entre posições consecutivas em cada tarefa.
4. Definição do instante de cada tarefa em cada posição na tarefa.

3.4 Problema de programação para alocação de recursos limitados com múltiplas habilidades em projeto

A alocação de recursos limitados com múltiplas habilidades ou *Multi Skill Resource Constraint Project Scheduling Problem* MSRCPSP ou MS-RCPSP, bem como o problema RCMPSP, é uma generalização do problema RCPSP, com a diferença que este problema busca uma solução para a alocação de recursos que possuem diversas habilidades nos projetos (MYSZKOWSKI; SKOWROŃSKI; OLECH; OŚLIZŁO, 2015).

O MSRCPSP consiste na alocação de K tipos de recursos ($R_1, R_2, \ldots, R_K$) e N habilidades diferentes ($Q_1, Q_2, \ldots, Q_N$) em uma tarefa j ($j=1,2,\ldots,J$), que são partes que compõe um projeto (T_j) e com duração d_j sendo que cada recurso possui uma série de habilidades com determinado nível de complexidade. Cada tarefa possui uma restrição de precedência entre pares de tarefas que significa que uma tarefa não pode ser iniciada até que todos as precedentes não tenham sido finalizadas. Além desta restrição, uma tarefa *T* só pode ser executada por recursos *R* que possuem um mínimo de habilidades *Q* necessárias, e que foi definida como necessária para conclusão da tarefa, como pode ser observado na Figura 4.

Figura 4. Alocação dos recursos em tarefas conforme habilidades



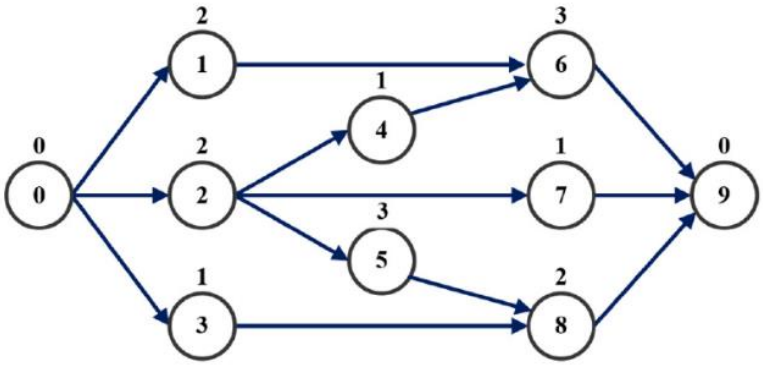
Fonte: Wang e Zheng (2018)

Cada recurso que possui a habilidade necessária para execução da tarefa possui um valor fixo de remuneração, por exemplo Q_2,2 representa que a habilidade Q_2, possui remuneração 2, determinando os custos totais do projeto. Assim como o RCMPSP, o MSRCPSP pode possuir diversos objetivos diferentes como de redução do *Makespan*, ou do custo, ou ambos (ZHENG *et al.*, 2015).

3.4.1 Representação gráfica do Problema MSRCPSP

O MSRCPSP também pode ser representado pelo gráfico de Gantt. Para melhor explicação, dado um projeto com 8 tarefas numeradas de 1 a 8 com o 0 representando o início e o 9 o final do projeto no interior de cada círculo, com os números na parte superior externa de cada círculo representando o esforço para execução da tarefa e as setas indicando a relação de precedência das tarefas, conforme a Figura 5, verifica-se a relação de tarefas escalonados e pode-se obter o tempo total de execução do projeto.

Figura 5. Representação gráfica da rede do projeto

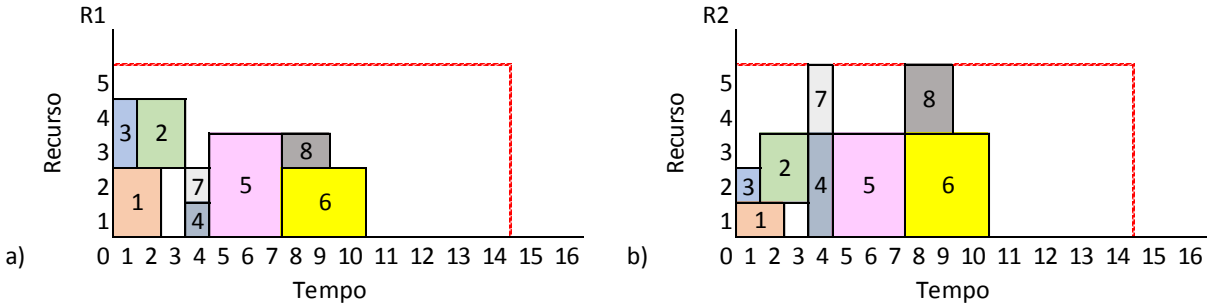


Fonte: Hosseinian, Baradaran e Bashiri (2019)

A alocação dos recursos no início do projeto pode ser realizada sem que se considere as habilidades destes (que poderão gerar soluções não factíveis), gerando um gráfico de Gantt conforme a Figura 6 que demonstra o tempo de execução do projeto. O tempo de execução pode variar conforme o número de recursos alocados, ou conforme a disponibilidade dos recursos. No gráfico da Figura 6(a), foi considerada a alocação de recursos R_1 com capacidade de atendimento simultâneo com 5 recursos disponíveis (definido pela linha vermelha) para atender as 8 tarefas do projeto da Figura 5. Na Figura 6(b), foi considerada a alocação de recursos R_2 com a mesma capacidade de atendimento simultâneo, ou seja, igual a 5 recursos também (definido pela linha vermelha) para atender as 8 mesmas tarefas do projeto da Figura

5. O tempo de execução total foi semelhante, porém com duas soluções possíveis diferentes. Como exemplo a tarefa 3, que o esforço de execução é 1, na Figura 6(a) é executado em 0,5 unidade de tempo e por dois recursos e Figura 6(b) é executado por 1 recurso em um unidade de tempo.

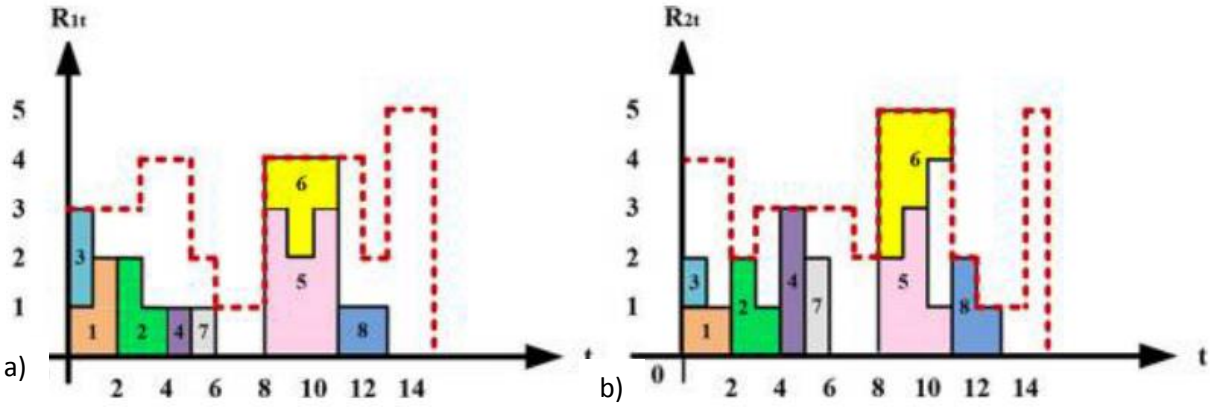
Figura 6. Alocação de 1 recurso e de dois recursos respectivamente no projeto da Figura 5 sem considerar habilidade deste



Fonte: Hosseinian, Baradaran e Bashiri (2019)

Ao se alocar os recursos no projeto que demandam habilidades específicas destes, como o definido no problema MSRCPSP, existe a possibilidade de aumento do tempo total do projeto como pode-se verificar na Figura 7, que passará a contar com capacidade variável devido a necessidade de contar com o recurso correto no momento necessário. A linha vermelha não é mais constante em cinco recursos, podendo variar conforme os recursos com determinada habilidade ficam disponíveis para executar uma tarefa. Na Figura 7(a), por exemplo, no intervalo de tempo entre 6 e 8, só existe um recurso disponível para executar tarefa dentre os recursos de R_{1t}. Como este recurso não possui a habilidade necessária, nenhuma tarefa é executada, e o projeto fica parado. Já no intervalo entre 0 e 3, existem 3 recursos disponíveis com a habilidade necessária para executar as tarefas do projeto, executando as tarefas 1, 3 e iniciando a 2 com a utilização de 2 recursos no intervalo de tempo entre 2 e 3, e depois reduzindo para 1 recurso no intervalo de tempo entre 3 e 4. Na Figura 7(b), os recursos R_{2t} possuem outra disponibilidade, como por exemplo o intervalo de tempo entre 6 e 8, que possui 3 recursos disponíveis entre 6 e 7, depois reduz para 2, mas de forma semelhante, estes recursos não possuem a habilidade necessária para executar tarefas neste intervalo de tempo, e o projeto também fica parado. Ainda na Figura 7(b), no intervalo de tempo entre 0 e 2, existem 4 recursos disponíveis, porém somente 2 são utilizados no intervalo entre 0 e 1, e depois somente 1 recurso passa a ser utilizado, havendo recursos sem executar tarefas.

Figura 7. Alocação de 1 recurso e de 2 recursos no projeto da Figura 5 considerando a habilidade deste



Fonte: Hosseinian, Baradaran e Bashiri (2019)

Como pode se verificar na Figura 7, a complexidade de alocação de recursos que demandam habilidades específicas afeta o tempo total do projeto devido a capacidade dos recursos disponíveis passar a ser variável. Quanto maior o número de habilidades específicas e quantidade de recursos disponíveis, maior será a complexidade de alocação destes recursos no projeto, necessitando de algoritmos mais eficientes e eficazes, além de uma quantidade maior de processamento para que o *Makespan* e o custo total do projeto sejam reduzidos.

3.4.2 Formulação matemática do MSRCPSP

Myszkowski *et al.* (2015) com a finalidade de alocação de recursos com múltiplas habilidades em projetos, realizaram alterações práticas na formulação do problema RCPSP. A primeira alteração realizada pelos autores foi a introdução do salário pago por hora, e que é pago conforme a performance do recurso. A segunda alteração diz respeito as tarefas de início e fim do projeto que foram retiradas devido ao fato de uma tarefa pode não ter relação de precedência com outra tarefa, e assim, tarefas de início e fim que são fictícias, poderiam impactar nos tempos do projeto (MYSZKOWSKI *et al.*, 2015a). Por fim, como se trata de um problema de alocação de recursos com múltiplas habilidades, e do pressuposto no problema RCPSP que não deve haver sobreposição de recursos, somente os recursos com as habilidades necessárias podem ser atribuídos as tarefas, havendo um procedimento de resolução de conflitos para impedir que recursos que não tenham a habilidade requerida possam ser alocados para execução de determinada tarefa que exija uma habilidade específica.

Ainda segundo Morales (2012), a formulação do problema apresenta quatro particularidades que podem ser consideradas como:

1. Designação das tarefas (cada tarefa desenvolvida por cada RH) nas posições (representadas pelos RH).
2. Definição do instante de início do projeto.
3. Existência de tempo ocioso entre posições consecutivas em cada projeto.
4. Definição do instante de cada tarefa em cada posição das tarefas.

No APÊNDICE I nas tabelas: 9, 10, 11 e 12 são apresentas as formulações do MSRCPSP.

3.5 Curva de aprendizado

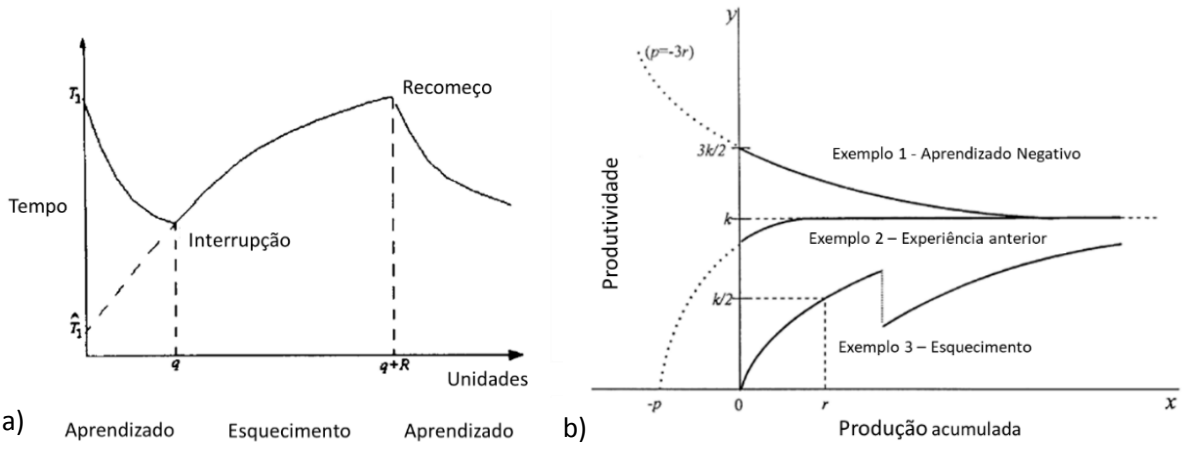
O problema de curva de aprendizado ou *Learn Forget Curve Model* (LFCM) é um algoritmo importante para análise do comportamento dos recursos e tarefas, e a representação gráfica e formulação matemática será apresentada na sequência.

3.5.1 Representação gráfica do LFCM

Existem duas representações principais para a curva de aprendizado e esquecimento, que são os algoritmos de Jaber e Bonney (1996) e Nembhard e Uzumeri (2000).

O algoritmo de Jaber e Bonney (1996) considera o aprendizado, as interrupções, que podemos considerar como a interrupção de utilização do recurso e o recomeço, conforme a Figura 8(a), que demonstra que o tempo diminui conforme o número de unidades durante o período de aprendizado. Inversamente, com a interrupção, aplica-se a curva de esquecimento, e o tempo passa a aumentar, e volta a diminuir com o recomeço da utilização. O impacto da interrupção sempre ocorre, porém com um cálculo separado para cada uma das curvas.

Figura 8. Curva de aprendizado e esquecimento de Jaber e Bonney (1996) e Nembhard e Uzumeri (2000)



Fonte: Adaptado de Jaber e Bonney (1996) e Nembhard e Uzumeri (2000)

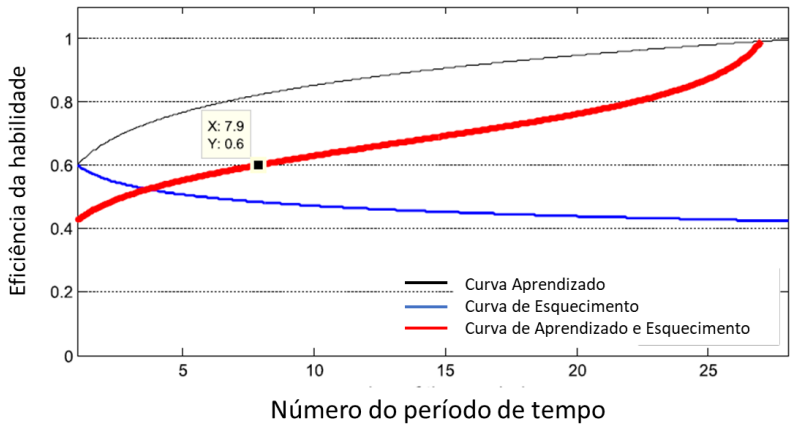
O algoritmo de Nembhard e Uzumeri (2000) considera três tipos de curvas, que foram unificadas para a modelagem. A primeira, é a curva de aprendizado negativo, de forma que a produtividade é reduzida estabilizando a produção. A segunda, é quando já existe uma experiência anterior, que encurta de forma significativa o aumento da produção, saindo de um patamar maior no momento zero. Por fim, a curva de aprendizado sendo impactada pelo esquecimento, que gera um efeito imediato na produtividade e por consequência na produção, conforme pode-se verificar na Figura 8(b).

Os gráficos da Figura 8 de curva de aprendizado e esquecimento, são exemplos de curvas utilizadas para medir a produção dos recursos. O primeiro, expressa qual a produção no tempo, respectivamente nos eixos x e y, enquanto o segundo, demonstra a produção acumulada conforme a produtividade do recurso, respectivamente nos eixos x e y. O primeiro gráfico destaca o impacto do aprendizado e esquecimento na produção, enquanto o segundo, além do destaque do aprendizado e esquecimento, inclui a experiência prévia do recurso na tarefa.

O algoritmo utilizado nesta tese será o de Chen *et al.* (2017), que utiliza um algoritmo integrado de cálculo entre o aprendizado e esquecimento ao longo do tempo, conforme a Figura 9.

.

Figura 9. Curva de aprendizado e esquecimento de Chen et al. (2017),



Fonte: Adaptado de Chen *et al.* (2017)

Assim como os demais algoritmos, a eficiência da habilidade cresce ao longo do tempo, porém, nesta curva, o esquecimento influencia um impacto direto ao longo do tempo, fazendo com que a curva de aprendizado seja menor, e com isto, a eficiência da habilidade não cresce tão rápida e o esquecimento não gera um impacto diminuindo a curva do aprendizado durante um período para então voltar a aumentar novamente, ele gera impacto constante no aprendizado.

3.5.2 Formulação matemática do LFCM

O algoritmo de Chen *et al.* (2017) apresenta uma formulação matemática unificada para as curvas de aprendizado e esquecimento, sendo demonstrada pelas tabelas: 13, 14, 15 e 16 no APÊNDICE I.

3.6 Colônia de formigas

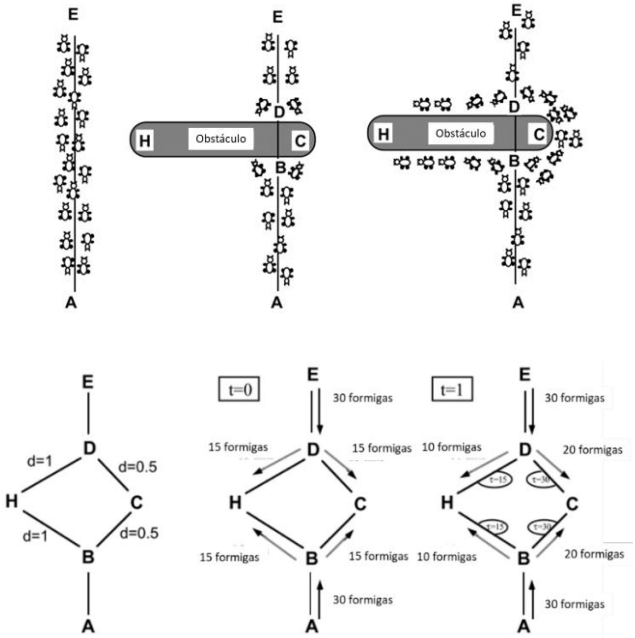
A metaheurística de Otimização Colônia de Formigas ou *Ant Colony Optimization (ACO)* foi proposto para resolver problemas que podem ser representados por caminhos ou fluxogramas, como o problema do Caixeiro Viajante com diversas evoluções e adaptações para resolução de problemas diferentes, como o RCMPSP e o MSRCPSP, entre outros.

3.6.1 Metaheurística ACO

O primeiro algoritmo de Dorigo *et al.* (1996) é baseado no mimetismo de como uma colônia de formigas resolve um problema que as formigas possuem para ir e voltar entre os pontos A e E, que representam a colônia no ponto A como ponto de partida, o alimento no ponto E, que elas procuram para alimentar a colônia, e retornando ao ponto A com este alimento. No meio deste caminho, as formigas se

deparam um obstáculo, conforme Dorigo *et al.* (1996) representaram graficamente como ocorre a distribuição das formigas ao longo do tempo, que o feromônio, que é o odor que as formigas produzem, e liberam no caminho que vão passando. E que possui uma evaporação com o tempo, ou seja, quanto mais formigas transitam por um determinado caminho, maior será a quantidade de feromônio neste caminho, e mais ele será utilizado, ocorrendo o inverso no caminho menos utilizado, pois o feromônio irá evaporar, e outras formigas deixarão de utilizar este caminho, ou seja, o feromônio possui influência no aumento das formigas utilizando determinado caminho, conforme Figura 10.

Figura 10. Mimetismo e representação gráfica colônia de formigas de Dorigo et al. (1996)



Fonte: Adaptado de Dorigo *et al.* (1996)

3.6.2 Formulação matemática do ACO

Para a resolução de um problema apenas, o algoritmo ACO vêm sendo empregado, e aperfeiçoado ao longo do tempo. Porém, foi necessário um algoritmo com outras funcionalidades para atender aos problemas desta tese, utilizando-se o algoritmo de Li *et al.* (2019) que visa resolver dois problemas de forma unificada, mas sendo empregado para o problema de agendamento de manufatura na nuvem. Nas tabelas: 17, 18 e 19 do APÊNDICE I, segue a formulação matemática.

Na próxima seção são apresentados os algoritmos e instrumentos de pesquisa desta tese, com a apresentação do algoritmo proposto.

4 Métodos e instrumentos de pesquisa

Nesta seção, são apresentados os métodos e instrumentos de pesquisa propostos, com o detalhamento do algoritmo proposto, com sua representação gráfica e formulação matemática, bem como o detalhamento do modelo de simulação construído.

4.1 Escolha e justificativa da tipologia da pesquisa

Para esta tese foi adotada a técnica de simulação computacional e como tipologia metodológica a pesquisa axiomática quantitativa. A escolha da abordagem do problema pela simulação computacional se justifica, pois a modelagem e simulação, além de possibilitar uma visualização e análise temporal do funcionamento do sistema em estudo, permite um controle dos parâmetros do ambiente e repetição dos experimentos sob diferentes restrições (HUSSEIN; MOUSA; ALQARNI, 2019), como por exemplo o tratamento estocástico dos tempos de processamento das tarefas em função do grau de habilidade do recurso alocado. Ressalta-se que os poucos trabalhos que consideram essa aleatoriedade dos tempos de processamento das tarefas o fazem com base em números *Fuzzy* (HEMATIAN *et al.*, 2020).

De acordo com os autores Miguel, Morabito e Pureza (2012), a utilização de pesquisa axiomática quantitativa permite produzir conhecimento sobre determinadas variáveis do algoritmo, tendo como referência o comportamento de outras variáveis em questão.

Bertrand e Fransoo (2002), ressaltam que a utilização da pesquisa axiomática quantitativa é recomendada em situações em que o objetivo é obter soluções para o caso estudado e quando se usa como técnica para análise de dados a simulação computacional.

Nos quadros: 4, 5 e 6, pode-se verificar um quadro resumo dos problemas estudados nesta tese.

Quadro 4. Esta tese - RCMPSP analisado

Estudo	Tarefa				Algoritmo			
	Recursos Alocados	Duração	Ordenação	Área	Objetivo	Algoritmo	Dados de teste	Tempo
Esta tese	Lim	Var	Sim	Geral	Minimizar tempo e custo	ACO em duas camadas	Gerado	E

Fonte: Autor

Quadro 5. Esta tese - MSRCPSP analisado

Estudo	Tarefa						Algoritmo			
	Recursos Alocados	Variação Proficiência Habilidade	Aprendizado	Duração	Ordenação	Área	Objetivo	Algoritmo	Dados de teste	Tempo
Esta tese	Lim	Sim	Sim	Var	Sim	Geral	Minimizar tempo e custo	ACO em duas camadas	Gerados	E

Lim: Limitados; Amb: Ambos; Var: Variável; Fix: Fixo; D: Determinístico; E: Estocástico
Fonte: Autor

Quadro 6. Esta tese - Learn e Forget analisado

Estudo	Algoritmo	Geral	Dificuldade					Aprendizado		Esquecimento				Habilidade	
	Objetivo	Algoritmo	Dados de teste	Tempo	Habilidade Recurso	Nível Item	Nível Habilidade	Ganha	Perde	Intervalo Tempo do item	Intervalo Tempo da Habilidade	Intervalo Tempo de Interação	Uso da Habilidade	Aprende Sobre Seq. ordenada	Habilidade Prévia
Esta tese	Minimizar tempo e custo	ACO em duas camadas	Gerados	E	S	S	S	S	S	S	S	S	S	S	S

Fonte: Autor

A construção do modelo de simulação e otimização utilizado nesta tese teve como referência o trabalho de Santos (2017), sobre simulação e otimização com regras heurísticas de escalonamento em sistemas JSP, no qual foi utilizado a biblioteca de algoritmo genético GAlib, desenvolvida por Wall (1996), por ser de livre acesso, de código aberto e por ser amplamente referenciada. Para esta tese, o modelo de Simulação foi alterado para que fique adequado a necessidade de alocação de múltiplos recursos com múltiplas habilidades, e o algoritmo de otimização foi substituído pelo algoritmo proposto que é detalhado nas seções subsequentes.

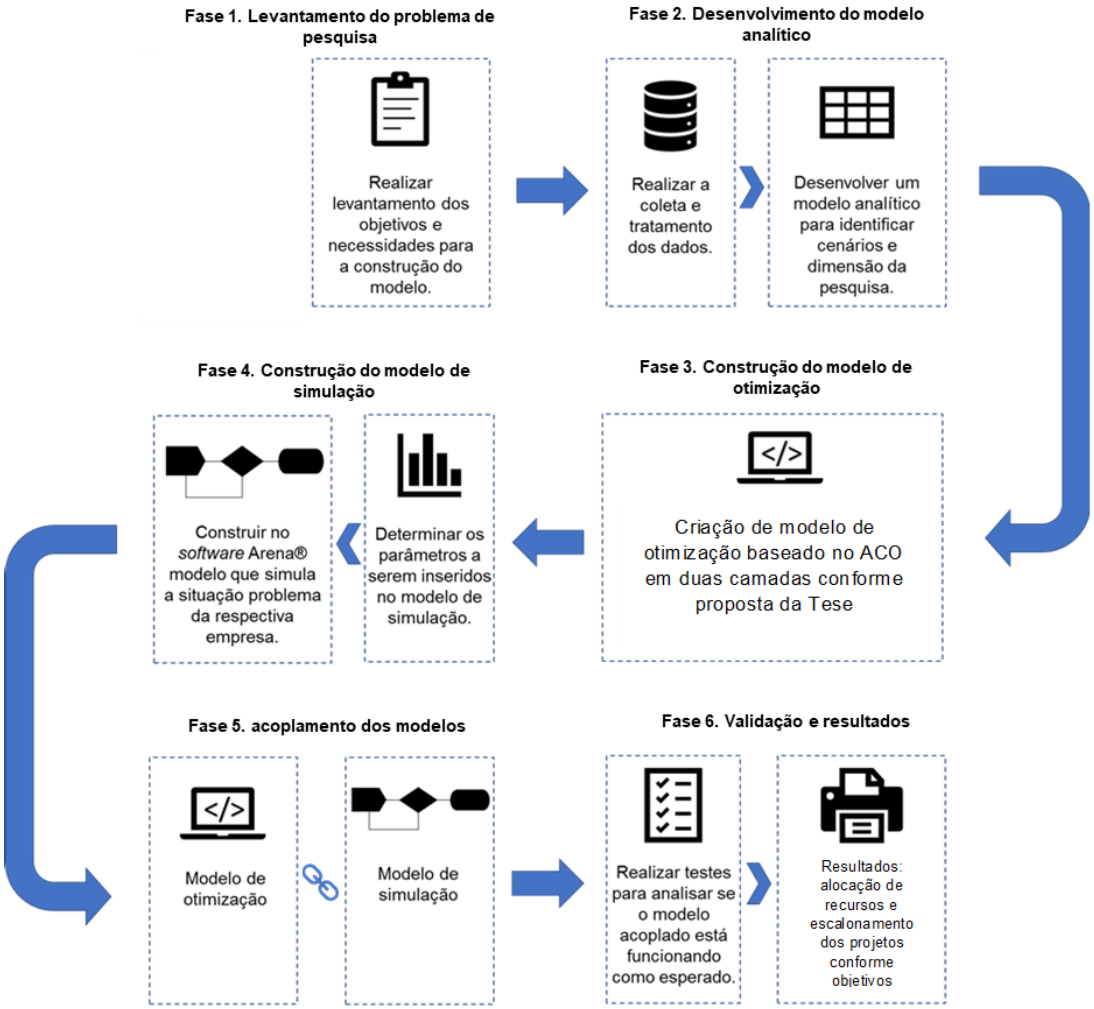
4.2 Instrumentos de pesquisa

É apresentada nesta seção a descrição geral de como o algoritmo foi desenvolvido partindo da identificação do problema, foi elaborada a construção conceitual para dar embasamento a pesquisa. Desta forma, foi possível, desenvolver a implementação do modelo de simulação e apresentar os pontos relevantes sobre o algoritmo de otimização com a utilização do algoritmo híbrido de otimização Colônia de Formigas em duas camadas.

E para concretizar, foi realizado o acoplamento entre o modelo de simulação com o algoritmo de otimização, este último responsável por determinar o escalonamento e a alocação dos recursos nas tarefas que otimizam o sistema e alocação dos recursos.

Visando uma melhor compreensão do leitor sobre esse capítulo, é apresentado na Figura 11 uma visão geral de todas as etapas desta tese, desde a construção do algoritmo até a análise dos resultados.

Figura 11. Visão geral do algoritmo.



Fonte: Adaptado de Santos (2017)

4.3 Técnicas de coleta e tratamento de dados

Para a validação do algoritmo proposto neste trabalho, foram utilizadas bases de dados já existentes. Para validação do MRCPSP, foi utilizada a base RCMPSPLIB disponibilizada pelos autores (VÁZQUEZ; CALVO; ORDÓÑEZ, 2013) e utilizada nos estudos de Pérez *et al.* (2016). Para validação do MSRSPSP, foi utilizada a base iMOPSE (MYSZKOWSKI *et al.*, 2015b) e utilizadas por autores como Myszkowski (2016); Wang e Zheng (2018); e Quoc *et al.* (2020c), com a finalidade de mensurar a capacidade do algoritmo proposto de realizar o escalonamento das tarefas e dos projetos, alocando os recursos, sem sobreposição, bem como identificar os recursos dentro de cada tarefa do projeto. Após a coleta dos dados, estes foram comparados com trabalhos anteriores.

4.4 Algoritmo Proposto

O algoritmo proposto de otimização, é baseado na metaheurística de Otimização Colônia de Formigas ou *Ant Colony Optimization (ACO),)* em duas camadas. Este ACO foi proposto para resolver problemas que podem ser representados por caminhos ou fluxogramas, como o problema do Caixeiro Viajante com diversas evoluções e adaptações para resolução de problemas diferentes, como o RCMPSP (SANCHEZ et al., 2019), o MSRCPSP (MYSZKOWNSKI et al., 2015), processamento em nuvem (CHEN *et al.*, 2019; LI *et al.*, 2019; LIN *et al.*, 2019; WEI, 2020; e JIA *et al.*, 2021) e outros. Nos próximos tópicos, serão abordados os itens do algoritmo proposto.

4.4.1 Representação gráfica do algoritmo proposto

Para a resolução de um problema apenas, o algoritmo ACO vêm sendo empregado e generalizado ao longo do tempo. Como esta tese propôs a solução de múltiplos problemas, buscou-se na literatura um algoritmo que pudesse atender esta necessidade, e adaptou-se o algoritmo de Li *et al.* (2019) que não atendia estes problemas em específico para que pudesse atender aos problemas unificados.

No artigo Li *et al.* (2019), os autores propõe como trabalhos futuros: outros impactos no escalonamento como a data; utilização das incertezas; e paralelizar o processamento da solução. Todas as propostas foram incluídas no algoritmo proposto nesta tese, com exceção do processamento paralelo, e foram considerados como melhoria do algoritmo de Li *et al.* (2019).

Devido ao fato de ser baseado no algoritmo ACO com múltiplas camadas, a primeira camada do algoritmo é a de processamento dos projetos. Nesta camada, são disponibilizadas formigas para percorrer os caminhos para o processamento dos projetos. para cada aresta que as formigas passam, os feromônios são recalculados, bem como a probabilidade de execução do projeto, com a finalidade de verificar o projeto com melhor probabilidade de execução. Nesta camada superior, ainda existe um algoritmo para ajustar a escalonamento dos projetos, evitando assim soluções não factíveis.

Após o ajuste para execução dos projetos, a camada inferior é invocada para realizar a alocação dos recursos e escalonamento das tarefas de cada projeto. Assim como na camada superior, existe um algoritmo para garantir o correto escalonamento das tarefas para que não ocorra soluções não factíveis. Após este escalonamento, o

algoritmo determina a probabilidade de execução das tarefas e inicia o processamento de cada aresta pelas formigas. Nesta camada, são utilizadas diversas colônias de formigas, uma para cada conjunto de tarefas de cada projeto, ou seja, as múltiplas tarefas de um único projeto serão processadas por um único conjunto de formigas. A cada aresta que as formigas percorrem, o feromônio associado com aquele conjunto de formigas/tarefas é atualizado.

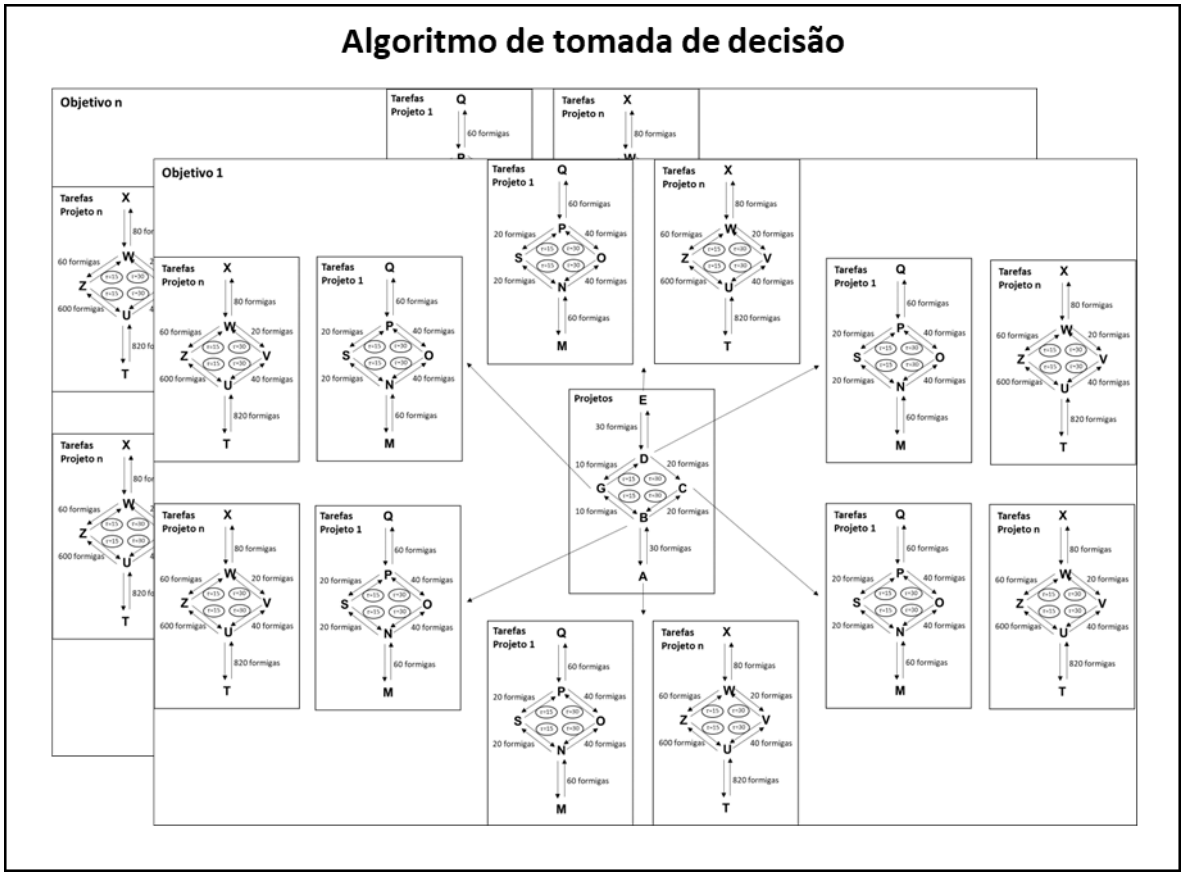
Após o término do processamento de todos os conjuntos de tarefas/colônias, o controle é retornado para a camada superior do algoritmo, que irá atualizar os valores das variáveis globais e os feromônios associados com os projetos. Após esta atualização, esta camada invoca o algoritmo de simulação, e por fim, os dados são armazenados para cada execução.

Por se tratar de um algoritmo multiobjetivos, cada conjunto de camadas superiores e inferiores é processado por cada objetivo de forma sequencial, ou seja, ocorrerá a execução do objetivo 1 para escalonamento e alocação dos recursos disponíveis para todos os projetos e tarefas necessários por todas as formigas para todos os projetos e todas as tarefas, após, será processado o mesmo conjunto anterior, porém para o objetivo 2 e assim se repetindo até que o último objetivo seja executado.

Ao término da execução de todos os objetivos, o algoritmo de tomada de decisões será utilizado para definir a solução mais adequado para o escalonamento das tarefas e alocação dos recursos disponíveis para aquele conjunto de projetos e tarefas definidos, a partir das melhores, piores e de todas as soluções executadas anteriormente, com soluções factíveis, podendo definir a solução ótima.

Na Figura 12 é possível verificar as duas camadas de processamento pelas formigas para o escalonamento dos projetos e tarefas com os recursos sendo alocados, bem como as camadas de processamento por objetivo, e por fim o algoritmo de tomada de decisão que fornecerá a solução para todo o conjunto. Nesta figura, as arestas das tarefas foram demonstradas com o mesmo nome, porém o algoritmo possui métodos para modificar as arestas que as formigas percorrem, buscando assim soluções diferentes a cada execução.

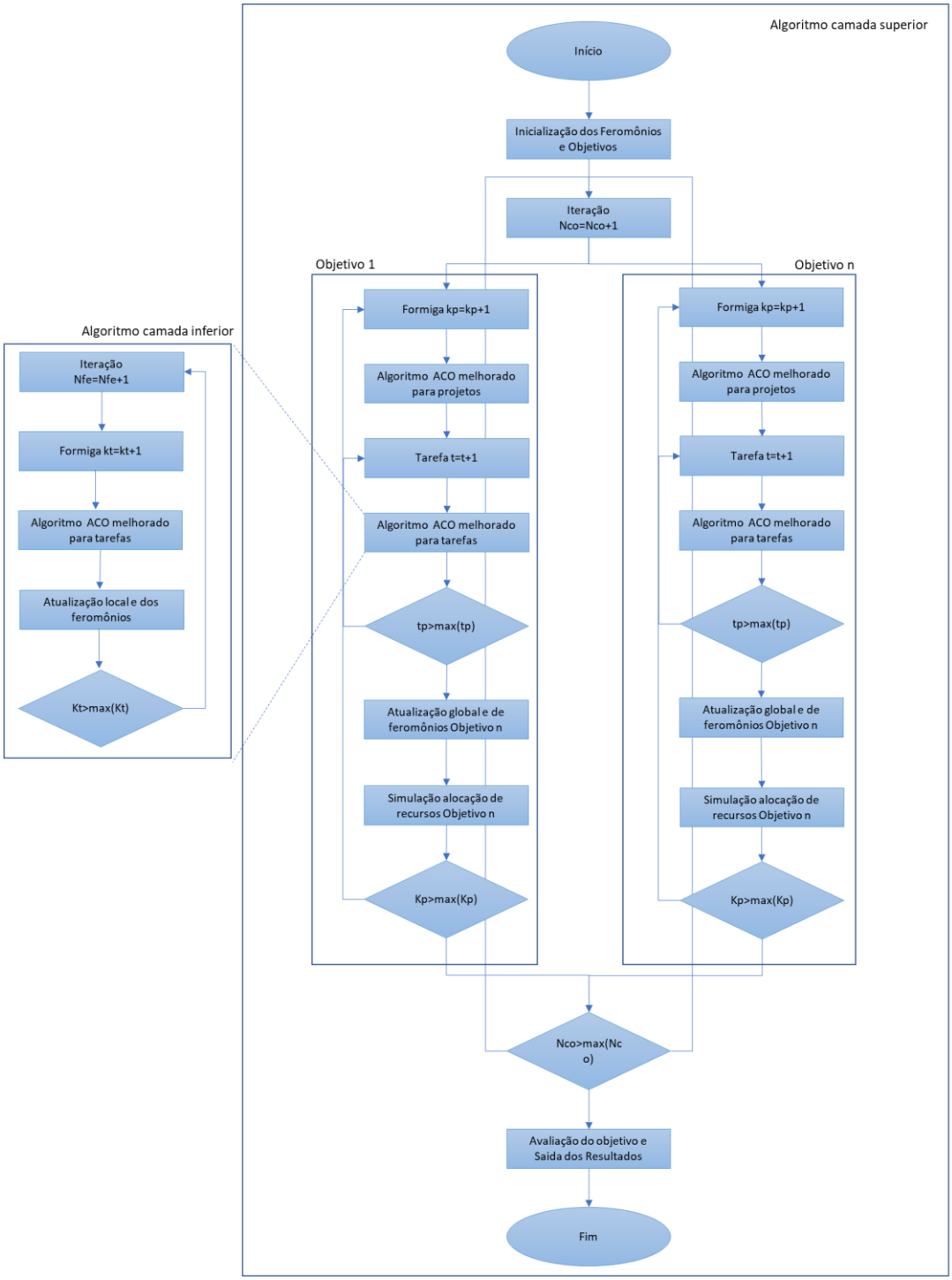
Figura 12. Algoritmo ACO proposto



Fonte: Autor

Com as evoluções que serão abordadas no próximo tópico, e para melhor ilustrar o fluxo de processamento do algoritmo na Figura 13 é demonstrado o fluxograma do algoritmo proposto.

Figura 13. Fluxograma do algoritmo proposto



Fonte: Autor

No próximo tópico, segue o detalhamento das alterações e melhorias realizadas no algoritmo original

4.4.2 Alterações e melhorias do algoritmo proposto

As principais alterações e melhorias propostas em relação ao algoritmo original de Li *et al.* (2019), seguem destacadas abaixo:

- Solução pode ser multiobjetivo e cada objetivo pode ser parametrizado individualmente, podendo executar o algoritmo com quantos objetivos forem necessários ou desejados, com impacto direto no tempo de processamento do algoritmo, pois a solução adotada calcula um objetivo por vez, ou seja, é sequencial;
- Possibilidade de cálculo do objetivo por dados determinísticos (algoritmo atual utiliza este algoritmo), ou utilização de dados estocásticos pela utilização de curva de aprendizado e esquecimento ou da curva de probabilidade exponencial;

Das principais alterações, derivam outras, que são detalhadas abaixo para facilitar o entendimento das principais diferenças entre o algoritmo original e o algoritmo proposto nesta tese:

- Adaptação para trabalhar com projetos, tarefas e recursos humanos;
- Garantia de que somente soluções factíveis serão processadas;
- Limitação dos recursos disponíveis;
- Os recursos são sempre únicos, ou seja, cada recurso deve ser identificado individualmente, mesmo que existam mais recursos com a mesma habilidade e que consigam executar tarefas de mesma complexidade;
- As habilidades dos recursos foram consideradas para a alocação;
- A complexidade das tarefas em relação as habilidades dos recursos foram consideradas para a alocação, e por consequência, impactam diretamente no cálculo da função objetivo dos objetivos;
- Possibilidade de dependência entre os projetos;
- Possibilidade de priorização dos projetos;
- Retirada da dependência de término de um projeto para que outro possa iniciar, caso não haja dependência entre os projetos;
- Novo algoritmo de ajuste para garantir a dependência dos projetos, fazendo com que, projetos que possuam dependência, sejam escalonados na sequência correta;
- Aleatoriedade do primeiro recurso a ser utilizado no cálculo das probabilidades de associação de recurso com tarefas possíveis no início da

iteração das tarefas, evitando que os cálculos sejam executados com a mesma sequência de recursos;

- Possibilidade de ligar ou desligar a utilização adicional de habilidade e de complexidade da tarefa que o recurso pode executar, quando os recursos não possuem a habilidade e a complexidade para execução da tarefa, com valor de pênalti associado (Detalhamento no próximo item);
- Alocação de recursos que não atendam de forma plena as necessidades da tarefa[2], desde que o algoritmo seja parametrizado para permitir tal funcionalidade. Assim, possibilita a alocação de um recurso, mesmo que não tenha a habilidade ou a complexidade para atender a tarefa, caso não exista um outro recurso com a habilidade e complexidade necessários, fazendo com que a tarefa possa ser atendida. Caso contrário, a tarefa não pode ser atendida e o algoritmo de otimização será interrompido. Quando ocorre a alocação de um recurso sem habilidade ou complexidade existente, é inserida uma penalidade no tempo de processamento.
- Pesos de avaliação por objetivo, permitindo assim, que um objetivo possa ter um peso maior no algoritmo de decisão que outro;
- Incorporação do resultado de simulação ao cálculo da melhor solução do projeto;
- Algoritmo de decisão da melhor solução baseado na teoria das distâncias entre as soluções, e permitindo que soluções com dimensões diferentes possam ser avaliadas (Como por exemplo tempo e custo que podem ter valores muito diferentes) baseados nos melhores e piores resultados avaliadas com os resultados de cada iteração[3];
- Utilização de arquivos em formato JSON (*JavaScript Object Notation*) que possibilitam uma legibilidade dos dados de forma mais simples e que podem ser incorporados em outros softwares para utilização dos dados;
- Parametrização do sistema realizada por arquivo de entrada, com fácil substituição e execução;

[2] O recurso não possui a habilidade e complexidade necessárias. Porém, este recurso possui a mesma complexidade para executar uma outra habilidade ou o recurso possui a habilidade necessária, porém com uma complexidade maior ou menor.

[3] O algoritmo de decisão foi utilizado por Chen *et al.* (2017) no artigo para o cálculo de multiobjetivos. O algoritmo foi alterado para permitir quantos objetivos forem parametrizados, desde que o objetivo possa ser calculado com uma função de minimização ou maximização.

- Parâmetro para número de repetições da execução global do sistema (Não é necessário ficar esperando ou acompanhando a execução para testes);
- Tipo de dado a utilizar nos cálculos por parâmetro;
- Possibilidade de execução do algoritmo de otimização com ou sem simulação via parâmetro;
- O modelo de simulação foi construído, de forma que, se tiver um arquivo de entrada no formato desejado, ao se abrir o modelo, a simulação será executada de forma imediata, gerando o arquivo de saída, ou seja, o modelo de simulação é independente do algoritmo de otimização, e vice-versa;
- Modelo de simulação permite a utilização de dado determinístico ou curva de distribuição exponencial. Quando utilizada a curva de aprendizado e esquecimento no algoritmo de otimização, o modelo de simulação é alimentado com o último valor dos recursos calculado conforme a curva de aprendizado e cálculo como um dado determinístico;
- Modelo de simulação flexível e gerado em tempo de processamento, possibilitando qualquer quantidade de recursos e projetos com suas tarefas;
- Os impactos da curva de aprendizado e esquecimento podem ser parametrizados para que não impactem o valor final da otimização. Como por exemplo, não podem exceder por iteração, o valor de 5% o ganho ou perda de eficiência do recurso, ou como outro exemplo, o valor do objetivo, não pode ser 20% superior ou inferior ao valor original do recurso (Minimizar a super utilização ou a sobre alocação de recurso);
- O recalculo da curva de aprendizado pode ser realizado a cada tarefa, ou por uma quantidade pré-determinada de tarefas conforme a parametrização;
- Diferente dos algoritmos propostos para o MSRCPSP e o RCMPSP, os valores objetivos são calculados a partir dos dados dos recursos, e não por um valor dado para a tarefa. Ou seja, recursos com mesma habilidade e que podem executar tarefas de mesma complexidade podem ter valores diferentes para executar a tarefa, dependendo da curva de aprendizado e esquecimento, ou do valor definido, por se tratar de dados estocástico para o recurso;

- As tarefas podem ser utilizadas de forma simples, de tal forma que todos os recursos possam atender a todas as tarefas, através da utilização de uma única habilidade e uma única complexidade para as tarefas e para os recursos, atendendo o RCMPSP; Cada recurso, pode ter distintivas habilidades, com a complexidade igual para todos, com os recursos sendo de mesma forma e com apenas um projeto, atendendo o MSRCPSP; Pode-se utilizar múltiplos projetos e tarefas, com habilidades variadas e uma única complexidade, e os recursos com habilidades variadas e uma única complexidade, tratando de forma integrada o RCMPSP e o MSRCPSP, e escolhendo o tipo de curva a utilizar, podendo ser determinística ou estocástica (aprendizado e esquecimento ou exponencial); pode realizar tudo citado anteriormente, e ainda acrescentar um novo elemento, que é a complexidade das tarefas e nos recursos, potencializando a complexidade da solução final, gerando uma nova categoria de problema.

Na sequência descreve-se a formulação matemática do algoritmo proposto.

4.4.3 Formulação matemática do algoritmo de otimização

Ao algoritmo de Li *et al.* (2019), foram acrescentados novas equações e restrições, que seguem no APÊNDICE I nas tabelas: 20, 21,22 e 23.

4.4.4 Sub algoritmos utilizados pelo algoritmo de otimização

O algoritmo de otimização é composto de duas camadas correspondendo a quatro sub algoritmos principais para o processamento. São apresentados apenas o fluxograma e os pseudocódigos dos algoritmos por questões referentes a proteção de propriedade intelectual.

A camada superior é composta por um sub algoritmo principal e um sub algoritmo para ajuste da sequência dos projetos. Os outros dois sub algoritmos são utilizados na camada inferior do algoritmo, sendo o primeiro para controlar o processamento de toda a camada das tarefas em um projeto e o segundo para o ajuste de sequência das tarefas. Na Figura 13, o primeiro sub algoritmo é identificado como Algoritmo camada superior, o segundo como Algoritmo ACO melhorado para projetos, o terceiro como Algoritmo camada inferior e o último, como Algoritmo ACO melhorado para tarefas. Os pseudocódigos constam do APÊNDICE II.

4.5 Modelo de Simulação

Para a simulação do modelo, foi construído no Software Arena® com o intuito de simular o comportamento de alocação dos recursos ao longo do tempo nos projetos, utilizando-se dados da curva de aprendizado, para verificar os impactos nas variações dos tempos e custo dos projetos que estão sendo otimizados.

Para a construção do modelo de simulação, foi utilizado como base o modelo construído por Santos (2017), utilizando-se a sua lógica em geral e parametrização.

Ao contrário do modelo base, foram adicionados recursos que permitem variar tanto a quantidade de projetos quanto o número de tarefas para cada projeto, conforme a informação enviada em arquivo. É possível variar ainda a quantidade de recursos disponíveis para a alocação e escalonamento das tarefas nos projetos. Além disto, os tempos de execução das tarefas podem ser determinísticos ou estocásticos, conforme dados disponibilizados para o modelo de simulação. O modelo de simulação será detalhado na sequência.

Os dados necessários para criação e execução do modelo de simulação são fornecidos pelo algoritmo de otimização, e integrados por meio de um arquivo. Os dados de entrada do modelo de simulação são:

- Projetos a serem executados;
- Escalonamento dos projetos a serem executados;
- Escalonamento das tarefas dos projetos a serem executados;
- Recursos disponíveis;
- Dados de valor determinístico, último valor da curva de aprendizado, estocástico de execução por recurso;
- Valor de pênalti por utilização de recurso que não possuía a habilidade e complexidade necessário, sendo igual a 1 quando não utilizado;
- Quantidade de recursos disponíveis (sempre igual a 1);
- Tipo de cálculo a utilizar (estocástico ou determinístico).

O resultado da execução do modelo de simulação será gravado em arquivo e utilizado pelo algoritmo de Otimização. Como saída, o modelo de simulação irá fornecer os dados de:

- Tempo total de execução dos projetos por simulação (Parametrizado em 30)

4.5.1 Modelo de simulação no Arena®

Nesta seção serão apresentadas as fases para o desenvolvimento do modelo no software Arena.

A construção foi composta de 2 grandes etapas, que são:

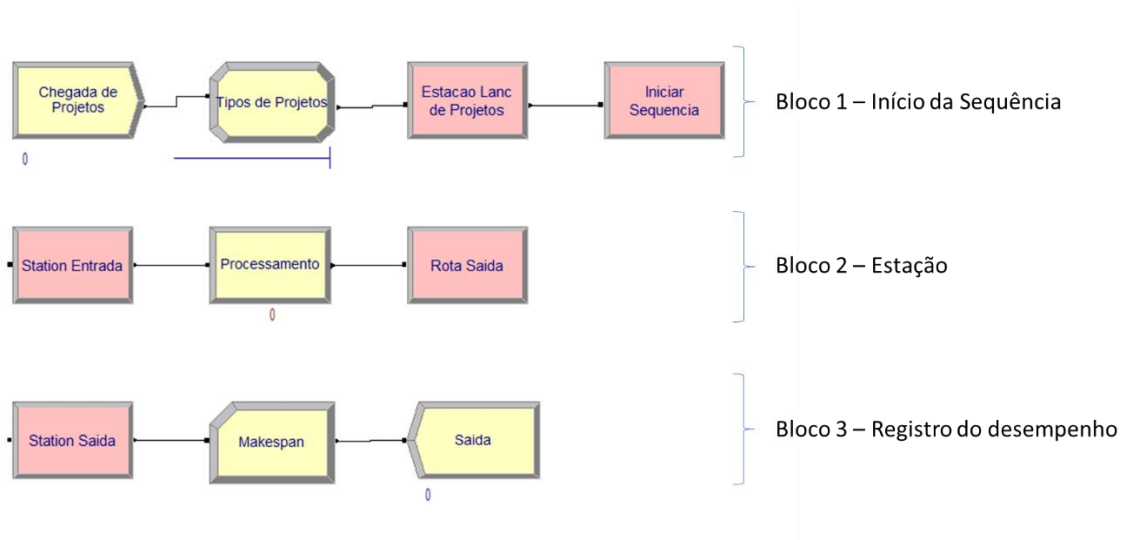
1. Definição de quais dados e fluxos utilizar – que consiste na parte lógica do modelo, com a definição de quais módulos seriam necessários, e quais parâmetros precisariam ser definidos para o processamento automático via Algoritmo de Otimização.
2. Desenvolvimento do código VBA – item responsável pela integração do algoritmo de otimização com o modelo de simulação construído no Arena® e geração do arquivo de saída para o algoritmo de otimização com os dados após o término da simulação.

4.5.2 Definição de dados e fluxo necessários para o modelo de simulação

Na primeira etapa, da construção do fluxo de processamento gráfico, foi definido quais os dados e componentes necessários para processamento. O modelo é dividido em três grandes blocos, conforme definição abaixo e que podem ser verificados na Figura 14.

- Bloco 1 - Início da sequência. Neste bloco do modelo, os projetos são iniciados no sistema e ficam disponíveis para o processamento. Além disto, as variáveis são iniciadas e os dados de desempenho, como tempo e custo são iniciados. Por fim, inicia-se a sequência de processamento.
- Bloco 2 - Estação. Nesta etapa, cada uma das tarefas dos projetos é processada pelos recursos definidos no passo anterior, conforme dados parametrizados e critérios definidos. Esta seção se repete conforme o número de recursos disponibilizados.
- Bloco 3 – Registro do desempenho. Este bloco final armazena os dados dos tempos, custos e demais métricas que foram definidos e serão retornados para o algoritmo de otimização.

Figura 14. Fluxograma do modelo de simulação do respectivo estudo de caso.



Fonte: Autor

A segunda grande etapa na construção do modelo, é a parametrização de dados que serão utilizados por alguns módulos do fluxo gráfico dentro do Arena®. Nesta etapa, o código em VBA lê os dados do arquivo de entrada, realiza as parametrizações necessárias para execução do modelo.

A construção do fluxograma de processamento, bem a parametrização do modelo e início e término da execução ocorre de forma automática através do estímulo do algoritmo de otimização. Este destaque para esta funcionalidade é importante, pois não foi encontrada nas buscas realizadas em bases e na Internet, uma automatização deste nível para simulação, tendo alguns artigos que criam e controlam partes da simulação por um sistema externo.

4.6 Acoplamento do algoritmo de Otimização e do modelo de Simulação

Para o acoplamento do algoritmo de Otimização e do modelo de Simulação, foi utilizada a troca de arquivos e um código de programação em VBA integrado ao modelo de Simulação.

Este código possui as seguintes funcionalidades:

- Ler o arquivo criado pelo algoritmo de Otimização;
- Inicializar o modelo de Simulação que será processado;
- Preencher os valores lidos do algoritmo de Otimização, preenchendo estes conforme detalhado no tópico 4.4;

- Ler os dados de saída e gravar no arquivo que será utilizado pelo algoritmo de Otimização.

Na Figura 15 pode-se observar como foi realizada a integração do algoritmo de Otimização e do modelo de Simulação com o fluxo completo dos dois e a integração destes com a finalidade de atingir o objetivo principal desta tese.

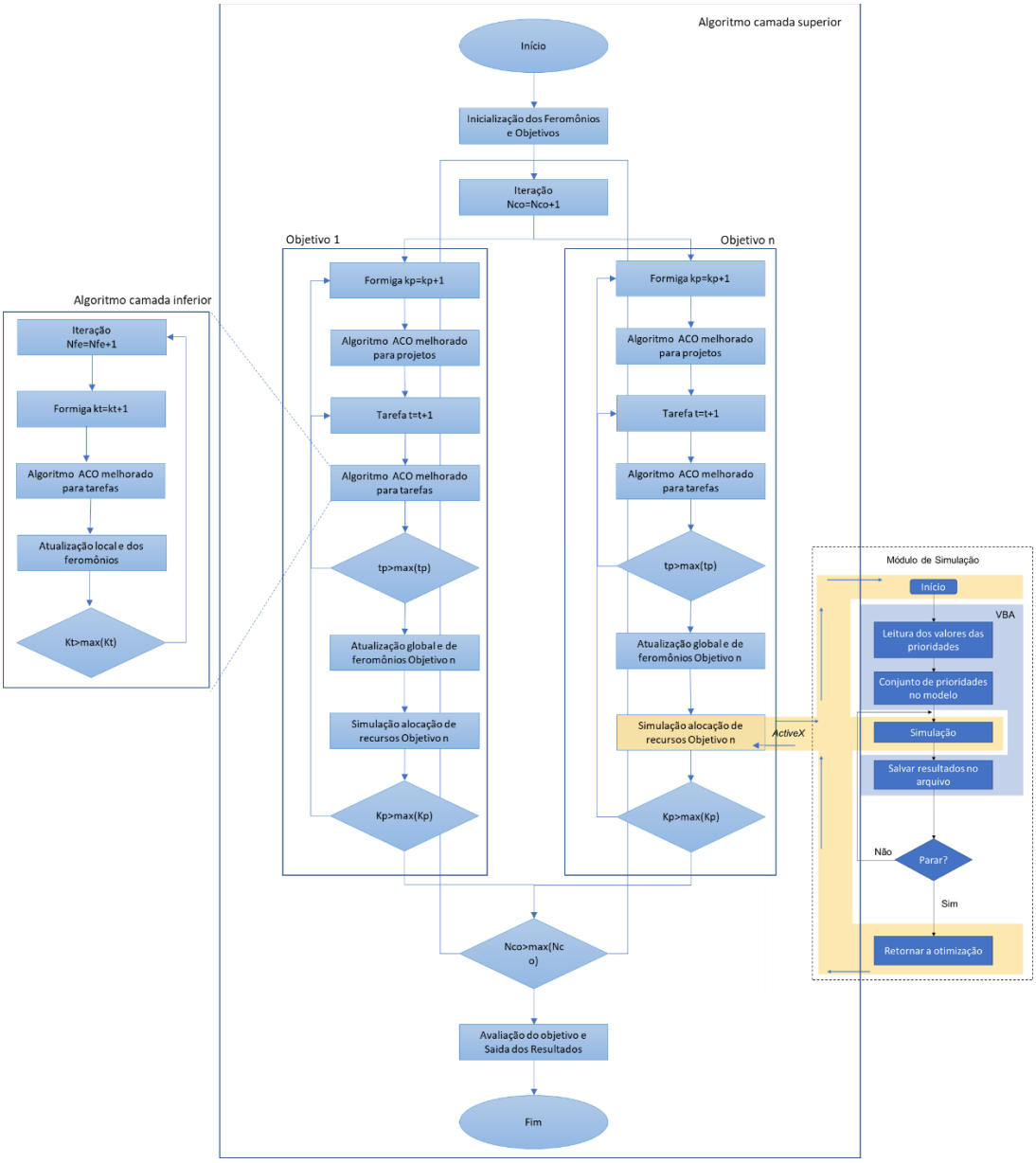
A integração pode ser ligada ou desligada com a ativação ou não de parâmetro no algoritmo de Otimização. Quando o parâmetro está marcado como ligado, o algoritmo gera o arquivo para o modelo de simulação, que cria o modelo de simulação e parametriza conforme os dados informados no arquivo.

O modelo realiza a simulação conforme parametrização definida previamente, com 30 execuções e gera um arquivo de saída com o resultado de cada uma destas execuções, ou é encerrado conforme parametrização de tempo de execução definido no algoritmo de otimização (Impedindo que o algoritmo fique esperando até o término da simulação, caso este tempo seja muito alto).

Os resultados são ponderados por média simples, e acrescentados aos dados gerados pelo algoritmo de otimização, que posteriormente serão utilizados pelo algoritmo de tomada de decisão da melhor solução.

O modelo de simulação foi integrado ao algoritmo de otimização com a finalidade de melhorar a solução global do algoritmo proposto. Ao unificar o algoritmo de otimização com o modelo de simulação, a expectativa é que as soluções dos dois seja tratada de forma integrada e que gere uma solução ótima, ou subótimas mais próxima da realidade da empresa que o algoritmo está sendo empregado no escalonamento das tarefas.

Figura 15. Representação gráfica do acoplamento do algoritmo de Otimização e do modelo de Simulação



Fonte: Autor

Na próxima seção são apresentados os resultados de validação e comparação do algoritmo proposto.

5 Resultados

Neste capítulo serão apresentados os resultados da aplicação do algoritmo proposto na solução dos problemas de escalonamento e alocação de recursos com a utilização de curva de aprendizados.

As validações no algoritmo proposto e no algoritmo integrado tiveram avaliação de 36 itens, que são destacados no Quadro 7.

Quadro 7. Itens validados no algoritmo proposto e no algoritmo integrado

Algoritmo / Modelo	item validado
Algoritmo proposto	Escalonamento de tarefas
	Escalonamento de projetos
	Alocação de múltiplos recursos limitados
	Alocação de recursos com múltiplas habilidades
	Tempo de transferência entre recursos
	Tempo de inicialização da tarefa
	Número de formigas variáveis para os projetos
	Número de formigas variáveis para as tarefas
	Peso do feromônio para os projetos
	Peso do feromônio para as tarefas
	Peso da heurística para os projetos
	Peso da heurística para as tarefas
	Taxa de evaporação do feromônio para os projetos
	Taxa de evaporação do feromônio para as tarefas
	Taxa de atualização global para os projetos
	Taxa de atualização global para as tarefas
	Número de iteração dos projetos
	Número de iteração das tarefas
	Múltiplos objetivos
	Peso de avaliação por objetivo
	Área de aplicação genérica
	Tempo determinístico
	Tempo estocástico
	Tempo com curva de aprendizado
	Acoplamento com modelo de simulação
	Limitação do tempo de execução do modelo de simulação
	Múltiplas execuções
	Variação da complexidade
	Variação da habilidade
	Possibilidade de execução com recurso sem determinada habilidade
	Possibilidade de execução com recurso sem determinada complexidade
	Aplicação de penalidade, quando recurso sem habilidade ou complexidade
Modelo integrado	Todos os dados do algoritmo proposto

Algoritmo / Modelo	item validado
	Leitura do arquivo de entrada
	Parametrização automática do modelo
	Criação automática do modelo
	Dados de simulação na saída

Fonte: Autor

O algoritmo proposto foi inicialmente validado para os problemas de alocação e escalonamento, com dados determinísticos e demais parâmetros definidos conforme Li *et al.* (2019). Posteriormente, em um segundo conjunto de experimentos, foram inseridos tempos estocásticos conforme curva de aprendizado e esquecimento. Os resultados apresentados são baseados em 30 execuções do modelo.

Na sequência, validou-se os tipos de dados que os recursos gastam para executar as tarefas, sendo os dados possíveis: determinísticos; estocásticos com a curva probabilística exponencial; e a LFCM.

O algoritmo integrado foi executado primeiramente com modelo de simulação desabilitado e, posteriormente, em conjunto com a simulação, evidenciando que o algoritmo proposto pode ser executado isoladamente, ou acoplado ao modelo de simulação.

Continuando com as validações, foi habilitado o parâmetro que possibilita a alocação de um recurso que não possua a habilidade ou complexidade exigida para executar determinada tarefa[4]. Por fim, foram parametrizados mais de um objetivo, possibilitando que o algoritmo seja multiobjetivo.

Os resultados são comparados com a literatura correlata, como forma de validação da proposta conforme detalhado no Quadro 8.

.

[4] A tarefa necessita de um recurso que possua determinada habilidade e complexidade para executar a tarefa. Se não tiver recurso com esta habilidade e complexidade de execução da tarefa, a habilitação do parâmetro possibilita alocar um recurso que tenha a habilidade necessária, porém com uma complexidade maior ou menor. Ou possibilita a alocação de um recurso que tenha a complexidade da habilidade necessária, porém não tenha a habilidade. Para ambos os casos, é aplicada uma penalidade, por exemplo no tempo de execução. Com o parâmetro desabilitado, o algoritmo é interrompido, pois não terá solução factível.

Quadro 8. Quadro comparativo literatura correlata

Problema	Comparação	Base utilizada
Algoritmo de Li *et al.* (2019)	Algoritmo proposto resolve o problema da mesma forma que o algoritmo original?	Li *et al.* (2019)
RCMPSP	Algoritmo proposto consegue resolver o problema RCMPSP, escalonando as tarefas e alocando os recursos em múltiplos projetos?	RCMPSPLIB
MSRCPSP	Algoritmo proposto consegue resolver o problema MSRCPSP, escalonando as tarefas e alocando os recursos com em múltiplas habilidades em projeto?	iMOPSE
LFCM	Algoritmo proposto consegue resolver o problema LFCM, escalonando as tarefas e alocando os recursos com em múltiplas habilidades em projetos considerando a curva de aprendizado e esquecimento dos recursos?	Chen *et al.* (2017)

Fonte: Autor

Os comparativos do algoritmo proposto e no algoritmo integrado com a literatura correlata tiveram um total de 53 itens comparados, que são destacados no Quadro 9.

Quadro 9. Itens comparados do algoritmo proposto e no algoritmo integrado com literatura correlata

Algoritmo / Problema	Item comparado
Algoritmo original	Escalonamento de tarefas
	Escalonamento de projetos
	Alocação de múltiplos recursos limitados
	Alocação de recursos com múltiplas habilidades
	Tempo de transferência entre recursos
	Tempo de inicialização da tarefa
	Número de formigas variáveis para os projetos
	Número de formigas variáveis para as tarefas
	Peso do feromônio para os projetos
	Peso do feromônio para as tarefas
	Peso da heurística para os projetos
	Peso da heurística para as tarefas
	Taxa de evaporação do feromônio para os projetos
	Taxa de evaporação do feromônio para as tarefas
	Taxa de atualização global para os projetos
	Taxa de atualização global para as tarefas
	Número de iteração dos projetos
	Número de iteração das tarefas
RCMPSP	Alocação de recursos limitados
	Duração da tarefa variável
	Área de aplicação genérica

Algoritmo / Problema	Item comparado
	Múltiplos objetivos
	Dados do teste de base gerada
	Escalonamento de projetos
	Escalonamento de tarefas
	Precedência das tarefas
	Tempo estocástico
MSRCPSP	Variação da proficiência da habilidade
	Recursos limitados com múltiplas habilidades
	Aprendizado do recurso
	Duração da tarefa variável
	Área de aplicação genérica
	Múltiplos objetivos
	Dados do teste de base gerada
	Escalonamento de tarefas
	Precedência das tarefas
	Tempo estocástico
LFCM	Dados do teste de base gerada
	Tempo estocástico
	Habilidade do recurso
	Variação do nível da habilidade
	Nível da habilidade
	Ganho de habilidade
	Perda de habilidade
	Tempo influencia na habilidade
	Considera intervalo de tempo para ganho ou perda
	Escalonamento de projetos
	Escalonamento de tarefas
	Precedência das tarefas
	Considera o Intervalo de tempo entre cada iteração
	Aprende conforme escalonamento
	Múltiplos objetivos
	Considera que o recurso possui habilidade prévia

Fonte: Autor

Nos gráficos de Gantt dos resultados, os recursos tiveram seus nomes como Rx, de forma que o x é o número dos recursos, por exemplo, o R3 representa o recurso 3. Cada projeto foi destacado com uma cor diferente, ou seja, quando a cor de todos as tarefas é igual, representa apenas 1 projeto, e quando existir mais de uma cor, cada cor representa 1 projeto.

As tarefas foram nomeadas com Tyy. O número yy representa o número da tarefa, por exemplo, o T1010 representa a tarefa 10 do projeto 10, enquanto, T110,

representa a tarefa 10 do projeto 1. As setas entre as tarefas indicam a precedência da tarefa dentro do projeto, para setas que unem tarefas com a mesma cor. Em tarefas com cores distintas, a seta indica precedência de projetos.

Os experimentos foram todos executados em um computador com a configuração da Tabela 1, com dedicação exclusiva para o processamento.

Tabela 1. Configuração do computador utilizado nos testes

Organização	Configuração
Edição	Windows 10 Pro
Versão	21H2
Build SO	190.441.526
Processador	Intel(R) Core(TM) i7-3537U CPU @ 2.00GHz
RAM instalada	8,00 GB (7,88 GB utilizáveis)
Tipo de sistema	Sistema Operacional de 64 bits, processador com base em x64

Fonte: Autor

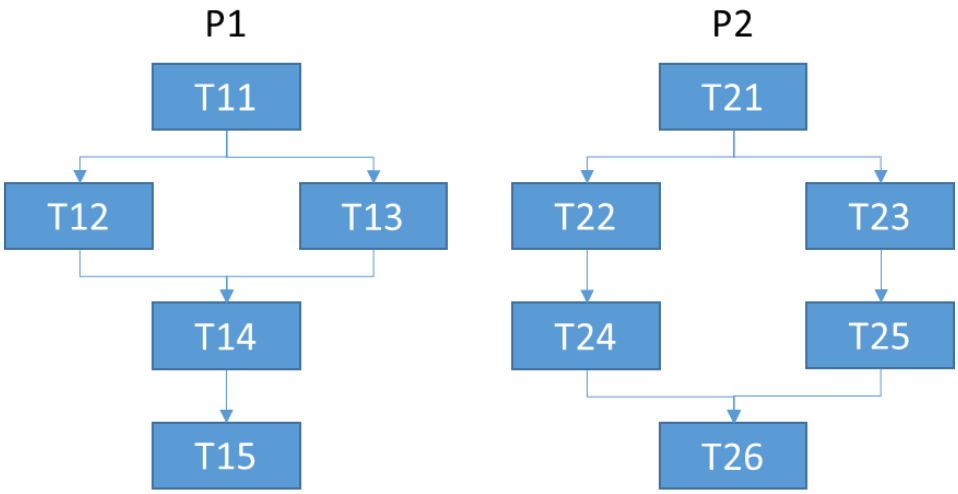
5.1 Validação das alterações e melhorias do algoritmo

Esta seção apresenta os resultados de verificação do algoritmo proposto em suas diferentes funcionalidades. As funcionalidades foram inseridas uma a uma e os respectivos resultados foram avaliados em relação a capacidade de encontrar uma solução viável. Assim, foram realizadas as verificações conforme Quadro 7.

5.1.1 Dados utilizados

Para esta primeira validação, foi utilizado um cenário com dois projetos e um total de 11 tarefas, conforme a Figura 16. Nesse caso, o experimento considera como objetivo único, que é minimizar o Makespan, conforme o algoritmo base de Li *et al.* (2019).

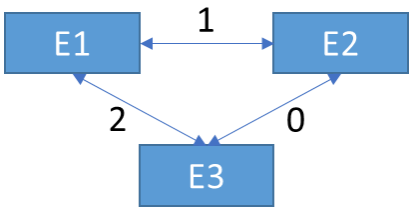
Figura 16. Projetos com suas tarefas utilizadas para comparação do algoritmo proposto com o algoritmo de Li et al. (2019)



Fonte: Autor

Os tempos de transferência entre recursos de diferentes empresas foi considerado[5], e segue os tempos na Figura 17. As empresas foram definidas como EX, sendo o X variável neste caso de 1 a 3, ou seja, existem 3 empresas distintas sendo consideradas. A terminação das setas, indica que, pode-se ocorrer a transferência da tarefa entre as empresas, em todos os casos sendo bidirecional, por exemplo, pode-se transferir tarefas da empresa 1 para a empresa 2 ou vice-versa. Por fim, o tempo da transferência é o valor definido sobre a linha, como por exemplo, o tempo é igual a 1 na transferência da empresa 1 para a empresa 2, ou vice-versa.

Figura 17. Tempo de transferência entre empresas



Fonte: Autor

Nas tabelas: 2 e 3 pode-se verificar a parametrização do tempo de execução da tarefa TXX (XX representa o número da Tarefa) pelos recursos RY (Sendo Y o número do recurso) e o tempo que os recursos gastam na inicialização para execução

[5] Na literatura, com o JSP trata de máquinas, é definido com os tempos de transporte e preparação, e no algoritmo, pode se tratar de recursos, que estão atrelados a empresas, foi utilizado como tempo de transferência entre recursos de empresas ou área distintas e como tempo de inicialização das tarefas.

da tarefa, respectivamente. As linhas que estão com o dado N/A significa que o recurso não consegue atender a tarefa.

Tabela 2. Tempo de execução da tarefa pelo recurso

	T11	T12	T13	T14	T15	T21	T22	T23	T24	T25	T26
R1	3	2	N/A	2	N/A	2	N/A	1	3	N/A	3
R2	N/A	3	3	1	3	N/A	1	N/A	1	3	N/A
R3	1	N/A	3	N/A	1	2	2	3	N/A	1	3

Fonte: Autor

Tabela 3. Tempo de inicialização gasto pelo recurso para a tarefa

	T11	T12	T13	T14	T15	T21	T22	T23	T24	T25	T26
R1	2	2	N/A	2	N/A	2	N/A	2	3	N/A	1
R2	N/A	2	2	1	2	N/A	1	N/A	3	1	N/A
R3	1	N/A	2	N/A	1	1	2	3	N/A	3	3

Fonte: Autor

O último dado utilizado foi o de carga de trabalho, que é um parâmetro definido para multiplicar o valor da tarefa conforme definição dos autores Li *et al.* (2019). Este valor foi definido como 5 para a comparação do algoritmo proposto, pois foi o valor observado no algoritmo base.

5.1.2 Parametrização do algoritmo proposto

As parametrizações do algoritmo de otimização foram realizadas de forma separada para tarefas e projetos, conforme a Tabela 4. Foram parametrizados: número de formigas; peso do feromônio; peso da informação heurística; taxa de evaporação; taxa de atualização global; e número máximo de iterações.

Tabela 4. Parametrização do algoritmo

Algoritmo	Número de formigas	Peso do feromônio	Peso da heurística	Taxa de evaporação	Taxa de atualização global	Número de iterações
Tarefa	[1,5 * |Número de Tarefas|]	1	0,7	0,2	1	100
Projeto	[0,5 * |Número de Projetos|]	1	2	0,3	1	10

Fonte: Autor

Por fim, o algoritmo foi parametrizado para duas execuções, utilizando dados determinísticos, com a possibilidade de inclusão de habilidades ou complexidade desligada, com a simulação desligada, com a carga de trabalho definido como cinco, com apenas um objetivo com a função objetivo para minimizar o tempo total de execução dos projetos.

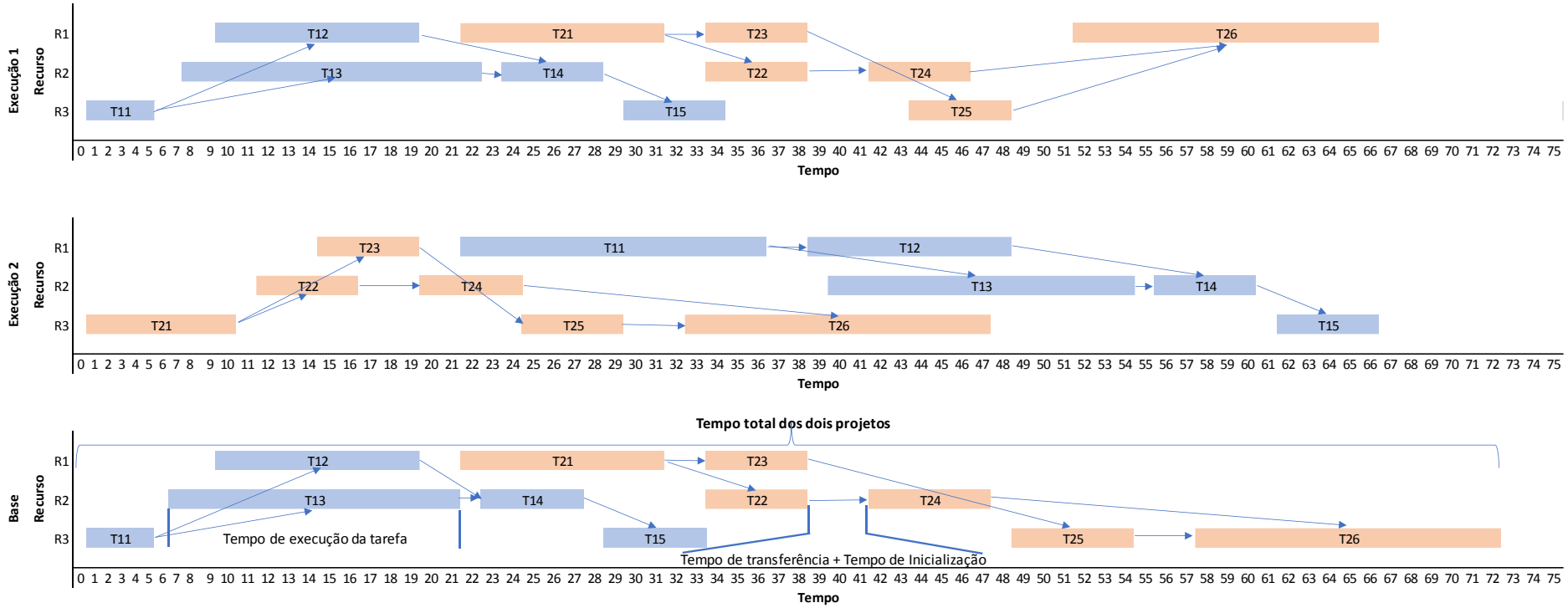
5.1.3 Resultados da validação

A execução do algoritmo com os dados informados no algoritmo base instância, resultou nas seguintes combinações de soluções das tarefas:

- Projeto 01 com 05 combinações;
- Projeto 02 com 16 combinações.

A Figura 18 ilustra os resultados obtido em duas execuções do algoritmo proposto e uma comparação com os resultados de Li *et al.* (2019). Na figura, o projeto 1 foi representado na cor azul, e o projeto 2 na cor laranja.

Figura 18. Comparativo entre execuções algoritmo proposto e algoritmo de Li et al. (2019)



Fonte: Autor

O tempo de execução da tarefa é definido pelo tempo que o recurso RX gasta para executar determinada tarefa, com o tempo sendo destacado na cor do projeto, por exemplo a tarefa T11 da Execução1 foi realizada pelo recurso R3 com um tempo igual a 5, pois é o valor de execução da tarefa que é 1, conforme Tabela 2, vezes o valor da carga de trabalho, que foi definido como 5.

Além deste tempo, foi verificado que os tempos de transferência e inicialização da tarefa respeitam as parametrizações definidas, havendo um tempo antes do início das tarefas, como por exemplo da tarefa T11 da Execução, que o tempo de transferência é 0, devido a ser a primeira tarefa e o tempo de inicialização é igual a 1, conforme Tabela 3.

A última informação em destaque, é o tempo total dos dois projetos, ou Makespan. Com as alterações propostas, tanto a primeira, quanto a segunda execução tiveram um tempo total de 67 unidades pelo algoritmo proposto, enquanto o algoritmo base possui um total de 73 unidades, com uma melhoria de 8% do tempo total para o algoritmo proposto.

O algoritmo foi executado em 25 segundos, encontrando 2 combinações diferentes para o projeto 1 e 6 para o projeto 2 na primeira execução e em 49 segundos, encontrando 5 combinações diferentes para o projeto 1 e 16 para o projeto 2 na execução 2. O programa gerado consome entre 30% e 40% da CPU total disponível do computador e utiliza entre 7,5% e 15% da memória total disponível.

5.1.4 Repetições na execução do algoritmo

A próxima validação realizada, foi em relação ao tempo de execução do algoritmo, tempo de execução dos projetos e soluções possíveis encontradas. Alterando-se o parâmetro de repetições para 30 execuções do algoritmo proposto de forma sequencial e automática.

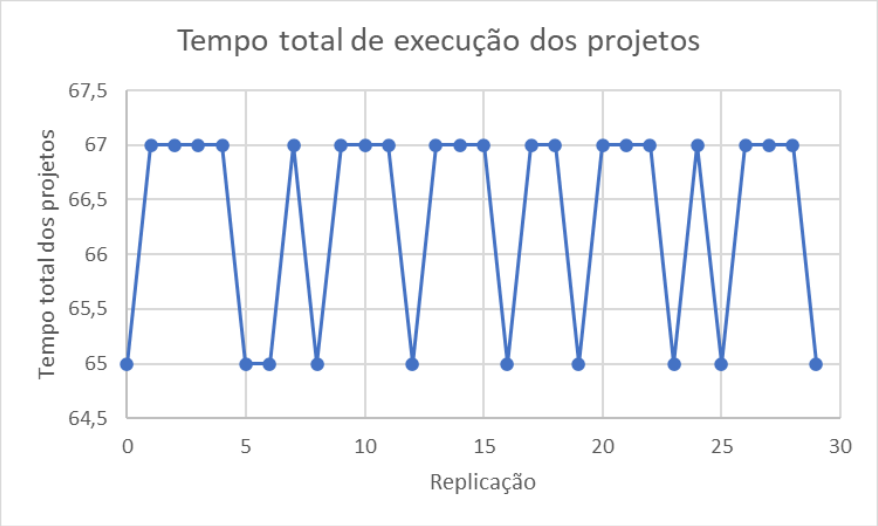
Na Figura 19 verifica-se que os tempos de execução possuem pouca variação, entre 24 e 28 segundos.

Figura 19. Tempo de execução em 30 replicações do algoritmo

Fonte: Autor

A figura 48 apresenta os resultados obtidos em relação ao tempo total de execução dos projetos.

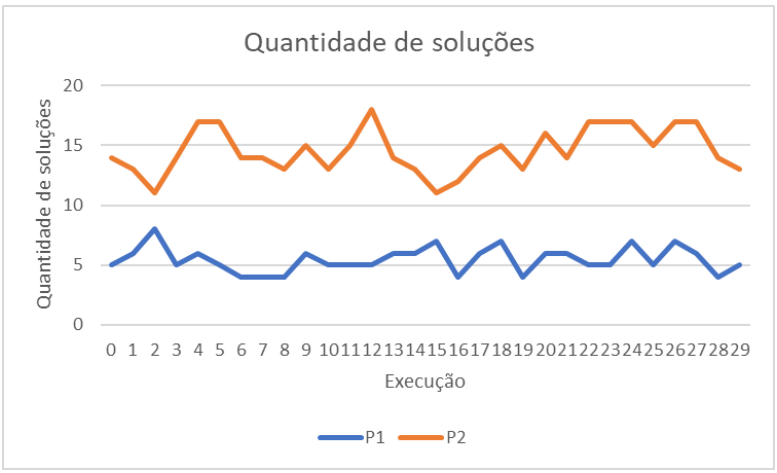
Figura 20. Tempo total de execução dos projetos



Fonte: Autor

A quantidade de soluções encontradas por projeto é demonstrada na Figura 21. A quantidade mínima de soluções para o projeto 1 foi de 4 soluções, e para o projeto 2 foi de 11 soluções. A quantidade máxima para o projeto 1 foi de 8 soluções e para o projeto 2 de 18 soluções. Isso indica que o algoritmo proposto possibilita combinações de tarefas dentro dos projetos com uma grande variação, e que se pode obter diversas soluções para a mesma base, fazendo com que a solução seja alterada conforme a iteração, e com o algoritmo de decisão, ofereça soluções distintas para execuções distintas.

Figura 21. Quantidade de soluções possíveis por projeto



Fonte: Autor

O parâmetro de repetições se mostrou eficaz para o algoritmo proposto, ou seja, permite a execução do algoritmo diversas vezes, conforme o parâmetro definido. Como os resultados se mostraram consistentes continuou-se com a validação do algoritmo proposto habilitando o módulo de simulação.

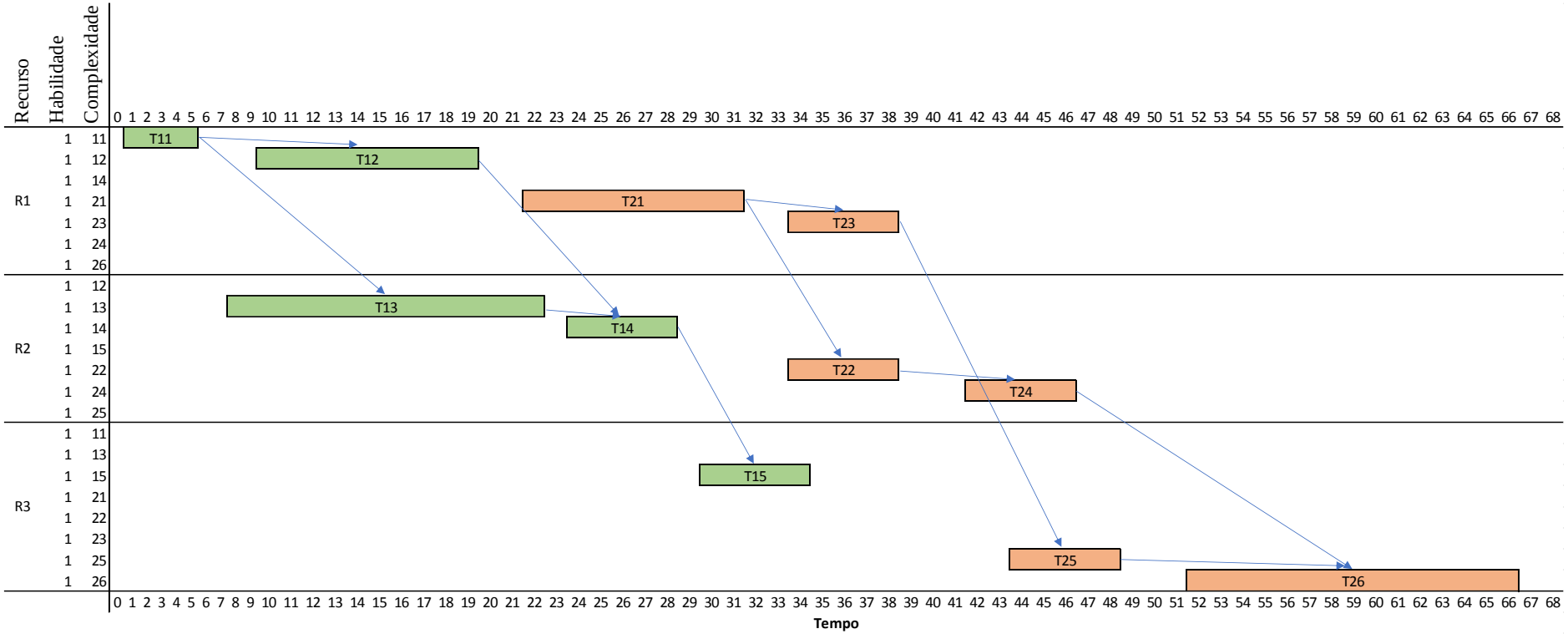
5.1.5 Algoritmo integrado com modelo de simulação

O modelo de simulação foi parametrizado para 30 execuções com cada uma das simulações. Com a integração do modelo, notou-se que o tempo total de processamento se elevou de forma considerável, que estava próximo dos 25 segundos para 188 e 202 segundos, ou seja, 7,5 a 8 vezes superior ao tempo de execução sem o modelo de simulação. O modelo permite a simulação de soluções factíveis[6], e incorpora o resultado da simulação com os dados do algoritmo proposto, para que o algoritmo de decisão ofereça a melhor solução.

Ao avaliar a solução gerada com o modelo de simulação, verifica-se que a solução é factível e dentro das restrições impostas ao modelo. Na Figura 22 verifica-se a solução gerada com o modelo de simulação ativado. O tempo total da solução dos projetos é igual ao encontrado na seção 5.1 encontrado previamente com o algoritmo.

[6] Soluções factíveis, são as soluções que atende a todas as restrições dos problemas e as condições de não negatividade. Por exemplo, garantir que um recurso só pode executar uma tarefa por vez, ou que o recurso, só possa executar tarefas que possua a habilidade e a complexidade exigida para execução daquela tarefa

Figura 22. Solução gerada pelo algoritmo com Simulação habilitada



Fonte: Autor

O modelo de simulação acoplado ao algoritmo de otimização se mostrou eficaz, pois realiza as simulações do modelo e produz saída de dados conforme dados previamente obtidos na seção 5.1, porém não se mostram eficientes em termos de tempo de execução.

Na próxima seção, são realizados novos testes de verificação com a inserção de tempos de processamento estocásticos.

5.1.6 Algoritmo com dados estocásticos

Nos experimentos dessa seção o algoritmo foi parametrizado com dados estocásticos com curva probabilística exponencial para os recursos, conforme detalhadamente Quadro 10. A parametrização dos projetos foi mantida conforme experimentos anteriores, definidos na seção 5.1.

Quadro 10. Parametrização de dados estocásticos

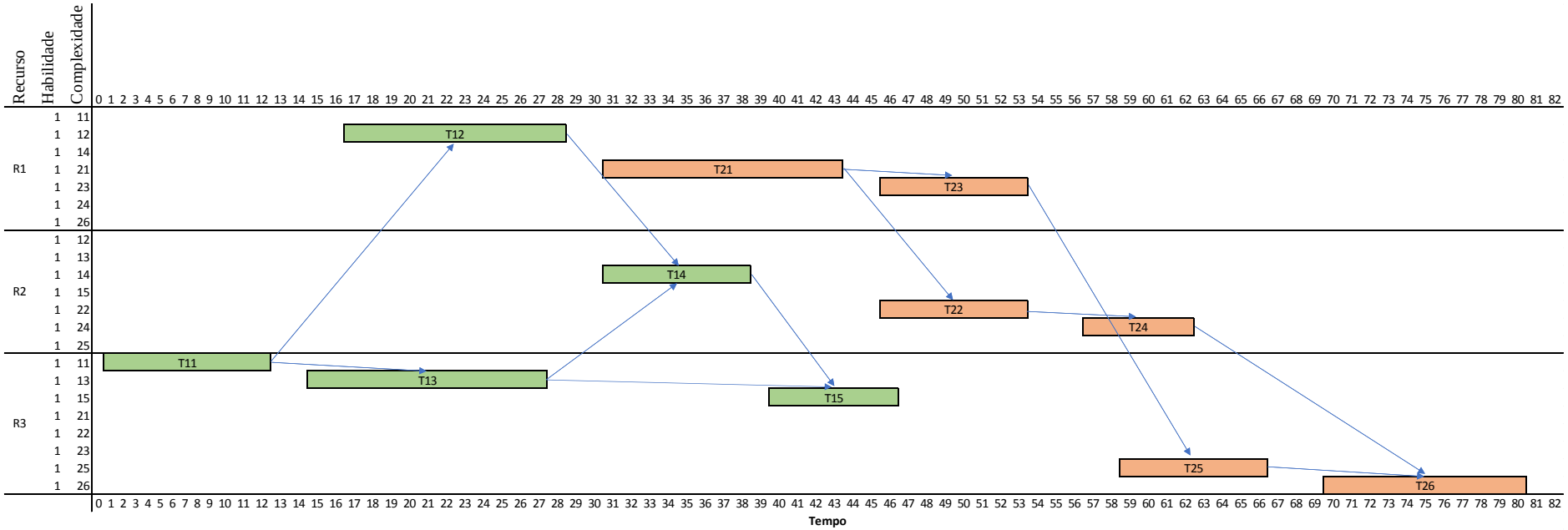
Recurso	Habilidade	Complexidade	Valor
1	1	11	0.5 + EXPO(1.08)
1	1	12	0.5 + EXPO(0.96)
1	1	14	0.5 + EXPO(0.96)
1	1	21	0.5 + EXPO(0.96)
1	1	23	EXPO(0.75)
1	1	24	0.5 + EXPO(1.08)
1	1	26	0.5 + EXPO(1.08)
2	1	12	0.5 + EXPO(1.08)
2	1	13	0.5 + EXPO(1.08)
2	1	14	EXPO(0.75)
2	1	15	0.5 + EXPO(1.08)
2	1	22	EXPO(0.75)
2	1	24	EXPO(0.75)
2	1	25	0.5 + EXPO(1.08)
3	1	11	EXPO(0.75)
3	1	13	0.5 + EXPO(1.08)
3	1	15	EXPO(0.75)
3	1	21	0.5 + EXPO(0.96)
3	1	22	0.5 + EXPO(0.96)
3	1	23	0.5 + EXPO(1.08)
3	1	25	EXPO(0.75)
3	1	26	0.5 + EXPO(1.08)

Fonte: Autor

Os tempos de execução em 30 replicações variaram entre 27 e 28 segundos, mostrando que a introdução de tempos exponencialmente distribuídos não produz acréscimo significativo no custo computacional em relação ao algoritmo com dado determinístico.

Avaliando a solução do algoritmo na Figura 23, o algoritmo atende a solução proposta, produzindo resultado conforme esperado e atendendo as restrições impostas.

Figura 23. Validação do algoritmo com dado probabilístico de curva exponencial



Fonte: Autor

O tempo total do projeto aumentou em aproximadamente 15% a 18% em relação aos tempos das seções anteriores, porém a qualidade da solução melhorou, pois os recursos, quando executam uma determinada tarefa, podem apresentar variação nos tempos, não sendo sempre padronizados. Os tempos utilizados por cada recurso para a execução da tarefa são armazenados no momento do processamento do algoritmo, e devido a este fato, os dados podem ser apresentados conforme o momento do cálculo.

A próxima seção apresenta os resultados para o caso em que o algoritmo proposto permite a alocação de um recurso que não possui a complexidade necessária para execução de uma tarefa. Lembrando que, nesse caso, a tarefa é executada com uma penalização no tempo de execução.do algoritmo.

5.1.7 Tarefa não possui recursos disponíveis com a complexidade exigida

Nessa seção, a parametrização do algoritmo permite que um recurso sem a complexidade necessária para execução de determinada tarefa, seja alocado para a tarefa, com penalização no tempo de execução e inclusão da complexidade necessária para execução da tarefa no recurso. Para definir os recursos elegíveis, verifica-se quais recursos possuem a habilidade necessária para execução da tarefa. Dentre os recursos escolhidos, verifica-se qual menor complexidade dentre as complexidades maiores que a demandada pela tarefa. Definida a complexidade do recurso, o tempo desta complexidade é multiplicado pelo valor da penalidade para continuidade do processamento do algoritmo proposto.

Por exemplo, uma tarefa que tenha complexidade de execução igual a 3 e exija uma habilidade 5. Verifica-se todos os recursos com habilidade 5 e que consigam executar uma tarefa de complexidade 3. Caso não tenha, verifica se existem recursos com habilidade 5. Ao encontrar estes recursos, separa os que possuem uma complexidade de execução maior que 3, podendo ter os valores 4 e 5, e escolhe o valor 4, pois é o menor entre os maiores da complexidade exigida para execução da tarefa. Seleciona o tempo que o recurso de habilidade 5 e complexidade 4 possui, e multiplica pelo valor de pênalti definido. Este valor será alocado para o recurso escolhido na habilidade 5 e complexidade 3 (Complexidade exigida para executar a tarefa).

Foram retiradas as parametrizações de complexidade com valor 12 dos recursos 1 e 2. A parametrização da penalidade foi definida com valor 2, mantendo-se os demais parâmetros da base e do algoritmo.

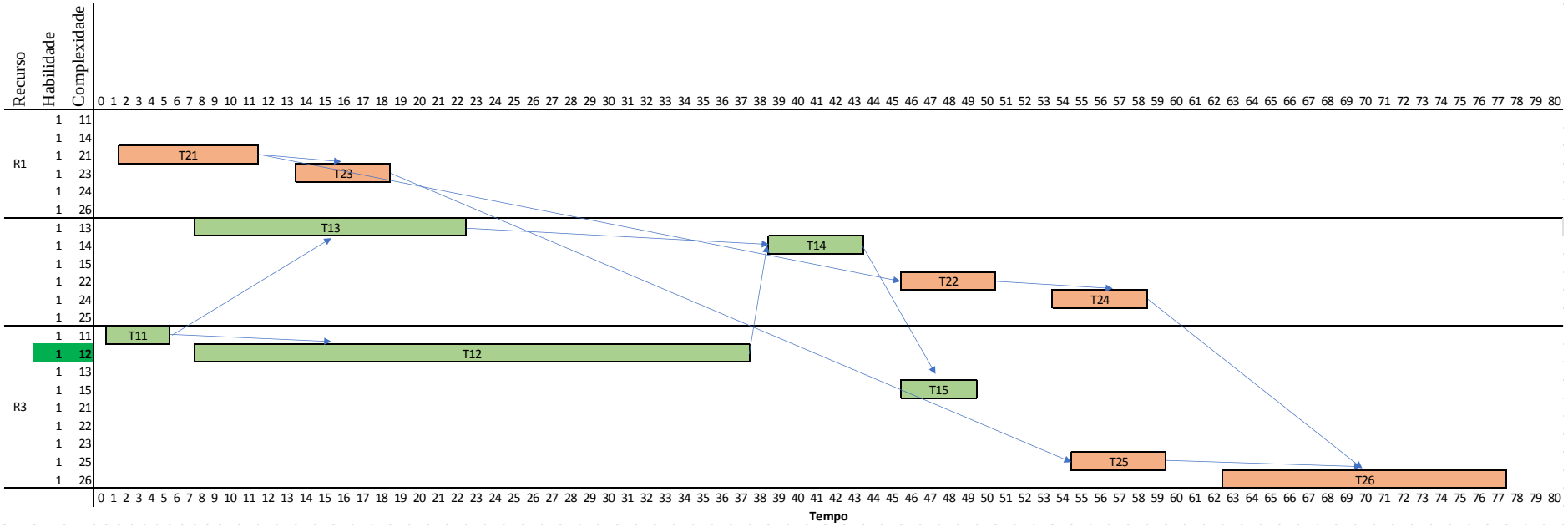
O tempo de processamento ficou entre 23 e 26 segundos, ou seja, o algoritmo consegue resolver a alocação de uma nova complexidade para o recurso dentro do tempo de previamente encontrado no item 5.2.1.

Na Figura 24 verifica-se a solução do algoritmo alocando uma complexidade para o Recurso 3, que originalmente não possuía a complexidade 12, porém possuía a habilidade 1. A complexidade 12 poderia ter sido adicionada para qualquer um dos recursos, visto que todos possuem a habilidade necessária. Para definir o valor a ser atribuído para a complexidade 12, o algoritmo escolheu a próxima complexidade maior do recurso, neste caso, definiu o valor 3.

Com isto, foi multiplicado o valor da penalidade vezes o valor da complexidade, ficando com o valor 6. Como o algoritmo está trabalhando com o valor de carga de trabalho igual a 5, o tempo total ficou igual a 30 para a tarefa T12.

Ainda conforme a Figura 24, verifica-se que o tempo total da solução para o algoritmo ficou em 78 unidades de tempo, condizente com os tempos encontrados anteriormente e aplicação da penalidade.

Figura 24. Validação do algoritmo com recurso sem complexidade para execução de determinada tarefa



Fonte: Autor

A próxima seção avalia a possibilidade de alocação de um recurso que não possui a habilidade necessária para execução da tarefa.

5.1.8 Tarefa não possui recursos disponíveis com a habilidade exigida

Nessa seção, a parametrização do algoritmo permite que um recurso sem a habilidade necessária seja alocado para uma tarefa, com penalização no tempo de execução. Para definir os recursos elegíveis, verifica-se quais recursos possuem a complexidade necessária para execução da tarefa. Dentre os recursos escolhidos, verifica-se qual o maior valor de tempo de execução de uma tarefa dentre as habilidades dos recursos elegíveis. Definida a habilidade do recurso, o tempo desta habilidade é multiplicado pelo valor da penalidade para continuidade do processamento do algoritmo proposto.

A forma de definição da habilidade é semelhante à da definição da complexidade da tarefa para o recurso do tópico anterior, porém, neste caso, ocorre a inversão da busca de complexidade para habilidade.

Foram alteradas as parametrizações de habilidade 1 com complexidade 12 dos recursos 1 e 2 para habilidade 2 com complexidade 12. A parametrização da penalidade foi definida com valor 2, mantendo-se os demais parâmetros da base e do algoritmo

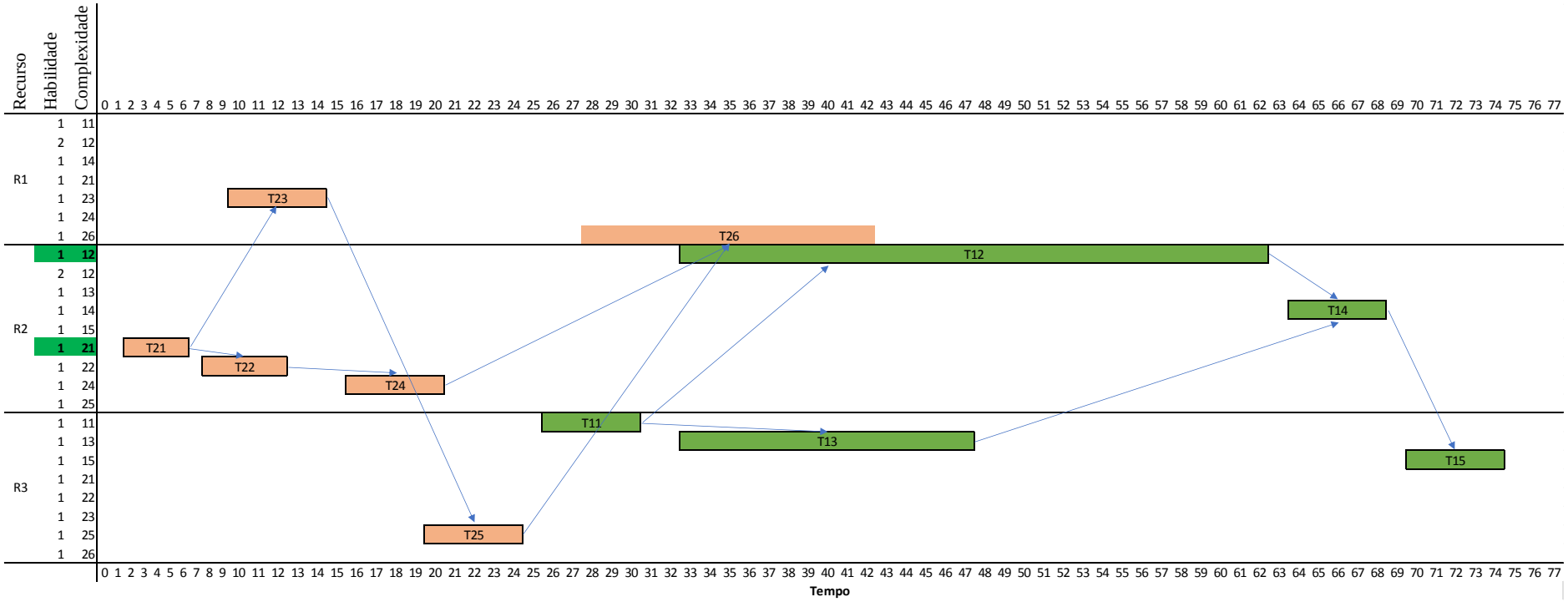
O tempo de processamento ficou entre 28 e 29 segundos, ou seja, o algoritmo consegue resolver a alocação de uma nova complexidade para o recurso dentro do tempo de previamente encontrado no item 5.2.1.

Na Figura 25 verifica-se a solução do algoritmo alocando uma habilidade para o Recurso 2, que originalmente não possuía a habilidade 1 com complexidade 12, porém possuía a habilidade 2 com complexidade 12. A habilidade 1 complexidade 12 poderia ter sido adicionada para recursos 1 e 2, visto que estes possuem a complexidade necessária. Para definir o valor a ser atribuído para a habilidade 2 com complexidade 12, o algoritmo escolheu a próxima complexidade semelhante do recurso, neste caso, definiu o valor 3.

Com isto, foi multiplicado o valor da penalidade vezes o valor da complexidade, ficando com o valor 6. Como o algoritmo está trabalhando com o valor de carga de trabalho igual a 5, o tempo total ficou igual a 30 para a tarefa T12.

Ainda conforme a Figura 25, verifica-se que o tempo total da solução para o algoritmo ficou em 77 unidades de tempo, condizente com os tempos encontrados anteriormente e aplicação da penalidade.

Figura 25. Validação do algoritmo com recurso sem habilidade para execução de determinada tarefa



Fonte: Autor

A próxima seção avalia a capacidade do algoritmo em resolver o problema considerando múltiplos objetivos.

5.1.9 Multiobjetivo

Essa seção considera com a execução do algoritmo com multiobjetivos, sendo o objetivo 1 como minimização do tempo total de execução, e o objetivo 2 a maximização do valor gerado por cada recurso. Para a solução com multiobjetivo, foi utilizado o algoritmo de decisão de Li *et al.* (2019) para definir a melhor solução atendendo os dois objetivos. O algoritmo de decisão realiza um cálculo baseado na melhor solução, na pior solução e a solução do processamento da iteração para cada um dos objetivos. O maior valor encontrado deste cálculo é considerado a melhor solução.

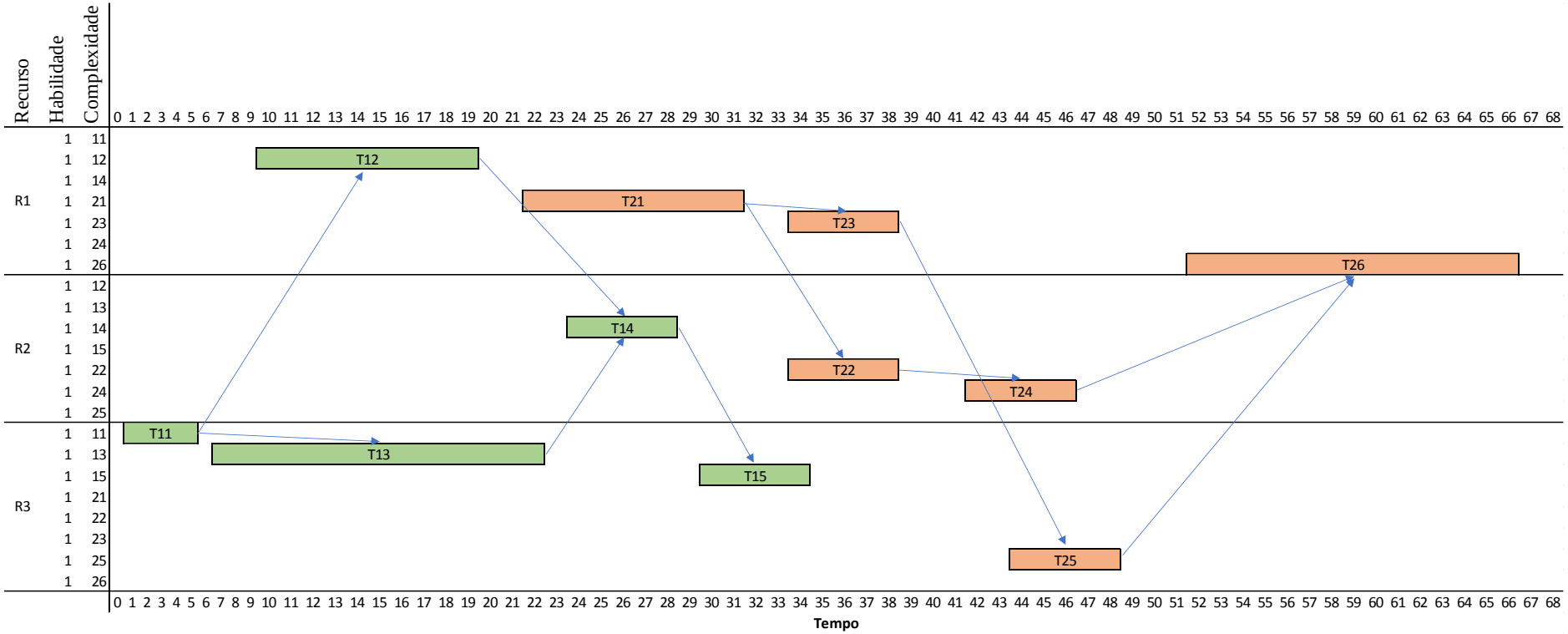
Para a execução do algoritmo, parametrizou-se os recursos com dados de valores com o objetivo 2. Parametrizou-se as tarefas dos projetos para que tenham os dados de inicialização para o segundo objetivo. Os dados utilizados para o objetivo 1 e as demais parametrizações foram mantidas conforme execuções anteriores.

A execução do algoritmo com a parametrização atual foi processada em 79 e 81 segundos. Era esperado um incremento no tempo de execução, pois o algoritmo precisa ser executado duas vezes, uma para cada objetivo parametrizado de forma sequencial, e depois realizar os cálculos do algoritmo de decisão da melhor solução.

O tempo esperado era para ser próximo ao dobro do tempo de execução, porém, como o ocorrido no item 5.2.1, o tempo foi superior ao dobro quando foi executado duas vezes de forma sequencial (Uma execução para cada objetivo), demonstrando que o algoritmo necessitará de ajustes para múltiplas execuções simultâneas.

Na Figura 26 verifica-se uma solução do algoritmo multiobjetivo demonstrando o tempo e o custo, visto que a parametrização do objetivo 2 foi realizada de forma idêntica ao objetivo 1.

Figura 26. Ilustração da solução obtida para o problema multiobjetivo



Fonte: Autor

Para terminar esta seção, segue um compilado do resultado das validações.

5.1.10 Resultado das validações

Durante o período de validação do algoritmo, foram realizados os ajustes necessários e reinício das validações até obter o resultado apresentado com a versão 6 do algoritmo proposto.

Nas validações, ficou evidente que o algoritmo precisará de ajustes para que possa processar de forma eficiente, pois para a execução das repetições, o tempo da próxima execução cresceu de forma quase exponencial, porém conseguindo escalonar as tarefas e alocar os recursos conforme proposto.

A integração com o modelo de simulação ocorreu conforme o desenvolvimento proposto, com a geração do arquivo pelo algoritmo. A criação, parametrização, processamento e geração do arquivo de saída do modelo de simulação foram realizadas de forma correta e os dados utilizados para o cálculo da solução. Quando o modelo de simulação foi habilitado para uma maior quantidade de projetos, tarefas e recursos, o tempo de processamento foi de cerca de 24 horas para uma simulação, o que torna inviável o processamento, necessitando de uma revisão do modelo de simulação para uma quantidade maior que dois projetos com 11 tarefas e 3 recursos, que são os dados obtidos com sucesso.

O processamento com os dados estocásticos ocorreu de forma correta, com o escalonamento das tarefas e alocação dos recursos conforme os dados calculados para cada recurso. Como o algoritmo salva os dados do processamento, é possível saber os tempos utilizados e o tempo total dos projetos.

De semelhante forma, o algoritmo escalonou as tarefas e realizou a alocação dos recursos aplicando penalidade no tempo total da solução final quando uma das tarefas necessita de uma habilidade ou uma complexidade que os recursos disponíveis não possuem.

Por fim, a função de multiobjetivo também atingiu a validação esperada, com a utilização do algoritmo de tomada de decisão, pois realizou o escalonamento das tarefas e alocação dos recursos, realizando o cálculo com um objetivo de minimizar o tempo total do sistema e outro de maximizar o valor gerado por cada recurso. No Quadro 11, segue o resultado das validações do algoritmo proposto.

Quadro 11. Resultado das validações do algoritmo integrado

Algoritmo / Modelo	Item validado	Status Validação
Algoritmo proposto	Escalonamento de tarefas	☑
	Escalonamento de projetos	☑
	Alocação de múltiplos recursos limitados	☑

Algoritmo / Modelo	Item validado	Status Validação
	Alocação de recursos com múltiplas habilidades	☑
	Tempo de transferência entre recursos	☑
	Tempo de inicialização da tarefa	☑
	Número de formigas variáveis para os projetos	☑
	Número de formigas variáveis para as tarefas	☑
	Peso do feromônio para os projetos	☑
	Peso do feromônio para as tarefas	☑
	Peso da heurística para os projetos	☑
	Peso da heurística para as tarefas	☑
	Taxa de evaporação do feromônio para os projetos	☑
	Taxa de evaporação do feromônio para as tarefas	☑
	Taxa de atualização global para os projetos	☑
	Taxa de atualização global para as tarefas	☑
	Número de iteração dos projetos	☑
	Número de iteração das tarefas	☑
	Múltiplos objetivos	☑
	Peso de avaliação por objetivo	☑
	Área de aplicação genérica	☑
	Tempo determinístico	☑
	Tempo estocástico	☑
	Tempo com curva de aprendizado	☑
	Acoplamento com modelo de simulação	☑
	Limitação do tempo de execução do modelo de simulação	☑
	Múltiplas execuções	☑
	Variação da complexidade	☑
	Variação da habilidade	☑
	Possibilidade de execução com recurso sem determinada habilidade	☑
	Possibilidade de execução com recurso sem determinada complexidade	☑
	Aplicação de penalidade, quando recurso sem habilidade ou complexidade	☑

Algoritmo / Modelo	Item validado	Status Validação
Modelo integrado	Todos os dados do algoritmo proposto	☑
	Leitura do arquivo de entrada	☑
	Parametrização automática do modelo	☑
	Criação automática do modelo	☑
	Dados de simulação na saída	☑

Fonte: Autor

Após todas as validações do algoritmo, realizou-se a comparação com os problemas correlatos da literatura, que são abordados na próxima sessão.

5.2 Comparativos com problemas da literatura correlata

Nesta seção, serão realizados os comparativos do algoritmo proposto com os problemas da literatura correlata, que são os problemas: RCMPSP, MSRCPSP e LFCM.

5.2.1 Problema RCMPSP

Para análise do algoritmo proposto em relação ao problema RCMPSP, foi utilizada, a base de dados RCMPSPLIB[7] disponibilizada por Vázquez, Calvo e Ordóñez (2013). A escolha da base se dá devido ao fato de ser uma base estabelecida e utilizada por outros autores ao longo do tempo, conforme revisão da literatura realizada sobre o RCMPSP.

A base RCMPSPLIB é composta de 5 diferentes instâncias com 10 tarefas em 10 projetos e 6 recursos, disponibilizadas com as respectivas soluções com o menor tempo de execução dos projetos.

5.2.1.1 Dados utilizados

As instâncias utilizadas para esse conjunto de experimentos contêm as seguintes informações:

- Número de projetos;
- Número de tarefas por projeto;
- Número de recursos disponíveis;
- Valor total de cada recurso;
- Duração de cada tarefa de cada projeto;

[7] Disponível no site https://www.eii.uva.es/elena/RCMPSPLIB.htm

- Recurso que pode atender cada tarefa;
- Matriz de precedência de cada tarefa em cada projeto.

Para a realização dos experimentos foi necessário realizar um ajuste na informação de duração de cada tarefa, pois o algoritmo proposto o tempo de execução é determinado pela habilidade do recurso alocada para a tarefa.

Foi utilizada a primeira base disponibilizada pelos autores Vázquez, Calvo e Ordóñez (2013), com um tempo total de execução mínimos dos projetos de 1.332 unidades segundo os autores. Os mesmos autores disponibilizam 300 combinações diferentes de soluções para este problema em uma planilha Excel.

Para este comparativo, os dados de transferência, inicialização e carga de trabalho foram definidos com o valor 1.

5.2.1.2 Parametrização do algoritmo proposto

A parametrização do algoritmo proposto foi alterada, reduzindo o número de repetições para 1 e o número de iterações das tarefas para 50, definidas conforme Li *et al.* (2019)

5.2.1.3 Resultados da comparação

A execução do algoritmo com a instância da base RCMPSPLIB escolhida, resultou nas seguintes combinações de soluções das tarefas:

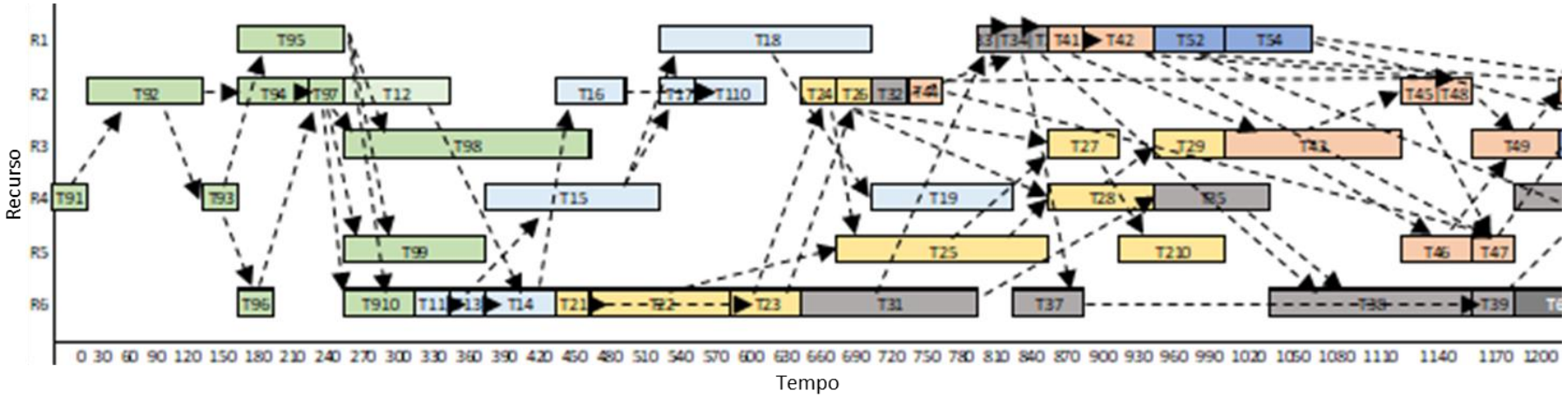
- Projeto 01 com 11 combinações;
- Projeto 02 com 06 combinações;
- Projeto 03 com 09 combinações;
- Projeto 04 com 15 combinações;
- Projeto 05 com 10 combinações
- Projeto 06 com 10 combinações;
- Projeto 07 com 14 combinações;
- Projeto 08 com 06 combinações;
- Projeto 09 com 06 combinações;
- Projeto 10 com 03 combinações

Na Figura 27 é apresentada uma solução do algoritmo proposta para os dados da base RCMPSPLIB escolhida para comparação. O tempo total de execução do algoritmo foi de 3.378 segundos, ou 56,3 minutos. Devido ao tamanho da solução, a

Figura 27 foi dívida nos itens a, b e c, sendo o item b é continuação da solução do item a, e o item c continuação da solução do item b.

Figura 27. Uma solução possível gerada pelo algoritmo proposto para a base RCMPSPLIB

a)



b)

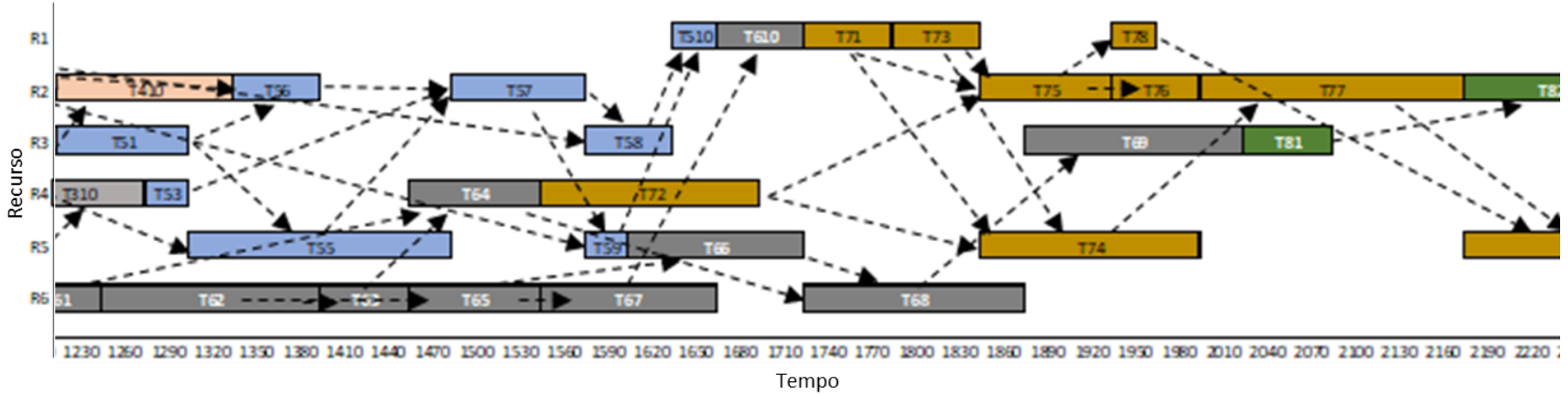
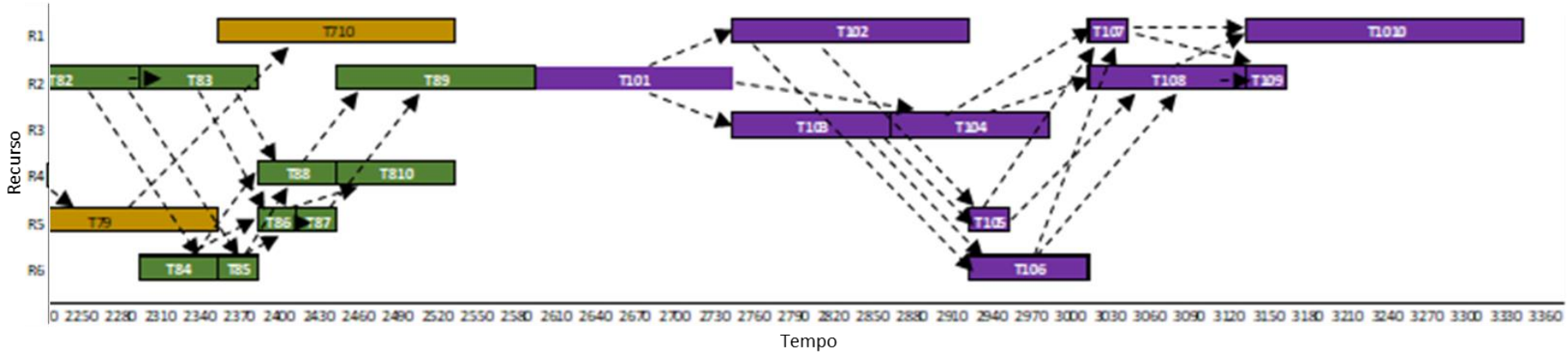


Figura 26. Uma solução possível gerada pelo algoritmo proposto para a base RCMPSPLIB

c)



Fonte: Autor

A Figura 27 demonstra o gráfico de Gantt gerado com a alocação dos recursos, o escalonamento das tarefas e das dependências entre as tarefas.

O algoritmo proposto foi capaz de encontrar uma solução factível para a instância, mas o valor obtido para o tempo necessário para execução de todas as tarefas do projeto foi de 3352 unidades de tempo, cerca de 2,5 vezes maior que o menor valor conhecido de 1332.

5.2.2 Problema MSRCPSP

Para análise do algoritmo proposto em relação ao problema MSRCPSP, foi utilizada a base iMOPSE[8]. A escolha da base se dá devido ao fato de ser uma base estabelecida e utilizada por outros autores ao longo do tempo, conforme revisão da literatura realizada sobre o MSRCPSP.

A geração da base é realizada através de um programa disponibilizado, e que pode ser parametrizado, porém não fornece nem as soluções possíveis e nem os menores *Makespan*.

5.2.2.1 Dados utilizados

O programa gerou os dados conforme a parametrização abaixo:

- Número de Tarefas: 99
- Duração mínima das tarefas: 8 horas
- Duração máxima das tarefas: 40 horas
- Relacionamentos: 50
- Atribuição dos recursos: Sim
- Número de recursos: 20
- Taxa mínima padrão: 10
- Taxa mínima de sobre alocação: 100
- Tipos mínimos de habilidades dos recursos: 1
- Tipos de habilidades: 9
- Taxa máxima padrão: 100
- Taxa máxima de sobre alocação: 200
- Tipos máximos de habilidades dos recursos: 9

Para este comparativo, os dados de transferência, inicialização e carga de trabalho foram definidos com o valor 1.

5.2.2.2 Parametrização do algoritmo proposto

A parametrização do algoritmo foi mantida em relação a comparação do RCMPSP.

[8] Disponível em http://imopse.ii.pwr.wroc.pl/

5.2.2.3 Resultados da comparação

A execução do algoritmo com os dados informados no algoritmo base instância, resultou nas seguintes combinações de soluções das tarefas:

- Projeto 01 com 19 combinações.

A Figura 31 do APÊNDICE III foi dívida nas partes a, b, c e d para poder demonstrar o gráfico de Gantt gerado a partir de uma das soluções oferecidas pelo algoritmo, com a alocação dos recursos, o escalonamento das tarefas e das dependências entre as tarefas, além de permitir a visualização das habilidades dos recursos. A parte b é sequência da parte de baixa da parte a, e com as demais sendo sequência da parte inferior da parte antecessora.

Na figura, pode-se verificar que existe apenas 1 projeto, e devido a este fato que todas as tarefas possuem a cor verde. Não houve sobreposição de recursos e tarefas, bem como as dependências entre as tarefas propostas pela base foram mantidas.

Uma solução possível gerada pelo algoritmo proposto é de 255 unidades de tempo, ou seja, o algoritmo desta tese consegue resolver o problema de forma factível, sequenciando as tarefas e alocando os recursos conforme as restrições do problema. O tempo total de execução deste algoritmo foi de 259.834 segundos, ou 72,16 horas (3 dias).

5.2.3 Problema LFCM

Para análise do algoritmo proposto em relação ao problema LFCM, foram utilizados os dados do artigo Chen *et al.* (2017) referentes a curva de aprendizado.

5.2.3.1 Dados utilizados

A instância utilizada nos experimentos dessa seção possuem as seguintes características, conforme Chen *et al.* (2017):

- Número de projetos: 10
- Precedência entre os projetos: Sim
- Habilidades dos recursos: 8
- Número de recursos: 40
- Valores iniciais das habilidades: Sim
- Número máximo de tarefas por projeto: 4

Os parâmetros de inicialização das tarefas, bem como os valores iniciais das habilidades foram definidas conforme os dados fornecidos no artigo. As habilidades possuem variação de 1 a 8, enquanto as complexidades variam de 1 a 10

5.2.3.2 Parametrização do algoritmo proposto

Foi alterado o tipo de dados a utilizar, trocando para curva estocástica com curva de aprendizado e esquecimento. Ao utilizar estes dados, parâmetros específicos para o processamento da curva de aprendizado e esquecimento precisaram ser parametrizados.

Os parâmetros específicos da curva de aprendizado são descritos a seguir:

- Valor inicial de eficiência: 0,5
- Percentual de aprendizado: 0,6
- Percentual de esquecimento: 0,4
- Percentual de impacto na avaliação do aprendizado: 0,05
- Limite de impacto da avaliação do aprendizado: 0,5
- Atualização da curva de aprendizado: 0

Os parâmetros de percentual de impacto na avaliação do aprendizado, limite de impacto da avaliação do aprendizado e atualização da curva de aprendizado foram introduzidos no algoritmo proposto, não existindo no algoritmo original de LFMC proposto por Chen *et al.* (2017). Tais parâmetros foram criados para limitar: o efeito do uso excessivo ou da falta de uso do recurso por longos períodos; os efeitos do aprendizado nos tempos totais de execução; e melhorar o desempenho computacional, respectivamente. Os demais parâmetros do algoritmo proposto não foram alterados.

5.2.3.3 Resultados da comparação

A execução do algoritmo com os dados informados no algoritmo base instância, resultou nas seguintes combinações de soluções das tarefas:

- Projeto 01 com 03 combinações;
- Projeto 02 com 03 combinações;
- Projeto 03 com 01 combinação;
- Projeto 04 com 02 combinações;
- Projeto 05 com 04 combinações;

- Projeto 06 com 04 combinações;
- Projeto 07 com 04 combinações;
- Projeto 08 com 03 combinações;
- Projeto 09 com 01 combinação;
- Projeto 10 com 01 combinação.

A Figura 35 do APÊNDICE III foi dívida nas partes a, b, c e d para poder demonstrar o gráfico de Gantt gerado a partir de uma das soluções oferecidas pelo algoritmo, com a alocação dos recursos, o escalonamento das tarefas e das dependências entre as tarefas, além de permitir a visualização das habilidades dos recursos. A parte b é sequência da parte de baixa da parte a, e com as demais sendo sequência da parte inferior da parte antecessora.

A solução obtida considera a alocação dos recursos, o escalonamento das tarefas e das dependências entre as tarefas, além das habilidades dos recursos. O algoritmo foi capaz de gerar uma solução factível, na qual não houve sobreposição de recursos e tarefas, bem como as dependências entre as tarefas propostas pela base foram mantidas, além de dependência existente entre os projetos

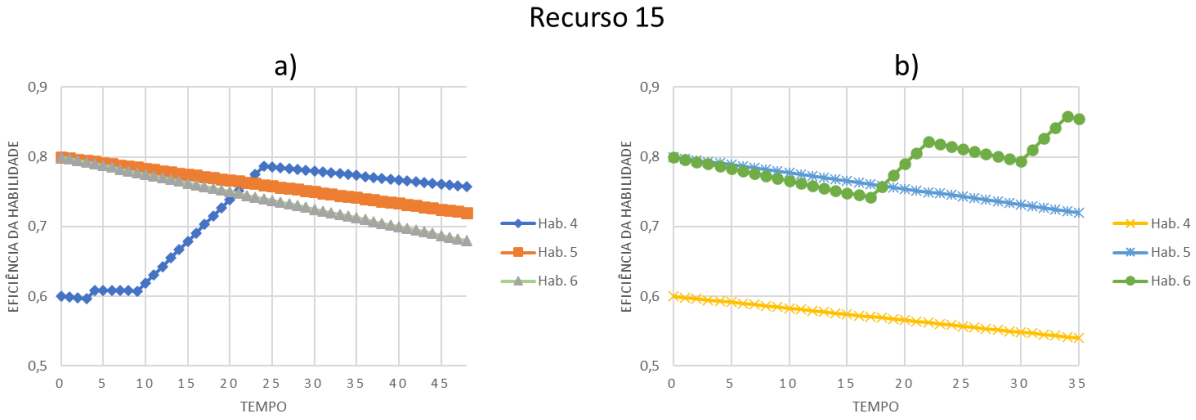
A solução oferecida pelo algoritmo é de 47,2 unidades de tempo, ou seja, o algoritmo proposto consegue resolver o problema de forma factível, sequenciando as tarefas e alocando os recursos conforme as restrições do problema, além de calcular a curva de aprendizado. O algoritmo original teve um tempo total de 34,9 unidades de tempo, ou seja, um tempo 27% menor que o algoritmo proposto.

O tempo total de execução do algoritmo proposto foi de 26.135 segundos, ou 7,25 horas. Este tempo é muito superior ao obtido com o algoritmo para a base RCMPSP que possui o mesmo número de projetos e de tarefas, porém com uma quantidade bem inferior de recursos (3 ao invés de 40 para esta base). Ou seja, o fator da quantidade de recursos disponíveis gera um impacto direto no tempo de processamento do algoritmo. Um outro fator de diferença entre as execuções é o tipo de dado utilizado, pois quando executado para a base RCMPSP, foi utilizado um dado determinístico, e para a base LFCM, foi utilizado um dado estocástico que precisa de recalculo a cada iteração das tarefas, gerando impacto no tempo de processamento.

Além da informação de escalonamento das tarefas com a alocação dos recursos, foi comparada a curva de aprendizado dos recursos. Na Figura 28(a)

verifica-se o ganho de eficiência do recurso 15 no artigo de Chen *et al.* (2017) de comparação do LFCM e na Figura 28(b) têm-se o ganho de eficiência do mesmo recurso para o algoritmo proposto. O recurso 15 foi comparado, pois é o único que possui dados no artigo e que foi alocado pelo algoritmo proposto.

Figura 28. Ganho de eficiência do Recurso 15



Fonte: Adaptado de Chen *et al.* (2017) e autor

Verifica-se no comparativo entre as duas figuras anteriores que o tempo total é diferente, enquanto no algoritmo original o tempo total foi de 35 unidades de tempo, o tempo total no algoritmo proposto foi de 47 unidades de tempo. Além disso, nota-se que a habilidade utilizada nos dois algoritmos é diferente. No algoritmo original, foi utilizada a habilidade 4, enquanto na execução do algoritmo proposto, foi utilizada a habilidade 6. A semelhança é quanto a funcionalidade da curva, que demonstra nos dois algoritmos que, quando a habilidade é utilizada, a eficiência do recurso aumenta, e quando não é utilizada, reduz.

Com os dados desta figura, fica evidente o impacto causado pela curva de aprendizado e esquecimento. Quando ocorre a utilização do recurso, a habilidade tem um salto positivo, fazendo com que o recurso execute a tarefa com um tempo menor. No inverso, quando o recurso não é utilizado, ocorre o decréscimo da eficiência da habilidade, que impactará no aumento do tempo de execução da tarefa. Um ponto importante, é que o aumento é muito mais significativo que a queda, ou seja, o impacto positivo da utilização do recurso é muito maior que o impacto negativa quando deixa de ser utilizado.

No próximo item, seguem os resultados das validações e comparativos.

5.3 Resultado dos comparativos

Nos comparativos do algoritmo proposto com os algoritmos originais, demonstrou-se que o algoritmo proposto consegue atender os requisitos propostos, ou seja, conseguiu atender de forma integral e unificada os problemas RCMPSP, MSRCPSP e LFCM, pois conseguiu alocar os recursos, escalonando as tarefas e atendendo todas as restrições e precedências entre os projetos e tarefas.

No Quadro 12, segue o resultado das comparações com a literatura correlata do algoritmo proposto.

Quadro 12. Resultado comparativo com a literatura correlata

Problema	Comparação	Comparativo
Algoritmo de Li *et al.* (2019)	Algoritmo proposto resolve o problema proposto da mesma forma que o algoritmo original?	☑
RCMPSP	Algoritmo proposto consegue resolver o problema RCMPSP, escalonando as tarefas e alocando os recursos em múltiplos projetos?	☑
MSRCPSP	Algoritmo proposto consegue resolver o problema MSRCPSP, escalonando as tarefas e alocando os recursos com em múltiplas habilidades em projeto?	☑
LFCM	Algoritmo proposto consegue resolver o problema LFCM, escalonando as tarefas e alocando os recursos com em múltiplas habilidades em projetos considerando a curva de aprendizado e esquecimento dos recursos?	☑

Fonte: Autor

Na próxima seção segue a conclusão desta tese, com as contribuições para a área, limitações da pesquisa e sugestão de trabalhos futuros.

6 Conclusão

Esta pesquisa visou contribuir com o conhecimento teórico, que ainda é pouco explorado no meio acadêmico com a unificação de três problemas que costumam ser tratados de forma isolada em um único algoritmo, que são: o escalonamento de tarefas em múltiplos projetos, a alocação de recursos com múltiplas habilidades em projetos, e a curva de aprendizado e esquecimento dos recursos ao longo do tempo. Esta junção acrescenta a possibilidade de troca de recursos ao longo do tempo e/ou alocação de recursos externos a empresa que o escalonamento das tarefas está sendo realizado, de forma temporária.

A junção dos três problemas visa simular o comportamento que é verificado no mundo real, nos quais estes problemas ocorrem diariamente, e com alocações sendo realizadas de forma manual com baixa eficiência, como por exemplo nos resultados apresentados em Santos (2017), que possuía uma baixa eficiência de alocação, e que com o modelo de simulação e otimização de recursos em múltiplos projetos conseguiu otimizar o tempo de alocação dos recursos e diminuir o tempo total de alocação nos projetos.

Além da evolução citada acima, o algoritmo proposto é multiobjetivo, podendo ter diversos objetivos sendo processados de forma única. Este multiobjetivo não estava sendo considerado anteriormente, tendo-se apenas avaliação no tempo do projeto, sendo que, outros objetivos, como os custos, possuem relação direta com os tempos de execução dos projetos e com os recursos alocados nos projetos, e que, conforme a literatura, possui uma relação inversamente proporcional com a curva de aprendizado.

Analisando os resultados comparativos e os resultados de validação, conclui-se que o algoritmo proposto é eficaz e atende todas as restrições impostas por todos os algoritmos utilizados como base. O algoritmo proposto é eficiente para solução de poucos projetos e com poucas tarefas, além de poucos recursos e que não possuem uma grande variação das habilidades e complexidades, porém precisa de melhorias tanto em tempo de processamento, quanto nas soluções propostas quando os dados utilizados para processamento incluem uma grande quantidade de projetos, ou de tarefas, ou de recursos.

Ao retomar os objetivos desta tese, segue resumo no Quadro 13, identificando que todos os objetivos da tese foram atendidos.

Quadro 13. Resultado objetivo da tese

Tipo	Objetivo	Resultado
Geral	Propor um novo algoritmo de otimização integrado com um modelo de simulação para solução do problema integrado de escalonamento de tarefas e alocação de recursos em múltiplos projetos, considerando a dependência entre tarefas e recursos com múltiplas habilidade e curva de aprendizado	☑
Específico	Desenvolver um modelo de simulação e um algoritmo de otimização para a alocação de recursos e escalonamento de tarefas de forma integrada	☑
Específico	Propor uma versão multiobjetivo da metaheurística colônia de formigas para solução do problema integrado	☑

Fonte: Autor

6.1 Contribuição para a área

A tese contribui com a literatura ao criar um modelo de simulação integrado com um algoritmo de otimização para resolução dos três problemas estudados de forma conjunta, que possui poucas referências na literatura, tratando do escalonamento dos recursos e suas habilidades, além do acréscimo da possibilidade de utilização de complexidade da tarefa que o recurso pode executar.

Ao investigar como a curva de aprendizado impacta na alocação dos recursos nos diversos níveis de hierarquia, em múltiplos projetos, o estudo permitiu entender a relação entre a curva de aprendizado dos recursos e os objetivos definidos, como por exemplo o custo e o prazo dos projetos, avaliando as condições nas quais o impacto do aprendizado pode comprometer o sucesso do projeto nesse contexto. Dada a originalidade da proposta, espera-se que após a conclusão da tese um artigo resultante do trabalho seja publicado em periódico internacional. Ademais, evoluções parciais do trabalho poderão ser utilizadas em trabalhos futuros.

Nesta tese, o algoritmo proposto utilizada duas camadas para resolução dos problemas integrados, com a primeira camada resolvendo o problema de escalonamento dos projetos (Devido a possibilidade de múltiplos projetos e podendo ter interdependência entre os projetos) e a segunda camada, resolvendo o problema de alocação dos recursos para resolução das tarefas em cada projeto.

O algoritmo proposto possibilita o processamento multiobjetivos (por exemplo avaliar o tempo e custo), com a avaliação de peso para cada objetivo e variação de

tempo de execução de cada tarefa, além de ser adaptado para trabalhar com projetos, tarefas e recursos humanos, com restrição de recursos, que passam a ser restritos e que podem ser processados com múltiplas habilidades e níveis de proficiência nas diversas habilidades. Neste trabalho, os níveis de proficiência podem aumentar ou diminuir ao longo do tempo, conforme a curva de aprendizado.

O algoritmo proposto possui a possibilidade de processamento com: dependência entre os projetos; possibilidade de priorização de projetos e por fim, o escalonamento dos projetos.

Por fim, a última evolução no algoritmo integrado é o acoplamento do algoritmo de otimização com o modelo de simulação, que foi construído no Software Arena® e detalhada na próxima seção, com a finalidade de simular a alocação dos recursos nos projetos.

Além da contribuição teórica, ampliar a compreensão da relação entre o aprendizado dos recursos e o problema de alocação pode trazer uma contribuição para as empresas de software que rotineiramente enfrentam essa situação. Nesse contexto, o trabalho contribui com o meio profissional, pois criou um software de alocação de múltiplos recursos em múltiplos projetos que, espera-se, possa ser utilizado de forma prática no meio profissional, já que existe uma carência de softwares que realizem a alocação de recursos que possuem múltiplas habilidades em múltiplos projetos levando em consideração o grau de conhecimento atual dos recursos e a curva de aprendizado ao longo do tempo. Será solicitada uma proteção intelectual e registro do software gerado pelo do modelo de simulação e algoritmo de otimização desenvolvido.

6.2 Limitações da pesquisa

O algoritmo de otimização utiliza sempre a quantidade de um recurso por recurso, ou seja, caso exista mais de um recurso com a mesma habilidade, a mesma complexidade e possua os mesmos valores para cada objetivo, este recurso precisa ser parametrizado *n* vezes conforme o número desejado alterando-se a identificação do recurso, o que acaba por dificultar a parametrização dos recursos, pois é necessário replicar uma informação por diversas vezes.

O modelo de simulação foi construído na ferramenta Arena®, ferramenta esta que possui licenciamento para a utilização, e que pode acarretar altos custos para que

a solução seja utilizada. A versão de estudante possui um licenciamento diferente, com a liberação de uso, porém com uma limitação dos modelos a serem construídos e utilizados.

O modelo de simulação permite a simulação a partir de dados determinísticos e dados estocásticos. No modelo de simulação construído, estamos utilizando os dados determinísticos para a simulação de projetos que são processados com curvas estocásticas de aprendizado e esquecimento, por não ter sido implementado no software de simulação o modelo de cálculo da curva de aprendizado e esquecimento.

Sendo assim, caso a parametrização do algoritmo completo seja com dados determinísticos, tanto o algoritmo de otimização, quanto o modelo de simulação utilizarão dados determinísticos. Quando parametrizado para utilizar dados estocásticos da curva de probabilidade exponencial, tanto o algoritmo de otimização, quanto o modelo de simulação utilizarão dados estocásticos. Quando parametrizado para utilizar a curva de aprendizado e esquecimento, o algoritmo de otimização utilizará os dados estocásticos produzidos, porém o modelo de simulação utilizará o último dado estocástico gerado pelo algoritmo de otimização de forma determinística.

A última limitação identificada no algoritmo é em a origem do dado para cálculo dos algoritmos. Nas bases utilizadas para comparação do RCMPSP e do MSRCPSP, a origem do valor de cálculo do objetivo está na tarefa, ou seja, as bases RCMPSPLIB e iMOPSE determinam o valor que a tarefa tem para ser executada, como por exemplo, o tempo de execução de uma tarefa. No algoritmo construído nesta tese, a origem do valor está no recurso que executa a tarefa. Como exemplo, um recurso que tenha determinada habilidade e consiga atender uma tarefa de complexidade alta, possui o tempo de 5 unidades, enquanto um segundo recurso, que possui a mesma habilidade, e também consiga atender uma tarefa de complexidade alta, possua 7 unidades de tempo para o atendimento, o tempo final de atendimento da tarefa poderá ser impactado com uma diferença de 2 unidades de tempo, fazendo com que seja difícil comparar com as bases, pois o tempo pode ter sido definido como 5 unidades para executar a tarefa, ou como 7 unidades.

6.3 Sugestões para trabalhos futuros

Como sugestão para continuidade deste trabalho, têm-se os apontamentos:

- ✓ Paralelizar o processamento;

- ✓ Acrescentar restrições de calendário;
- ✓ Otimização do código;
- ✓ Revisão da quantidade de recursos.

O algoritmo integrado de otimização e o modelo de simulação estão sendo processados de forma sequencial, gerando um tempo alto de processamento para bases maiores. A sugestão, é que seja processado de forma paralela, diminuindo o tempo total de processamento do algoritmo de forma significativa.

Além disto, pode-se acrescentar novas restrições de tempo de trabalho diário, calendários de trabalho, percentual aceitável de horas extras, que permitirá uma representação maior da realidade das empresas, com a necessidade de novos fluxos e restrições de processamento, aumentando a complexidade atual do algoritmo.

Necessário uma revisão para otimização do código e do tempo de processamento em geral, conhecido como refatoração do código, permitindo um consumo menor de recursos computacionais e entregas mais rápidas dos dados para o usuário do algoritmo. Além disto, pode-se adequar o software para trabalhar com micro serviços e processamento na nuvem.

A quantidade de recursos pode ser considerada durante o escalonamento e associação as tarefas, assim, o algoritmo de otimização pode ser utilizado de forma mais simples para outras áreas, como por exemplo de otimização de tarefas fabris, que os recursos (máquinas) possuem as mesmas características, não precisando replicar em diversas linhas, pois o recurso que vai processar, não precisa ser identificado, precisa apenas que tenha o recurso disponível, desde que no mesmo espaço físico.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLWOOD, J. M.; LEE, W. L. The impact of job rotation on problem solving skills. **International Journal of Production Research**, *[S. l.]*, v. 42, n. 5, p. 865–881, 2004. DOI: 10.1080/00207540310001631566.

ALMEIDA, Bernardo F.; CORREIA, Isabel; SALDANHA-DA-GAMA, Francisco. Priority-based heuristics for the multi-skill resource constrained project scheduling problem. **Expert Systems with Applications**, *[S. l.]*, v. 57, p. 91–103, 2016. DOI: 10.1016/j.eswa.2016.03.017. Disponível em: http://dx.doi.org/10.1016/j.eswa.2016.03.017.

ÁLVAREZ, Alejandro; PATIÑO, Albeiro. Enterprise architecture and agile methodologies - An effective combination to tackle the frequent business changes Arquitectura empresarial y metodologías ágiles - Una combinación efectiva para hacer frente a los frecuentes cambios en el negocio. *[S. l.]*, v. 1, p. 145–152, 2015.

ANZANELLO, Michel Jose; FOGLIATTO, Flavio Sanson. Learning curve models and applications: Literature review and research directions. **International Journal of Industrial Ergonomics**, *[S. l.]*, v. 41, n. 5, p. 573–583, 2011. DOI: 10.1016/j.ergon.2011.05.001. Disponível em: http://dx.doi.org/10.1016/j.ergon.2011.05.001.

ATTIA, El Awady; DUQUENNE, Philippe; LE-LANN, Jean Marc. Considering skills evolutions in multi-skilled workforce allocation with flexible working hours. **International Journal of Production Research**, *[S. l.]*, v. 52, n. 15, p. 4548–4573, 2014. DOI: 10.1080/00207543.2013.877613. Disponível em: http://dx.doi.org/10.1080/00207543.2013.877613.

BADIRU, A. B. Multivariate analysis of the effect of learning and forgetting on product quality. **International Journal of Production Research**, *[S. l.]*, v. 33, n. 3, p. 777–794, 1995. DOI: 10.1080/00207549508930179.

BEN ISSA, Samer; PATTERSON, Raymond A.; TU, Yiliu. Solving resource-constrained multi-project environment under different activity assumptions. **International Journal of Production Economics**, *[S. l.]*, v. 232, p. 107936, 2021. DOI: 10.1016/j.ijpe.2020.107936. Disponível em: https://doi.org/10.1016/j.ijpe.2020.107936.

BERRICHI, A.; YALAOUI, F.; AMODEO, L.; MEZGHICHE, M. Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling. **Computers and Operations Research**, *[S. l.]*, v. 37, n. 9, p. 1584–1596, 2010. DOI: 10.1016/j.cor.2009.11.017. Disponível em: http://dx.doi.org/10.1016/j.cor.2009.11.017.

BERTRAND, J. Will M.; FRANSOO, Jan C. Operations management research methodologies using quantitative modeling. **International Journal of Operations and Production Management**, *[S. l.]*, v. 22, n. 2, p. 241–264, 2002. DOI: 10.1108/01443570210414338.

BROWNING, Tyson R. T. R.; YASSINE, Ali A. A. A random generator of resource-constrained multi-project network problems. **Journal of Scheduling**, *[S. l.]*, v. 13, n. 2, p. 143–161, 2010. a. DOI: 10.1007/s10951-009-0131-y. Disponível em: http://link.springer.com/10.1007/s10951-009-0131-y.

BROWNING, Tyson R.; YASSINE, Ali A. A random generator of resource-constrained multi-project network problems. **Journal of Scheduling**, *[S. l.]*, v. 13, n. 2, p. 143–161, 2010. b. DOI: 10.1007/s10951-009-0131-y.

CAVAGNINI, Rossana; HEWITT, Mike; MAGGIONI, Francesca. Workforce production planning under uncertain learning rates. **International Journal of Production Economics**, *[S. l.]*, v. 225, n. December 2019, p. 107590, 2020. DOI: 10.1016/j.ijpe.2019.107590. Disponível em: https://doi.org/10.1016/j.ijpe.2019.107590.

CHEN, H.; DING, G.; ZHANG, J.; QIN, S. Research on priority rules for the stochastic resource constrained multi-project scheduling problem with new project arrival. **Computers and Industrial Engineering**, *[S. l.]*, v. 137, 2019. a. DOI: 10.1016/j.cie.2019.106060.

CHEN, Rong; LIANG, Changyong; GU, Dongxiao; LEUNG, Joseph Y. T. A multi-objective model for multi-project scheduling and multi-skilled staff assignment for IT product development considering competency evolution. **International Journal of Production Research**, *[S. l.]*, v. 55, n. 21, p. 6207–6234, 2017. DOI: 10.1080/00207543.2017.1326641. Disponível em:

https://doi.org/10.1080/00207543.2017.1326641.

CHEN, W.; ZHANG, J. Ant colony optimization for software project scheduling and staffing with an event-based scheduler. **IEEE Transactions on Software Engineering**, *[S. l.]*, v. 39, n. 1, p. 1–17, 2013. DOI: 10.1109/TSE.2012.17.

CHEN, Zong Gan et al. Multiobjective Cloud Workflow Scheduling: A Multiple Populations Ant Colony System Approach. **IEEE Transactions on Cybernetics**, *[S. l.]*, v. 49, n. 8, p. 2912–2926, 2019. b. DOI: 10.1109/TCYB.2018.2832640. Disponível em: https://ieeexplore.ieee.org/document/8360973/.

CHIU, Huan Neng; TSAI, Deng Maw. An efficient search procedure for the resource-constrained multi-project scheduling problem with discounted cash flows. **Construction Management and Economics**, *[S. l.]*, v. 20, n. 1, p. 55–66, 2002. DOI: 10.1080/01446190110089718. Disponível em: http://www.tandfonline.com/doi/abs/10.1080/01446190110089718.

CHIU, Tsai D. M. An efficient search procedure for the resource-constrained multi-project scheduling problem with discounted cash flows. **Construction Management and Economics**, *[S. l.]*, v. 20, n. 1, p. 55–66, 2002. DOI: 10.1080/01446190110089718.

CHOFFIN, Benoît; POPINEAU, Fabrice; BOURDA, Yolaine; VIE, Jill Jênn. DAS3H: Modeling student learning and forgetting for optimally scheduling distributed practice of skills. **EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining**, *[S. l.]*, p. 29–38, 2019.

DAI, Huafeng; CHENG, Wenming; YANG, Wucheng; WANG, Yupu. A general variable neighbourhood search for multi-skill resource-constrained project scheduling problem with step-deterioration. **International Journal of Industrial and Systems Engineering**, *[S. l.]*, v. 34, n. 2, p. 145–164, 2020. DOI: 10.1504/IJISE.2020.105288.

DALFARD, V. M.; RANJBAR, V. Multi-projects scheduling with resource constraints & priority rules by the use of simulated annealing algorithm | Programiranje više projekata uz ograničena sredstva & Pravila prioriteta primjenom algoritma simuliranog žarenja. **Tehnicki Vjesnik**, *[S. l.]*, v. 19, n. 3, p. 493–499, 2012.

DANG QUOC, Huu; NGUYEN THE, Loc; NGUYEN DOAN, Cuong; XIONG, Naixue. Effective Evolutionary Algorithm for Solving the Real-Resource-Constrained Scheduling Problem. **Journal of Advanced Transportation**, *[S. l.]*, v. 2020, 2020. DOI: 10.1155/2020/8897710.

DAR-EL, E. M.; AYAS, K.; GILAD, I. Predicting performance times for long cycle time tasks. **IIE Transactions (Institute of Industrial Engineers)**, *[S. l.]*, v. 27, n. 3, p. 272–281, 1995. DOI: 10.1080/07408179508936741.

DENG, Wu; XU, Junjie; ZHAO, Huimin. An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem. **IEEE Access**, *[S. l.]*, v. 7, p. 20281–20292, 2019. DOI: 10.1109/ACCESS.2019.2897580.

DONG, Ning; GE, Dongdong; FISCHER, Martin; HADDAD, Zuhair. A genetic algorithm-based method for look-ahead scheduling in the finishing phase of construction projects. **ADVANCED ENGINEERING INFORMATICS**, THE BOULEVARD, LANGFORD LANE, KIDLINGTON, OXFORD OX5 1GB, OXON, ENGLAND, v. 26, n. 4, p. 737–748, 2012. DOI: 10.1016/j.aei.2012.03.004.

DORIGO, Marco; MANIEZZO, Vittorio; COLORNI, Alberto. Ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, *[S. l.]*, v. 26, n. 1, p. 29–41, 1996. DOI: 10.1109/3477.484436. Disponível em: papers://82ac23f7-2eaf-4339-a5e1-4600c19d7f01/Paper/p2331.

FERNANDEZ-VIAGAS, Victor; FRAMINAN, Jose M. Integrated project scheduling and staff assignment with controllable processing times. **Scientific World Journal**, *[S. l.]*, v. 2014, 2014. DOI: 10.1155/2014/924120.

FERRANDI, Fabrizio; LANZI, Pier Luca; PILATO, Christian; SCIUTO, Donatella; TUMEO, Antonino. Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, *[S. l.]*, v. 29, n. 6, p. 911–924, 2010. DOI: 10.1109/TCAD.2010.2048354.

GAGNÉ, C.; PRICE, W. L.; GRAVEL, M. **Comparing an ACO algorithm with other heuristics for**

the single machine scheduling problem with sequence-dependent setup timesJournal of the Operational Research Society, 2002. DOI: 10.1057/palgrave.jors.2601390.

GAN, Wenbin; SUN, Yuan; PENG, Xian; SUN, Yi. Modeling learner's dynamic knowledge construction procedure and cognitive item difficulty for knowledge tracing. **Applied Intelligence**, *[S. l.]*, v. 50, n. 11, p. 3894–3912, 2020. DOI: 10.1007/s10489-020-01756-7.

GANJI, Maliheh; KAZEMIPOOR, Hamed; HADJI MOLANA, Seyyed Mohammad; SAJADI, Seyed Mojtaba. A green multi-objective integrated scheduling of production and distribution with heterogeneous fleet vehicle routing and time windows. **Journal of Cleaner Production**, *[S. l.]*, v. 259, p. 120824, 2020. DOI: 10.1016/j.jclepro.2020.120824. Disponível em: https://doi.org/10.1016/j.jclepro.2020.120824.

GRÄSSLER, Iris; ROESMANN, Daniel; CAPPELLO, Chiara; STEFFEN, Eckhard. Skill-based worker assignment in a manual assembly line. **Procedia CIRP**, *[S. l.]*, v. 100, p. 433–438, 2021. DOI: 10.1016/j.procir.2021.05.100. Disponível em: https://doi.org/10.1016/j.procir.2021.05.100.

GUO, Y.; JI, J.; JI, J.; GONG, D.; CHENG, J.; SHEN, X. Firework-based software project scheduling method considering the learning and forgetting effect. **Soft Computing**, *[S. l.]*, v. 23, n. 13, p. 5019–5034, 2019. DOI: 10.1007/s00500-018-3165-2. Disponível em: https://doi.org/10.1007/s00500-018-3165-2.

HEIMERL, C.; KOLISCH, R. Work assignment to and qualification of multi-skilled human resources under knowledge depreciation and company skill level targets. **International Journal of Production Research**, *[S. l.]*, v. 48, n. 13, p. 3759–3781, 2010. DOI: 10.1080/00207540902852785.

HEMATIAN, Milad; SEYYED ESFAHANI, Mir Mehdi; MAHDAVI, Iraj; MAHDAVI-AMIRI, Nezam; REZAEIAN, Javad. A multiobjective integrated multiproject scheduling and multiskilled workforce assignment model considering learning effect under uncertainty. **Computational Intelligence**, *[S. l.]*, v. 36, n. 1, p. 276–296, 2020. DOI: 10.1111/coin.12260. Disponível em: https://onlinelibrary.wiley.com/doi/10.1111/coin.12260.

HOEDT, Steven; CLAEYS, Arno; AGHEZZAF, El Houssaine; COTTYN, Johannes. Real time implementation of learning-forgetting models for cycle time predictions of manual assembly tasks after a break. **Sustainability (Switzerland)**, *[S. l.]*, v. 12, n. 14, 2020. DOI: 10.3390/su12145543.

HOFFMANN, Luise Sophie; KELLENBRINK, Carolin; HELBER, Stefan. **Simultaneous structuring and scheduling of multiple projects with flexible project structures**. [s.l.] : Springer Berlin Heidelberg, 2020. v. 90 DOI: 10.1007/s11573-020-00993-z. Disponível em: https://doi.org/10.1007/s11573-020-00993-z.

HOSSEINIAN, A. H.; BARADARAN, V. P-GWO and MOFA: two new algorithms for the MSRCPSP with the deterioration effect and financial constraints (case study of a gas treating company). **Applied Intelligence**, *[S. l.]*, v. 50, n. 7, p. 2151–2176, 2020. DOI: 10.1007/s10489-020-01663-x.

HOSSEINIAN, Amir Hossein A. H.; BARADARAN, Vahid; BASHIRI, Mahdi. Modeling of the time-dependent multi-skilled RCPSP considering learning effect. **Journal of Modelling in Management**, *[S. l.]*, v. 14, n. 2, p. 521–558, 2019. a. DOI: 10.1108/JM2-07-2018-0098. Disponível em: https://www.emerald.com/insight/content/doi/10.1108/JM2-07-2018-0098/full/html.

HOSSEINIAN, Amir Hossein; BARADARAN, Vahid; BASHIRI, Mahdi. Modeling of the time-dependent multi-skilled RCPSP considering learning effect: An evolutionary solution approach. **Journal of Modelling in Management**, *[S. l.]*, v. 14, n. 2, p. 521–558, 2019. b. DOI: 10.1108/JM2-07-2018-0098.

HUANG, Shan Huen; LIN, Pei Chun. A modified ant colony optimization algorithm for multi-item inventory routing problems with demand uncertainty. **Transportation Research Part E: Logistics and Transportation Review**, *[S. l.]*, v. 46, n. 5, p. 598–611, 2010. DOI: 10.1016/j.tre.2010.01.006. Disponível em: http://dx.doi.org/10.1016/j.tre.2010.01.006.

HUSSEIN, Mohamed K.; MOUSA, Mohamed H.; ALQARNI, Mohamed A. A placement architecture for a container as a service (CaaS) in a cloud environment. **Journal of Cloud Computing**, *[S. l.]*, v. 8, n. 1, p. 1–15, 2019. DOI: 10.1186/s13677-019-0131-1.

JABER, M. Y.; PELTOKORPI, J.; GLOCK, C. H.; GROSSE, E. H.; PUSIC, M. Adjustment for cognitive interference enhances the predictability of the power learning curve. **International Journal of**

Production Economics, *[S. l.]*, v. 234, n. August 2020, p. 108045, 2021. DOI: 10.1016/j.ijpe.2021.108045. Disponível em: https://doi.org/10.1016/j.ijpe.2021.108045.

JABER, Mohamad Y.; BONNEY, Maurice. Production breaks and the learning curve: The forgetting phenomenon. **Applied Mathematical Modelling**, *[S. l.]*, v. 20, n. 2, p. 162–169, 1996. DOI: 10.1016/0307-904X(95)00157-F.

JABER, Mohamad Y.; BONNEY, Maurice. A comparative study of learning curves with forgetting. **Applied Mathematical Modelling**, *[S. l.]*, v. 21, n. 8, p. 523–531, 1997. DOI: 10.1016/S0307-904X(97)00055-3.

JABER, Mohamad Y.; BONNEY, Maurice. Lot sizing with learning and forgetting in set-ups and in product quality. **International Journal of Production Economics**, *[S. l.]*, v. 83, n. 1, p. 95–111, 2003. DOI: 10.1016/S0925-5273(02)00322-5.

JABER, Mohamad Y.; KHER, Hemant V.; DAVIS, Darwin J. Countering forgetting through training and deployment. **International Journal of Production Economics**, *[S. l.]*, v. 85, n. 1, p. 33–46, 2003. DOI: 10.1016/S0925-5273(03)00084-7.

JABER, Mohamad Y.; SIKSTRÖM, Sverker. A numerical comparison of three potential learning and forgetting models. **International Journal of Production Economics**, *[S. l.]*, v. 92, n. 3, p. 281–294, 2004. DOI: 10.1016/j.ijpe.2003.10.019.

JERIĆ, Silvija Vlah; FIGUEIRA, José Rui. Multi-objective scheduling and a resource allocation problem in hospitals. **Journal of Scheduling**, *[S. l.]*, v. 15, n. 5, p. 513–535, 2012. DOI: 10.1007/s10951-012-0278-9.

JIA, Ya-Hui; CHEN, Wei-Neng; YUAN, Huaqiang; GU, Tianlong; ZHANG, Huaxiang; GAO, Ying; ZHANG, Jun. An Intelligent Cloud Workflow Scheduling System With Time Estimation and Adaptive Ant Colony Optimization. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, *[S. l.]*, v. 51, n. 1, p. 634–649, 2021. DOI: 10.1109/TSMC.2018.2881018. Disponível em: https://ieeexplore.ieee.org/document/8561182/.

JIA, Zhaohong; YAN, Jianhai; LEUNG, Joseph Y. T.; LI, Kai; CHEN, Huaping. Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities. **Applied Soft Computing Journal**, *[S. l.]*, v. 75, p. 548–561, 2019. DOI: 10.1016/j.asoc.2018.11.027. Disponível em: https://doi.org/10.1016/j.asoc.2018.11.027.

JOSHI, D.; MITTAL, M. L.; SHARMA, M. K.; KUMAR, M. An effective teaching-learning-based optimization algorithm for the multi-skill resource-constrained project scheduling problem. **Journal of Modelling in Management**, *[S. l.]*, v. 14, n. 4, p. 1064–1087, 2019. DOI: 10.1108/JM2-07-2018-0108.

JOSHI, Dheeraj; MITTAL, M. L.; KUMAR, Manish. Modeling a bi-objective multi-skill resource-constrained project scheduling problem to minimize project makespan and skill divergence span. **Proceedings of the International Conference on Industrial Engineering and Operations Management**, *[S. l.]*, v. 0, n. March, p. 1017–1026, 2020.

KANNIMUTHU, Marimuthu; EKAMBARAM, Palaneeswaran; RAPHAEL, Benny; KUPPUSWAMY, Ananthanarayanan. Resource Unconstrained and Constrained Project Scheduling Problems and Practices in a Multiproject Environment. **Advances in Civil Engineering**, *[S. l.]*, v. 2018, 2018. DOI: 10.1155/2018/9579273.

KIM, Ilhyung; SPRINGER, Mark; ZHANG, Zhe George; PARK, Young Sun. Organizational learning: Approximation of multiple-level learning and forgetting by an aggregated single-level model. **Computers and Industrial Engineering**, *[S. l.]*, v. 131, n. xxxx, p. 442–454, 2019. DOI: 10.1016/j.cie.2018.10.004. Disponível em: https://doi.org/10.1016/j.cie.2018.10.004.

KRÜGER, Doreen; SCHOLL, Armin. Managing and modelling general resource transfers in (multi-)project scheduling. **OR Spectrum**, *[S. l.]*, v. 32, n. 2, p. 369–394, 2010. DOI: 10.1007/s00291-008-0144-5.

KUHPFAHL, J.; BIERWIRTH, C. A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. **Computers and Operations Research**, *[S. l.]*, v. 66, p. 44–57, 2016. DOI: 10.1016/j.cor.2015.07.011. Disponível em: http://dx.doi.org/10.1016/j.cor.2015.07.011.

LAM, K. C.; LEE, Donald; HU, Tiesong. Understanding the effect of the learning-forgetting phenomenon to duration of projects construction. **International Journal of Project Management**, *[S. l.]*, v. 19, n. 7, p. 411–420, 2001. DOI: 10.1016/S0263-7863(00)00025-9.

LASZCZYK, Maciej; MYSZKOWSKI, Paweł B. Improved selection in evolutionary multi–objective optimization of Multi–Skill Resource–Constrained project scheduling problem. **Information Sciences**, *[S. l.]*, v. 481, p. 412–431, 2019. DOI: 10.1016/j.ins.2019.01.002. Disponível em: https://doi.org/10.1016/j.ins.2019.01.002.

LI, CHUNG-LUN -L; CHENG, T. C. E. an Economic Production Quantity Model With Learning and Forgetting Considerations. **Production and Operations Management**, *[S. l.]*, v. 3, n. 2, p. 118–132, 1994. DOI: 10.1111/j.1937-5956.1994.tb00114.x.

LI, Feng; LIAO, T. W.; ZHANG, Lin. Two-level multi-task scheduling in a cloud manufacturing environment. **Robotics and Computer-Integrated Manufacturing**, *[S. l.]*, v. 56, p. 127–139, 2019. DOI: 10.1016/j.rcim.2018.09.002. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0736584518300772.

LI, Feng; ZHANG, Lin; LIAO, T. W.; LIU, Yongkui. Multi-objective optimisation of multi-task scheduling in cloud manufacturing. **International Journal of Production Research**, *[S. l.]*, v. 57, n. 12, p. 3847–3863, 2019. DOI: 10.1080/00207543.2018.1538579. Disponível em: https://doi.org/00207543.2018.1538579.

LIAO, Ching-Jong; YOU, Chii-Tsuen. An Improved Formulation for the Job-Shop Scheduling Problem. **Journal of the Operational Research Society**, *[S. l.]*, v. 43, n. 11, p. 1047–1054, 1992. DOI: 10.1057/jors.1992.162. Disponível em: http://www.jstor.org/stable/3009290.

LIN, Canhong; CHOY, K. L.; HO, G. T. S.; CHUNG, S. H.; LAM, H. Y. Survey of Green Vehicle Routing Problem: Past and future trends. **Expert Systems with Applications**, *[S. l.]*, v. 41, n. 4 PART 1, p. 1118–1138, 2014. DOI: 10.1016/j.eswa.2013.07.107.

LIN, Jian; ZHU, Lei; GAO, Kaizhou. A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. **Expert Systems with Applications**, *[S. l.]*, v. 140, p. 112915, 2020. DOI: 10.1016/j.eswa.2019.112915. Disponível em: https://doi.org/10.1016/j.eswa.2019.112915.

LIN, Miao; XI, Jianqing; BAI, Weihua; WU, Jiayin. Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud. **IEEE Access**, *[S. l.]*, v. 7, p. 83088–83100, 2019. DOI: 10.1109/ACCESS.2019.2924414. Disponível em: https://ieeexplore.ieee.org/document/8744199/.

MANNE, Alan S. On the Job-Shop Scheduling Problem. **Operations Research**, *[S. l.]*, v. 8, n. 2, p. 219–223, 1960. DOI: 10.1287/opre.8.2.219.

MIGUEL, P. A. C.; MORABITO, R.; PUREZA, V. **Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações**. 2ª ed. [s.l.] : Elsevier, 2012.

MORALES, Sergio Gomez; RONCONI, Débora Pretti. Formulações matemáticas e estratégias de resolução para o problema job shop clássico. **Production**, *[S. l.]*, v. 26, n. 3, p. 614–625, 2015. DOI: 10.1590/0103-6513.058512. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-65132016000300614&lng=pt&tlng=pt.

MORALES, Sergio Wilson Gomez. **Formulações matemáticas e estratégias de resolução para o problema job shop clássico**. 2012. USP, *[S. l.]*, 2012. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3136/tde-13062013-164920/publico/Diss_SERGIOWILSONGOMEZMORALES.pdf.

MYSZKOWSKI, P. B.; SKOWROŃSKI, M. E.; OLECH, Ł. P.; OŚLIZŁO, K. Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. **Soft Computing**, *[S. l.]*, v. 19, n. 12, p. 3599–3619, 2015. a. DOI: 10.1007/s00500-014-1455-x.

MYSZKOWSKI, Pawel B. GRASP Applied to Multi – Skill Resource – Constrained Project. *[S. l.]*, p. 402–411, 2016. DOI: 10.1007/978-3-319-45243-2.

MYSZKOWSKI, Paweł B.; LASZCZYK, Maciej. Diversity based selection for many-objective

evolutionary optimisation problems with constraints. **Information Sciences**, *[S. l.]*, v. 546, p. 665–700, 2021. DOI: 10.1016/j.ins.2020.08.118.

MYSZKOWSKI, Paweł B.; OLECH, Łukasz P.; LASZCZYK, Maciej; SKOWROŃSKI, Marek E. Hybrid Differential Evolution and Greedy Algorithm (DEGR) for solving Multi-Skill Resource-Constrained Project Scheduling Problem. **Applied Soft Computing Journal**, *[S. l.]*, v. 62, p. 1–14, 2018. DOI: 10.1016/j.asoc.2017.10.014.

MYSZKOWSKI, Paweł B.; SKOWROŃSKI, Marek E.; OLECH, Łukasz P.; OŚLIZŁO, Krzysztof. Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. **Soft Computing**, *[S. l.]*, v. 19, n. 12, p. 3599–3619, 2015. b. DOI: 10.1007/s00500-014-1455-x. Disponível em: http://link.springer.com/10.1007/s00500-014-1455-x.

MYSZKOWSKI, Paweł B.; SKOWROŃSKI, Marek E.; SIKORA, Krzysztof. A new benchmark dataset for Multi-Skill resource-constrained project scheduling problem. **Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015**, *[S. l.]*, v. 5, p. 129–138, 2015. DOI: 10.15439/2015F273.

NEMBHARD, D. A.; UZUMERI, M. V. Experiential learning and forgetting for manual and cognitive tasks. **International Journal of Industrial Ergonomics**, *[S. l.]*, v. 25, n. 4, p. 315–326, 2000. DOI: 10.1016/S0169-8141(99)00021-9.

PÉREZ, E.; POSADA, M.; LORENZANA, A. Taking advantage of solving the resource constrained multi-project scheduling problems using multi-modal genetic algorithms. **Soft Computing**, *[S. l.]*, v. 20, n. 5, p. 1879–1896, 2016. DOI: 10.1007/s00500-015-1610-z.

PETTICREW, M.; ROBERTS, H. **Systematic Reviews in the Social Sciences**. Oxford, UK: Blackwell Publishing Ltd, 2006. DOI: 10.1002/9780470754887. Disponível em: http://doi.wiley.com/10.1002/9780470754887.

PINHA, Denis C.; AHLUWALIA, Rashpal S. Flexible resource management and its effect on project cost and duration. **Journal of Industrial Engineering International**, *[S. l.]*, v. 15, n. 1, p. 119–133, 2019. DOI: 10.1007/s40092-018-0277-3. Disponível em: https://doi.org/10.1007/s40092-018-0277-3.

PMI. **Gerenciamento de Projetos (Guia PMBOK®)**. 6ª edição ed. [s.l.] : Project Management Institute, 2018.

QUOC, Huu Dang; NGUYEN THE, Loc; DOAN, Cuong Nguyen; PHAN THANH, Toan. Solving resource constrained project scheduling problem by a discrete version of cuckoo search algorithm. **Proceedings - 2019 6th NAFOSTED Conference on Information and Computer Science, NICS 2019**, *[S. l.]*, p. 73–76, 2019. DOI: 10.1109/NICS48868.2019.9023867.

QUOC, Huu Dang; THE, Loc Nguyen; DOAN, Cuong Nguyen; THANH, Toan Phan. New Cuckoo Search Algorithm for the Resource Constrained Project Scheduling Problem. **Proceedings - 2020 RIVF International Conference on Computing and Communication Technologies, RIVF 2020**, *[S. l.]*, p. 16–18, 2020. a. DOI: 10.1109/RIVF48685.2020.9140728.

QUOC, Huu Dang; THE, Loc Nguyen; DOAN, Cuong Nguyen; THANH, Toan Phan. New Effective Differential Evolution Algorithm for the Project Scheduling Problem. **2020 2nd International Conference on Computer Communication and the Internet, ICCCI 2020**, *[S. l.]*, p. 150–157, 2020. b. DOI: 10.1109/ICCCI49374.2020.9145982.

QUOC, Huu Dang; THE, Loc Nguyen; DOAN, Cuong Nguyen; THANH, Toan Phan; XIONG, Neal. Intelligent Differential Evolution Scheme for Network Resources in IoT. **Scientific Programming**, *[S. l.]*, v. 2020, p. 1–12, 2020. c. DOI: 10.1155/2020/8860384.

RAMIREZ PALENCIA, Alberto E.; MEJIA DELGADILLO, Gonzalo E. A computer application for a bus body assembly line using Genetic Algorithms. **INTERNATIONAL JOURNAL OF PRODUCTION ECONOMICS**, PO BOX 211, 1000 AE AMSTERDAM, NETHERLANDS, v. 140, n. 1, p. 431–438, 2012. DOI: 10.1016/j.ijpe.2012.06.026.

ROSSI, Andrea; DINI, Gino. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. **Robotics and Computer-Integrated Manufacturing**, *[S. l.]*, v. 23, n. 5, p. 503–516, 2007. DOI: 10.1016/j.rcim.2006.06.004.

RUBAIEE, Saeed; YILDIRIM, Mehmet Bayram. An energy-aware multiobjective ant colony algorithm to minimize total completion time and energy cost on a single-machine preemptive scheduling. **Computers and Industrial Engineering**, *[S. l.]*, v. 127, n. December 2018, p. 240–252, 2019. DOI: 10.1016/j.cie.2018.12.020. Disponível em: https://doi.org/10.1016/j.cie.2018.12.020.

SANCHEZ, Mariam Gomez; GIL, Alejandro Fernandez; CASTRO, Carlos. Integrating a SMT Solver based Local Search in Ant Colony Optimization for Solving RCMPSP. **2019 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2019**, *[S. l.]*, n. 1, 2019. DOI: 10.1109/LA-CCI47412.2019.9036765.

SANTOS, Andre Luis Marques Ferreira. **Simulação e otimização para o problema integrado de alocação de recursos humanos especialistas e sequenciamento de tarefas em uma indústria criativa**. 2017. UNINOVE, *[S. l.]*, 2017.

SAYIN, S.; KARABATI, S. Assigning cross-trained workers to departments: A two-stage optimization model to maximize utility and skill improvement. **European Journal of Operational Research**, *[S. l.]*, v. 176, n. 3, p. 1643–1658, 2007. DOI: 10.1016/j.ejor.2005.10.045.

SHAFER, Scott M.; NEMBHARD, David A.; UZUMERI, Mustafa V. The effects of worker learning, forgetting, and heterogeneity on assembly line productivity. **Management Science**, *[S. l.]*, v. 47, n. 12, p. 1639–1653, 2001. DOI: 10.1287/mnsc.47.12.1639.10236.

SHTUB, Avraham; LEVIN, Nissan; GLOBERSON, Shlomo. Learning and forgetting industrial skills: an experimental model. **The International journal of human factors in manufacturing**, *[S. l.]*, v. 3, n. 3, p. 293–305, 1993. DOI: 10.1002/hfm.4530030307.

SHU, Xin; SU, Qiang; WANG, Qian; WANG, Qiugen. Optimization of Resource-Constrained Multi-Project Scheduling Problem Based on the Genetic Algorithm. **2018 15th International Conference on Service Systems and Service Management, ICSSSM 2018**, *[S. l.]*, p. 1–6, 2018. DOI: 10.1109/ICSSSM.2018.8465086.

SHYU, S. J.; LIN, B. M. T.; YIN, P. Y. Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. **Computers and Industrial Engineering**, *[S. l.]*, v. 47, n. 2–3, p. 181–193, 2004. DOI: 10.1016/j.cie.2004.06.006.

SKOWROŃSKI, Marek E.; MYSZKOWSKI, Paweł B.; ADAMSKI, Marcin; KWIATEK, Pawel. Tabu search approach for Multi-Skill Resource-Constrained Project Scheduling Problem. **2013 Federated Conference on Computer Science and Information Systems, FedCSIS 2013**, *[S. l.]*, p. 153–158, 2013.

SNAUWAERT, Jakob; VANHOUCKE, Mario. A new algorithm for resource-constrained project scheduling with breadth and depth of skills. **European Journal of Operational Research**, *[S. l.]*, v. 292, n. 1, p. 43–59, 2021. DOI: 10.1016/j.ejor.2020.10.032. Disponível em: https://doi.org/10.1016/j.ejor.2020.10.032.

SONMEZ, Rifat; UYSAL, Furkan. Backward-Forward Hybrid Genetic Algorithm for Resource-Constrained Multiproject Scheduling Problem. **Journal of Computing in Civil Engineering**, *[S. l.]*, v. 29, n. 5, 2015. DOI: 10.1061/(ASCE)CP.1943-5487.0000382.

SPRECHER, Arno; KOLISCH, Rainer; DREXL, Andreas. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. **European Journal of Operational Research**, *[S. l.]*, v. 80, n. 1, p. 94–102, 1995. DOI: 10.1016/0377-2217(93)E0294-8.

STRATMAN, J. K.; ROTH, A. V.; GILLAND, W. G. The deployment of temporary production workers in assembly operations: A case study of the hidden costs of learning and forgetting. **Journal of Operations Management**, *[S. l.]*, v. 21, n. 6, p. 689–707, 2004. DOI: 10.1016/j.jom.2003.11.001.

TIAN, M.; LIU, R. J.; ZHANG, G. J. Solving the resource-constrained multi-project scheduling problem with an improved critical chain method. **Journal of the Operational Research Society**, *[S. l.]*, 2019. DOI: 10.1080/01605682.2019.1609883.

TURNER, John R. G. Learning and memory in mimicry. I. Simulations of laboratory experiments. **Philosophical Transactions of the Royal Society B: Biological Sciences**, *[S. l.]*, v. 351, n. 1344, p. 1157–1170, 1996. DOI: 10.1098/rstb.1996.0100.

UYSAL, Furkan; SONMEZ, Rifat; ISLEYEN, Selcuk Kursat. A graphical processing unit-based parallel hybrid genetic algorithm for resource-constrained multi-project scheduling problem. **Concurrency and Computation: Practice and Experience**, *[S. l.]*, v. 33, n. 16, p. 1–11, 2021. DOI: 10.1002/cpe.6266.

VÁZQUEZ, E. P.; CALVO, M. P.; ORDÓÑEZ, P. M. Learning process on priority rules to solve the RCMPSP. **Journal of Intelligent Manufacturing**, *[S. l.]*, v. 26, n. 1, p. 123–138, 2013. DOI: 10.1007/s10845-013-0767-5.

VILLAFÁÑEZ, F. et al. A generic heuristic for multi-project scheduling problems with global and local resource constraints (RCMPSP). **SOFT COMPUTING**, ONE NEW YORK PLAZA, SUITE 4600, NEW YORK, NY, UNITED STATES, v. 23, n. 10, p. 3465–3479, 2019. a. DOI: 10.1007/s00500-017-3003-y.

VILLAFÁÑEZ, F.; POZA, D.; LÓPEZ-PAREDES, A.; PAJARES, J.; OLMO, R. A generic heuristic for multi-project scheduling problems with global and local resource constraints (RCMPSP). **Soft Computing**, *[S. l.]*, v. 23, n. 10, p. 3465–3479, 2019. b. DOI: 10.1007/s00500-017-3003-y.

VILLAFANEZ, Felix; POZA, David; LOPEZ-PAREDES, Adolfo; PAJARES, Javier; ACEBES, Fernando. Portfolio scheduling: an integrative approach of limited resources and project prioritization. **JOURNAL OF PROJECT MANAGEMENT**, 611, 141 DAVISVILLE AVE, TORONTO, ON M4S 1G7, CANADA, v. 5, n. 2, p. 103–116, 2020. DOI: 10.5267/j.jpm.2019.12.001.

VILLAFÁÑEZ, Félix; POZA, David; LÓPEZ-PAREDESA, Adolfo; PAJARESA, Javier; ACEBES, Fernando. Portfolio scheduling: an integrative approach of limited resources and project prioritization. **Journal of Project Management**, *[S. l.]*, v. 5, p. 103–116, 2020. DOI: 10.5267/j.jpm.2019.12.001.

WAGNER, Harvey M. An integer linear-programming model for machine scheduling. **Naval Research Logistics Quarterly**, *[S. l.]*, v. 6, n. 2, p. 131–140, 1959. DOI: 10.1002/nav.3800060205.

WALL, Matthew. GAlib : A C ++ Library of Genetic Algorithm Components. **Statistics**, *[S. l.]*, n. August, 1996.

WANG, L.; ZHENG, X. L. A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem. **Swarm and Evolutionary Computation**, *[S. l.]*, v. 38, p. 54–63, 2018. DOI: 10.1016/j.swevo.2017.06.001.

WANG, Yanting; HE, Zhengwen; KERKHOVE, Louis-Phillipe; VANHOUCKE, Mario. On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem. **COMPUTERS & INDUSTRIAL ENGINEERING**, THE BOULEVARD, LANGFORD LANE, KIDLINGTON, OXFORD OX5 1GB, ENGLAND, v. 114, p. 223–234, 2017. DOI: 10.1016/j.cie.2017.10.021.

WEI, Xianyong. Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. **Journal of Ambient Intelligence and Humanized Computing**, *[S. l.]*, 2020. DOI: 10.1007/s12652-020-02614-7. Disponível em: http://link.springer.com/10.1007/s12652-020-02614-7.

WILSON, J. M. Alternative Formulations of a Flow-shop Scheduling Problem. **Journal of the Operational Research Society**, *[S. l.]*, v. 40, n. 4, p. 395–399, 1989. DOI: 10.1057/jors.1989.58. Disponível em: https://www.tandfonline.com/doi/full/10.1057/jors.1989.58.

XING, Li Ning; CHEN, Ying Wu; WANG, Peng; ZHAO, Qing Song; XIONG, Jian. A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. **Applied Soft Computing Journal**, *[S. l.]*, v. 10, n. 3, p. 888–896, 2010. DOI: 10.1016/j.asoc.2009.10.006. Disponível em: http://dx.doi.org/10.1016/j.asoc.2009.10.006.

XING, Li Ning; CHEN, Ying Wu; YANG, Ke Wei. Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling. **Applied Soft Computing Journal**, *[S. l.]*, v. 9, n. 1, p. 362–376, 2009. DOI: 10.1016/j.asoc.2008.04.013.

YASSINE, A. A.; MOSTAFA, O.; BROWNING, T. R. Scheduling multiple, resource-constrained, iterative, product development projects with genetic algorithms. **Computers and Industrial Engineering**, *[S. l.]*, v. 107, p. 39–56, 2017. a. DOI: 10.1016/j.cie.2017.03.001.

YASSINE, Ali A. A. Ali A.; MOSTAFA, Omar; BROWNING, Tyson R. T. R. Tyson R. Scheduling multiple, resource-constrained, iterative, product development projects with genetic algorithms.

Computers and Industrial Engineering, *[S. l.]*, v. 107, p. 39–56, 2017. b. DOI: 10.1016/j.cie.2017.03.001. Disponível em: http://dx.doi.org/10.1016/j.cie.2017.03.001.

YAZDANI, Maziar; ALETI, Aldeida; KHALILI, Seyed Mohammad; JOLAI, Fariborz. Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem. **Computers and Industrial Engineering**, *[S. l.]*, v. 107, p. 12–24, 2017. DOI: 10.1016/j.cie.2017.02.019. Disponível em: http://dx.doi.org/10.1016/j.cie.2017.02.019.

ZAPATA, J. C.; HODGE, B. M.; REKLAITIS, G. V. The multimode resource constrained multiproject scheduling problem: Alternative formulations. **AIChE Journal**, *[S. l.]*, v. 54, n. 8, p. 2101–2119, 2008. DOI: 10.1002/aic.11522.

ZHENG, Huan yu; WANG, Ling; ZHENG, Xiao long. Teaching–learning-based optimization algorithm for multi-skill resource constrained project scheduling problem. **Soft Computing**, *[S. l.]*, v. 21, n. 6, p. 1537–1548, 2017. DOI: 10.1007/s00500-015-1866-3.

ZHENG, Xiaolong; WANG, Ling; ZHENG, Huanyu. A knowledge-based fruit fly optimization algorithm for multi-skill resource-constrained project scheduling problem. **Chinese Control Conference, CCC**, *[S. l.]*, v. 2015- Septe, p. 2615–2620, 2015. DOI: 10.1109/ChiCC.2015.7260039.

ZHENG, Xu; ZHOU, Shengchao; XU, Rui; CHEN, Huaping. Energy-efficient scheduling for multi-objective two-stage flow shop using a hybrid ant colony optimisation algorithm. **International Journal of Production Research**, *[S. l.]*, v. 58, n. 13, p. 4103–4120, 2020. DOI: 10.1080/00207543.2019.1642529. Disponível em: https://doi.org/00207543.2019.1642529.

ZHENG, Z.; SHUMIN, L.; ZE, G.; YUENI, Z. Resource-constraint Multi-project Scheduling with Priorities and Uncertain Activity Durations. **International Journal of Computational Intelligence Systems**, *[S. l.]*, v. 6, n. 3, p. 530–547, 2013. a. DOI: 10.1080/18756891.2013.789152.

ZHENG, Zheng; SHUMIN, Lin; ZE, Guo; YUENI, Zhu. Resource-constraint Multi-project Scheduling with Priorities and Uncertain Activity Durations. **International Journal of Computational Intelligence Systems**, *[S. l.]*, v. 6, n. 3, p. 530–547, 2013. b. DOI: 10.1080/18756891.2013.789152.

ZHU, Lei; LIN, Jian; LI, Yang Yuan; WANG, Zhou Jing. A decomposition-based multi-objective genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. **Knowledge-Based Systems**, *[S. l.]*, v. 225, p. 107099, 2021. DOI: 10.1016/j.knosys.2021.107099. Disponível em: https://doi.org/10.1016/j.knosys.2021.107099.

ZHU, Lei; LIN, Jian; WANG, Zhou Jing. A discrete oppositional multi-verse optimization algorithm for multi-skill resource constrained project scheduling problem. **Applied Soft Computing Journal**, *[S. l.]*, v. 85, n. xxxx, p. 105805, 2019. DOI: 10.1016/j.asoc.2019.105805. Disponível em: https://doi.org/10.1016/j.asoc.2019.105805.

ZHUANG, Miao; YASSINE, Ali A. Task scheduling of parallel development projects using genetic algorithms. **Proceedings of the ASME Design Engineering Technical Conference**, *[S. l.]*, v. 1, n. 217, p. 215–223, 2004. DOI: 10.1115/detc2004-57159.

ZUO, Liyun; SHU, Lei; DONG, Shoubin; ZHU, Chunsheng; HARA, Takahiro. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. **IEEE Access**, *[S. l.]*, v. 3, p. 2687–2699, 2015. DOI: 10.1109/ACCESS.2015.2508940.

APÊNDICE I. Formulação matemática dos problemas e modelo proposto

RCMPSP

Formulação matemática do RCMPSP que será utilizada nesta tese nas tabelas: 5, 6, 7 e 8, conforme Morales (2012).

Tabela 5. Índices da formulação matemática do RCMPSP

$\boldsymbol{n}$	Número total de tarefas
$\boldsymbol{m}$	Número de recursos
$\boldsymbol{i}$	Tarefa (*job*) i
$\boldsymbol{k}$	Recurso k
$\boldsymbol{l}$	Operação l
$\boldsymbol{j}$	Posição j

Fonte: Morales (2012)

Tabela 6. Parâmetros da formulação matemática do RCMPSP

$\boldsymbol{p_{ik}}$	Tempo de processamento da tarefa $\boldsymbol{i}$ pelo recurso $\boldsymbol{k}$
$\boldsymbol{r_{ilk}}$	Se a operação l da tarefa i requer o recurso k; observe que $\sum_{k=1}^{m} r_{ilk} = 1\ \forall i \forall l$
$\boldsymbol{M}$	Número suficientemente grande

Fonte: Morales (2012)

Tabela 7. Variáveis da formulação matemática do RCMPSP

$\boldsymbol{h_{jk}}$	Instante de início da tarefa na posição $\boldsymbol{j}$ na sequência do recurso $\boldsymbol{k}$
$\boldsymbol{S_{ik}}$	Instante de início da tarefa i pelo recurso k
$\boldsymbol{I_{1k}}$	Tempo ocioso do recurso k entre o instante de início do projeto e o instante de início da primeira tarefa na primeira posição do recurso k.
$\boldsymbol{I_{jk}}$	Tempo ocioso do recurso k entre o instante de término da tarefa na posição $(j-1)$ da sequência e o instante de início da tarefa na posição $\boldsymbol{j}$ para $j = 2,3,\ldots m$
$\boldsymbol{C_{max}}$	*Makespan*
$\boldsymbol{X_{ijk}} \begin{cases} \mathbf{1} \\ \mathbf{0} \end{cases}$	1, se a tarefa i é sequenciada na posição j do recurso k 0, caso contrário
$\boldsymbol{Z_{iuk}} \begin{cases} \mathbf{1} \\ \mathbf{0} \end{cases}$	1, se a tarefa i precede a tarefa u do recurso k 0, caso contrário

Fonte: Morales (2012)

Que são traduzidas nas seguintes restrições da Tabela 8:

Tabela 8. Formulação matemática para o RCMPSP

$\boldsymbol{minimizar\ C_{max}}$		**(1)**
$\boldsymbol{s.a.}$		
$\sum_{j=1}^{n} \boldsymbol{X_{ijk}} = \mathbf{1}$	$i = 1,2,\ldots,n$ $k = 1,2,\ldots,m$	(2)

$\sum_{i=1}^{n} X_{ijk} = 1$	$j = 1,2,\dots,n$ $k = 1,2,\dots,m$	(3)
$h_{jk} + \sum_{i=1}^{n} p_{ik} X_{ijk} \leq h_{j+1,k}$	$j = 1,2,\dots,n-1$ $k = 1,2,\dots,m$	(4)
$\sum_{k=1}^{m} r_{ilk}(h_{jk} + p_{ik}) \leq \sum_{k=1}^{m} r_{i,l+1,k} h_{wk} +$ $+M(1 - \sum_{k=1}^{m} r_{ilk} X_{ijk}) + M(1 - \sum_{k=1}^{m} r_{i,l+1,k} X_{iwk})$	$i,j,w = 1,2,\dots,n;$ $l = 1,2,\dots,m-1$	(5)
$h_{nk} + \sum_{i=1}^{n} p_{ik} X_{ink} \leq C_{max}$	$k = 1,2,\dots,m$	(6)
$h_{jk} \geq 0$	$j = 1,2,\dots,n$ $k = 1,2,\dots,m$	(7)
$X_{ijk} = 0 \; ou \; 1$	$i = 1,2,\dots,n$ $j = 1,2,\dots,n$ $k = 1,2,\dots,m$	(8)

Fonte: Morales (2012)

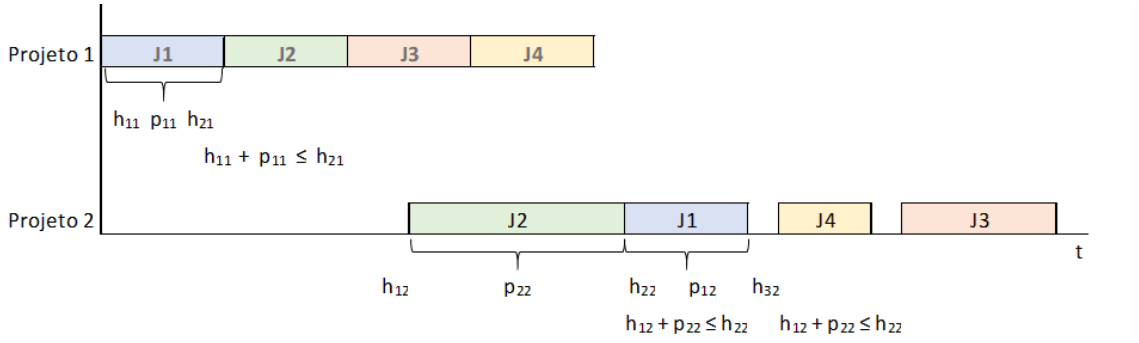
A função (1) representa o objetivo proposto pelo modelo que é de encontrar o menor valor para o *makespan*, como será explanado nos capítulos referente a construção do modelo e resultados obtidos.

A restrição de número (2) estabelece que cada tarefa só pode ser executada uma única vez por cada recurso.

Na restrição (3) é assegura a condição de que cada posição só pode conter uma tarefa por projeto.

A restrição de número (4), indica que o instante inicial da posição j + 1 um em um determinado projeto seja maior ou igual ao instante do início da posição anterior j mais o tempo de processamento da tarefa i estabelecida na posição j, como pode ser observado na Figura 29.

Figura 29. Exemplo do conjunto de restrições (4)



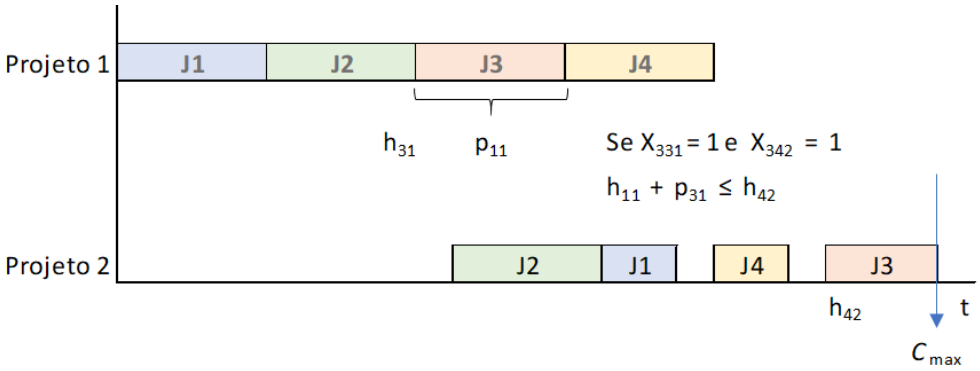
Fonte: Morales (2012)

Ao representar o RCMPSP, na Figura 29 através do funcionamento de um sistema com dois projetos é possível observar que no instante do início da posição 2 no Projeto 1 (h_{21}) é maior ou igual que o instante de início da posição 1 do projeto (h_{11}) mais o tempo de processamento da tarefa estabelecida nessa posição, no caso, a tarefa 1 (p_{11}), e assim sucessivamente para cada posição em cada projeto.

A restrição de número (5) determina que o instante inicial de uma posição em um projeto, seja igual à soma dos tempos ociosos, desde o início da produção até a posição a ser avaliada, mais o total dos tempos de processamento de todas as tarefas alocadas nas posições anteriores. Conforme pode ser exemplificado na Figura 29 (MORALES, 2012).

A restrição (6) estabelece que não pode haver duas tarefas sendo realizadas ao mesmo tempo em um mesmo projeto, ou seja, o início de processamento da operação $I + 1$ da tarefa i na posição w no respectivo projeto só pode ser iniciado quando finalizar a operação I na posição j. Observa-se que, de acordo com esta restrição, quando $X_{ijk_1} = 1\ e\ X_{iwk_2} = 1$ se terá $h_{jk_1} + p_{ik_1} \leq h_{wk_2}$, de forma a garantir que a posição w só inicie após o término de processamento da posição j, como pode ser visto no exemplo da Figura 30.

Figura 30 Exemplo do conjunto de restrições (6)



Fonte: Morales (2012)

Na Figura 30, quando $X_{331} = 1\ e\ X_{342} = 1$ o momento inicial na posição $j4$ no projeto 2 (h_{42}) que corresponde a operação $I + 1$ da tarefa $j3$ é igual ou maior que o instante de início de $j3$ no projeto 1 (h_{31}) correspondente a operação I da tarefa $j3$, mais o seu tempo de processamento (p_{31}), dessa forma há garantia que não ocorra sobreposição de operações na respectiva tarefa (MORALES, 2012).

Por fim, a restrição de número (7) determina o instante de término de todas as tarefas em todas as posições de cada projeto e considerando o início da primeira tarefa até a finalização do último projeto tem-se o valor do *makespan* do sistema. Já as restrições (8) e (9) estabelecem as variáveis como sendo não negativas e binárias respectivamente (MORALES, 2012).

MSRCPSP

Formulação matemática do MSRCPSP que será utilizada nesta tese nas tabelas: 9, 10, 11 e 12, conforme Myszkowski *et al.* (2015).

Tabela 9. Índices da formulação matemática do MSRCPSP

$\boldsymbol{n}$	Número total de tarefas
$\boldsymbol{m}$	Número de recursos
d	Duração da tarefa
$\boldsymbol{r}$	Taxa de salário
$\boldsymbol{K}$	Recurso K
$\boldsymbol{\tau}$	Duração do projeto
$\boldsymbol{j}$	Custo de performance
$\boldsymbol{J}$	Tarefas

Fonte: Myszkowski *et al.* (2015)

Tabela 10. Parâmetros da formulação matemática do MSRCPSP

$\boldsymbol{S_j}$	Tempo inicial
$\boldsymbol{F_j}$	Tempo Final
$\boldsymbol{d_j}$	Duração não preemptiva
$\boldsymbol{P_j}$	Predecessor da tarefa *j*
$\boldsymbol{s_k}$	Taxa do salário por hora
$\boldsymbol{l_q}$	Nível da habilidade requerida
$\boldsymbol{h_q}$	Tipo de habilidade requerida

Fonte: Myszkowski *et al.* (2015)

Tabela 11. Variáveis da formulação matemática do MSRCPSP

$\boldsymbol{Q^k}$	Recursos requeridos, que podem variar de 1 até r, enquanto os recursos requeridos disponíveis seja um subconjunto $\boldsymbol{Q^k} \in \boldsymbol{Q}$ das habilidades definidas no projeto
$\boldsymbol{q_j}$	Habilidade requerida do recurso j necessita realizar a tarefa
$\boldsymbol{J^k}$	Subconjunto de tarefas que podem ser realizadas pelo recurso k
$\boldsymbol{c_j^k}$	Custo de performance da tarefa *j* pelo recurso *k*, sendo que $c_j^k = d_j * s_k$. Como somente um recurso pode executar uma tarefa, c_j^k foi simplificado para $\boldsymbol{c_j}$
$\boldsymbol{U_{j,k}^t}$	Variável introduzida para definir quando o recurso *k* é atribuído para a tarefa *j* no tempo *t*, sendo que $U_{j,k}^t \in \{\mathbf{0};\mathbf{1}\}$. Quando o valor atribuído é 1, representa o recurso alocado, e de forma oposta, quando o valor atribuído é 0, o recurso não está alocado para a tarefa *j* no tempo *t*.
PS	Tempo de projeto factível é definido entre todos os projetos possíveis (Factíveis e Não factíveis – Violando requisitos de precedência, recursos e habilidades), de tal forma que $PS \in PS_{all}$

Fonte: Myszkowski *et al.* (2015)

Que são traduzidas nas seguintes restrições da Tabela 12:

Tabela 12. Formulação matemática para o MSRCPSP

$\boldsymbol{minimizar}\ \boldsymbol{f}(\boldsymbol{PS}) = \mathbf{min}[\boldsymbol{f}_{\tau}(\boldsymbol{PS}), \boldsymbol{f}_{C}(\boldsymbol{PS})]$	(1)
$\boldsymbol{sujeito\ a}$	
$\forall_{k \in K}\ \boldsymbol{S_k} \geq \mathbf{0}, \forall_{k \in K} \boldsymbol{Q^k} \neq \mathbf{0}$	(2)
$\forall_{j \in J}\ \boldsymbol{F_j} \geq \mathbf{0}, \forall_{j \in J} \boldsymbol{d_j} \geq \mathbf{0}$	(3)
$\forall_{j \in J, j \neq 1, i \in P_j,}\ \boldsymbol{F_i} \leq \boldsymbol{F_j} - \boldsymbol{d_j}$	(4)
$\forall_{i \in J^k}\ \exists_{q \in Q^k}\ \boldsymbol{h_q} = \boldsymbol{h_{q_i}} \wedge \boldsymbol{l_q} \geq \boldsymbol{l_{q_i}}$	(5)
$\forall_{k \in k} \forall_{t \in \tau} \sum_{i=1}^{n} \boldsymbol{U_{j,k}^t} \leq \mathbf{1}$	(6)
$\forall_{j \in J}\ \exists_{!t \in \tau, !k \in K}\ \boldsymbol{U_{j,k}^t} = \mathbf{1}$	(7)

Fonte: Myszkowski *et al.* (2015)

A função (1) representa o objetivo proposto pelo modelo que é de otimizar o valor para o tempo e custo. Depende da avaliação da função configurada, que nesta tese será realizada pela função descrita na seção correspondente. A função $\boldsymbol{f}_{\tau}(\boldsymbol{PS})$ otimiza o tempo do projeto, enquanto a função $\boldsymbol{f}_{C}(\boldsymbol{PS})$ otimiza o custo.

A restrição de número (2) visa preservar os valores positivos para os salários dos recursos, bem como a habilidade para realizar ao menos uma tarefa pelos recursos disponíveis.

Na restrição (3) é assegura a condição de que cada tarefa terá sempre um valor positivo de finalização e duração.

A restrição de número (4) apresenta as regras de restrição de precedência.

A restrição (5) apresenta a primeira alteração da formulação do problema RCPSP, introduzindo as restrições de habilidade, e assim, transformando o problema original no MSRCPSP.

A restrição (6) apresenta a segunda alteração em relação a formulação original, de forma que descreve que cada recurso só pode ser designado para não mais que uma tarefa por vez.

Por fim, a restrição (7) descreve que cada tarefa só pode ser realizada por um recurso no PS.

LFCM

Formulação matemática do LFCM que será utilizada nesta tese nas tabelas:13, 14, 15 e 16, conforme Chen *et al.* (2017).

Tabela 13. Variáveis da formulação matemática do LFCM

$\boldsymbol{a_{ij}}$	$-\ln(\boldsymbol{l_{ij}})/\ln 2$
$\boldsymbol{b_{ij}}$	$-\ln(1-f_{ij})/\ln 2$

Fonte: Chen *et al.* (2017)

Tabela 14. Índices da formulação matemática do LFCM

$\boldsymbol{T}$	Número total de períodos
$\boldsymbol{j}$	Habilidade dominada pelo recurso
i	Recurso
$\boldsymbol{t}$	Período

Fonte: Chen *et al.* (2017)

Tabela 15. Parâmetros da formulação matemática do LFCM

$\mathbf{E_{ijt}}$	Eficiência da habilidade
$\boldsymbol{t_{ij}}$	Número total de períodos usados pelo recurso *i* antes do período *t*
$\boldsymbol{a_{ij}}$	Fator de aprendizado do recurso *i* na habilidade *j*
$\boldsymbol{l_{ij}}$	Percentual de aprendizado. Maior valor de a_{ij} e o maior valor do efeito de aprendizado
$\boldsymbol{b_{ij}}$	Fator de esquecimento do recurso *i* na habilidade *j*
$\boldsymbol{f_{ij}}$	Percentual de esquecimento que expressa a taxa de depreciação do recurso *i* na habilidade *j*. O menor valor de f_{ij} é o maior valor de b_{ij} sendo o menor efeito do esquecimento

Fonte: Chen *et al.* (2017)

Tabela 16. Formulação matemática para o LFCM

$$\mathbf{E_{ijt}} = \mathbf{E_{ij1}}\boldsymbol{t_{ij}^{a_{ij}}}(\boldsymbol{T}-\boldsymbol{t_{ij}})^{\boldsymbol{b_{ij}}} \quad (1)$$

$\boldsymbol{sujeito\ a}$

$$\mathbf{0} \leq \mathbf{E_{ijt}} \leq \mathbf{1} \quad (2)$$

$$\mathbf{1} \leq \boldsymbol{t_{ij}} < \boldsymbol{T} \quad (3)$$

$$\mathbf{0} < \boldsymbol{l_{ij}} \leq \mathbf{1} \quad (4)$$

$$\mathbf{0} \leq \boldsymbol{f_{ij}} < \mathbf{1} \quad (5)$$

Fonte: Chen *et al.* (2017)

A equação (1) representa a função objetivo, que representa a curva de aprendizado e esquecimento integradas do recurso *i* com a habilidade *j* no tempo *t* sujeito as restrições, os parâmetros na Tabela 15 e por fim, as variáveis na Tabela 13.

A restrição (2) representa a eficiência das habilidades do recurso *i* com a habilidade *j* com limite inferior igual a zero e superior igual a um.

A restrição (3) representa o número total de períodos do recurso *i* com a habilidade *j* com limite inferior igual a zero e superior menor do que o número total de períodos avaliados.

A restrição (4) se refere ao percentual de aprendizado do recurso *i* com a habilidade *j* utilizado no cálculo do fator de aprendizado com limite inferior maior do que zero e limite superior igual a um.

A restrição (5) se refere ao percentual de esquecimento do recurso *i* com a habilidade *j* utilizado no cálculo do fator de esquecimento com limite inferior igual a zero e limite superior menor do que um.

ACO

Formulação matemática do ACO que será utilizada nesta tese nas tabelas: 17, 18 e 19, conforme Li *et al.* (2019).

Tabela 17. Índices da formulação matemática do ACO modificado

$\boldsymbol{T}$	Projetos
$\boldsymbol{w}$	Tempo de carga dos projetos
$\boldsymbol{n}$	Descrição do número do projeto
$\boldsymbol{t}$	Período
$\boldsymbol{K}$	Recurso
$\boldsymbol{i}$	Projeto
$\boldsymbol{J}$	Tarefa
$\boldsymbol{\alpha}$	Peso do feromônio
$\boldsymbol{\beta}$	Peso da informação da heurística
$\boldsymbol{\varphi}$	Conjunto de tarefas ainda não sequenciadas
$\boldsymbol{\phi}$	Taxa global de atualização
$\boldsymbol{\rho}$	Taxa de evaporação do feromônio
$\boldsymbol{FT_{maximo}}$	Menor valor de *makespan* de todas as tarefas por todas as formigas de todas as iterações
$\boldsymbol{Itr}$	Número de iterações
$\boldsymbol{M}$	Número de formigas
$\boldsymbol{Q}$	Constante

Fonte: Li *et al.* (2019)

Tabela 18. Parâmetros da formulação matemática do ACO modificado

$\boldsymbol{T_i}$	Número do projeto
$\boldsymbol{ES_{i_{j(k)}}}$	Elegibilidade do recurso K para atender a tarefa j do projeto i
$\boldsymbol{RC_{jl}}$	Denota a relação de dependência da tarefa j e da l
$\boldsymbol{P_{i_{j(k)}}}$	Tempo de processamento por unidade de carga
$\boldsymbol{ST}$	Tempo de inicialização da tarefa
$\boldsymbol{TT}$	Tempo de transferência entre recursos de empresas diferentes
$\boldsymbol{PST_{i_{j(k)}}}$	Tempo previsto de inicialização da tarefa
	É tempo de inicialização entre a execução do serviço t_{mn} e o serviço t_{ij} para recurso k
$\boldsymbol{FT_{i_{l(g)}}}$	Tempo final de processamento pelo recurso g para a tarefa t_{ij}
$\boldsymbol{pre_{ij}}$	Conjunto de precedência de tarefas para a tarefa t_{ij}
$\boldsymbol{TT_{i_{l(g)}i_{j(k)}}}$	Tempo necessário para transferir projeto t_{ij} executado pelo recurso g para o projeto t_{ij} executado pelo recurso k. Se os recursos foram da mesma empresa, o tempo zera 0

$AST_{i_{j(k)}}$	Tempo inicial da tarefa j do projeto i para ser processado pelo recurso k
$FT_{i_{j(k)}}$	Tempo final da tarefa j do projeto i para ser processado pelo recurso k
$AS_{i_{j(k)}}$	Recurso atual k para processar a tarefa j do projeto i sujeito a $\epsilon \{ES \mid ES_{i_{j(k)}} = 1\}$
$PT_{i_{j(k)}}$	Tempo de processamento do recurso k para executar a tarefa j do projeto i
$\overline{ST_{ij}}$	Tempo médio de inicialização da tarefa i para a tarefa j
TM_m	Tempo de *makespan* da sequência construída pela formiga m

Fonte: Li *et al.* (2019)

Tabela 19. Formulação matemática para o ACO modificado

$$ES_{ij(k)} = \begin{cases} 1, \textbf{se o recurso k pode atender a tarefa i do projeto j} \\ 0, \quad \textbf{caso contrário} \end{cases} \tag{1}$$

$$RC_{jl} = \begin{cases} 1, \textbf{se o recurso k pode atender a tarefa i do projeto j} \\ 0, \quad \textbf{caso contrário} \end{cases} \tag{2}$$

$$PST_{ij(k)} = \begin{cases} ST_{m_{n(k)}\, i_{j(k)}}, \quad \textbf{se j não possui predecessor} \\ ST_{m_{n(k)}\, i_{j(k)}} + \max\limits_{l \in pre_{ij}} \left(FT_{ij(g)} + TT_{ij(k)il(g)}\right), \quad \textbf{caso contrário} \end{cases} \tag{3}$$

$$ES_{i_{j(g)}} = ES_{i_{j(k)}} = ES_{m_{n(k)}} = 1 \tag{4}$$

$$AST_{i_{j(k)}} = max\left(PST_{i_{j(k)}}, AT_{i_{k(k)}}\right) \tag{5}$$

$$FT_{i_{j(k)}} = AST_{i_{j(k)}} + PT_{ij(k)} \tag{6}$$

$$Minimizar \max_{j \in T_i} \left(\sum_{k=1}^{K} FT_{i_{j(k)}} \; x \; AS_{i_{j(k)}} \right) (i = T_1, T_2 .. T_n) \tag{7}$$

$$Minimizar \max_{i \in T} \max_{j \in T_i} \left(\sum_{k=1}^{K} FT_{i_{j(k)}} \; x \; AS_{i_{j(k)}} \right) \tag{8}$$

$$\sum_{k=1}^{K} AS_{i_{j(k)}} = 1, \forall\, i, j \tag{9}$$

$$Se\ Pr_{ijcdk}, FT_{i_{j(k)}} \leq AS_{c_{d(k)}} \tag{10}$$

$$\sum_{k=1}^{K} ES_{i_{j(k)}} \leq K, \forall\, i, j \tag{11}$$

$$P^{a}_{i_{j(k)}} = \frac{\left(\tau^{a}_{i_{j(k)}}\right)^{\alpha} \left(\eta^{a}_{i_{j(k)}}\right)^{\beta}}{\sum_{k=1}^{K} \left(\tau^{a}_{i_{j(k)}}\right)^{\alpha} \left(\eta^{a}_{i_{j(k)}}\right)^{\beta}} \tag{12}$$

$$\eta^{a}_{i_{j(k)}} = \begin{cases} \dfrac{1}{PT_{ij(k)}}, se\ ES_{ij(k)} = 1 \\ 0, \quad \textbf{caso contrário} \end{cases} \tag{13}$$

$$P_{i_j}^b = \frac{\left(\tau_{i_j}^b\right)^\alpha \left(\eta_{i_j}^b\right)^\beta}{\sum_{l \epsilon \varphi} \left(\tau_{i_j}^b\right)^\alpha \left(\eta_{i_j}^b\right)^\beta} \quad (14)$$

$$\tau_{novo} = (1-\rho)\tau_{antigo} + \phi \frac{1}{FT_{maximo}} \quad (15)$$

$$\tau_{ij}(itr+1) = (1-\rho)\tau_{ij}(itr) + \Delta\tau_{ij} \quad (16)$$

$$\Delta\tau_{ij} = \sum_{m=1}^{M} \frac{Q}{TM_m} \quad (17)$$

$$PT_{ij(k)} = p_{ij(k)} * w_i \quad (18)$$

$$pre_{ij} = \{l | l \epsilon RC_{jl} = 1\} \quad (19)$$

$$\eta_{i_j}^b = \frac{1}{ST_{ij}} \quad (20)$$

Fonte: Li *et al.* (2019)

A restrição (1) representa a elegibilidade do recurso *k* para atender a tarefa *i* do projeto *j*.

A restrição (2) denota a relação de restrição entre a tarefa *j* e a tarefa *l*.

A restrição (3) assume que o recurso *k* está sempre disponível e o tempo previsto para iniciar a tarefa *j* do projeto *i*.

A restrição (4) restringe os tempos de inicialização de $ST_{m_{n(k)}i_{j(k)}}$.

A equação (5) demonstra o cálculo para o tempo inicial para execução da tarefa *j* do projeto *i* pelo recurso *k*.

A equação (6) demonstra o cálculo para o tempo final para execução da tarefa *j* do projeto *i* pelo recurso *k*.

A função (7) demonstra a função objetivo do cálculo do escalonamento das tarefas.

A função (8) demonstra a função objetivo do cálculo do escalonamento dos projetos.

A restrição (9) significa que toda tarefa deve ser atendida por apenas 1 recurso disponível.

A restrição (10) denota que a tarefa *j* do projeto *i* é arranjada para o recurso *k* antes da tarefa *d* do projeto *c*.

A restrição (11) indica que nem todo recurso pode atender toda tarefa, ou seja, o recurso deve poder atender a tarefa par executar a mesma.

A equação (12) demonstra a probabilidade de atribuição do recurso *k* para a tarefa *j* do projeto *i*.

A equação (13) demonstra a razão do tempo de processamento do recurso, se o recurso for elegível para atender a tarefa do projeto.

A equação (14) demonstra a probabilidade de escalonamento da tarefa *j* do projeto *i*.

A equação (15) demonstra o cálculo da atualização do feromônio para as tarefas.

A equação (16) demonstra o cálculo da atualização do feromônio para os projetos.

A equação (17) demonstra a taxa de cálculo de atualização do feromônio.

A equação (18) demonstra o valor de execução da tarefa no projeto.

A equação (19) demonstra o conjunto de precedências entre tarefas.

A equação (20) demonstra a razão do tempo de processamento da tarefa no projeto.

Algoritmo Proposto

Formulação matemática acrescentado ao algoritmo de Li *et al.* (2019) pelo algoritmo proposto nas tabelas: 20, 21, 22 e 23.

Tabela 20. Índices da formulação matemática do ACO híbrido proposto

O_{max}	**Valor máximo do objetivo encontrado durante o processamento**
O_{min}	Valor mínimo do objetivo encontrado durante o processamento
$O(K)$	Valor atual do objetivo para cálculo
A	Total de objetivos para processamento
γ	Valor do peso do enésimo objetivo sendo calculado
TT	Número total de períodos
h	Habilidade dominada pelo recurso
ω	Valor da priorização de projeto

Fonte: Autor

Tabela 21. Parâmetros da formulação matemática do ACO híbrido proposto

E_{Kht}	**Eficiência da habilidade**
t_{Kh}	Número total de períodos usados pelo recurso *K* antes do período *t*
a_{Kh}	Fator de aprendizado do recurso *K* na habilidade h
l_{Kh}	Percentual de aprendizado. Maior valor de a_{Kh} e o maior valor do efeito de aprendizado
b_{Kh}	Fator de esquecimento do recurso K na habilidade h
f_{Kh}	Percentual de esquecimento que expressa a taxa de depreciação do recurso K na habilidade h. O menor valor de f_{Kh} é o maior valor de b_{Kh} sendo o menor efeito do esquecimento

Fonte: Autor

Tabela 22. Variáveis da formulação matemática do ACO híbrido proposto

a_{ij}	$-\ln(l_{ij})/\ln 2$
b_{ij}	$-\ln(1-f_{ij})/\ln 2$

Fonte: Autor

Tabela 23. Formulação matemática para o ACO híbrido proposto

$$PC_{jl} = \begin{cases} 1, \textbf{se p projeto j é imediatamente precedida pela projeto j} \\ 0, \quad \textbf{caso contrário} \end{cases} \quad \textbf{(1)}$$

$$V^{+}_{maximizar} = \left(\frac{O_{max} - O(K)}{O_{max} - O_{min}}\right) \quad (2)$$

$$V^{+}_{minimizar} = \left(\frac{O(K) - O}{O_{max} - O_{min}}\right) \quad (3)$$

$$V^{-}_{maximizar} = \left(\frac{O(K) - O_{min}}{O_{max} - O_{min}}\right) \quad (4)$$

$$V^{-}_{minimizar} = \left(\frac{O_{max} - O(K)}{O_{max} - O_{min}}\right) \quad (5)$$

$$D(K^{+}) = \sum_{a=1}^{A} \gamma * (V^{+}_{minimizar}) + \sum_{a=1}^{A} \gamma * (V^{+}_{maximizar}) \quad (6)$$

$$D(K^{-}) = \sum_{a=1}^{A} \gamma * (V^{-}_{minimizar}) + \sum_{a=1}^{A} \gamma * (V^{-}_{maximizar}) \quad (7)$$

$$D(K) = \frac{D(K^{-})}{D(K^{+}) + D(K^{-})} \quad (8)$$

$$Se\ Pk_{ick}, FT_{i_{(k)}} \leq AS_{c_{(k)}} \quad (9)$$

$$pre_j = \{l | l\ \epsilon\ RC_{jl} = 1\} \quad (10)$$

$$E_{Kht} = E_{Kh1} t_{Kh}^{a_{Kh}} (TT - t_{Kh})^{b_{Kh}} \quad (11)$$

$$sujeito\ a$$

$$0 \leq E_{Kht} \leq 1 \quad (12)$$

$$1 \leq t_{Kh} < T \quad (13)$$

$$0 < l_{Kh} \leq 1 \quad (14)$$

$$0 \leq f_{Kh} < 1 \quad (15)$$

$$Maximizar \max_{j \in T_i} \left(\sum_{k=1}^{K} FT_{i_{j(k)}}\ x\ AS_{i_{j(k)}} \right) (i = T_1, T_2 .. T_n) \quad (16)$$

$$Maximizar \max_{i \in T} \max_{j \in T_i} \left(\sum_{k=1}^{K} FT_{i_{j(k)}}\ x\ AS_{i_{j(k)}} \right) \quad (17)$$

$$P_i^b = \omega \left(\frac{(\tau_i^b)^{\alpha} (\eta_i^b)^{\beta}}{\sum_{l \epsilon \varphi} (\tau_i^b)^{\alpha} (\eta_i^b)^{\beta}} \right) \quad (18)$$

Fonte: Autor

A restrição (1) denota a relação de restrição de precedência entre o projeto *j* e o projeto *l*.

A equação (2) demonstra o cálculo positivo do valor Objetivo quando a necessidade é maximizar o valor objetivo.

A equação (3) demonstra o cálculo positivo do valor Objetivo quando a necessidade é minimizar o valor objetivo.

A equação (4) demonstra o cálculo negativo do valor Objetivo quando a necessidade é maximizar o valor objetivo.

A equação (5) demonstra o cálculo negativo do valor Objetivo quando a necessidade é minimizar o valor objetivo.

A equação (6) demonstra o somatório dos cálculos positivos multiplicados pelo fator de impacto do objetivo na decisão final.

A equação (7) demonstra o somatório dos cálculos negativos multiplicados pelo fator de impacto do objetivo na decisão final.

A equação (8) demonstra o cálculo de tomada de decisão da solução a ser apresentada, não necessariamente, sendo a com maior valor de um dos objetivos, como por exemplo a melhor solução de minimizar o tempo, porém sendo o cálculo do conjunto das soluções com o melhor peso balanceado pelos fatores dos objetivos.

A restrição (9) denota que o projeto *i* é arranjado para o recurso *k* antes do projeto *c* (Priorização).

A restrição (10) demonstra o conjunto de precedências entre projetos.

A função (11) representa a função objetivo, que representa a curva de aprendizado e esquecimento integradas do recurso *i* com a habilidade *j* no tempo *t* sujeito as restrições que seguem.

A restrição (12) representa a eficiência das habilidades do recurso *i* com a habilidade *j* com limite inferior igual a zero e superior igual a um.

A restrição (13) representa o número total de períodos do recurso *i* com a habilidade *j* com limite inferior igual a zero e superior menor do que o número total de períodos avaliados.

A restrição (14) se refere ao percentual de aprendizado do recurso *i* com a habilidade *j* utilizado no cálculo do fator de aprendizado com limite inferior maior do que zero e limite superior igual a um.

A restrição (15) se refere ao percentual de esquecimento do recurso *i* com a habilidade *j* utilizado no cálculo do fator de esquecimento com limite inferior igual a zero e limite superior menor do que um.

A função (16) demonstra a função objetivo do cálculo do sequenciamento das tarefas, quando a função objetivo é maximizar o objetivo.

A função (17) demonstra a função objetivo do cálculo do sequenciamento dos projetos, quando a função objetivo é maximizar o objetivo.

A equação (18) demonstra a probabilidade de sequenciamento do projeto *i*.

APÊNDICE II. Sub algoritmos

Algoritmo 1. Algoritmo principal

Enquanto nc < maxnc
Escolhe-se um projeto aleatório como o projeto corrente
Utiliza-se o Algoritmo 2 para ajuste da sequência de projetos
Para cada formiga de projeto
Para cada projeto dentre os projetos ainda não processados
Seleciona-se um projeto dentre os não visitados conforme probabilidade dada pela Equação da restrição (14)
Fim
Associa-se uma sequência de projetos para cada formiga
Para cada sequência de projetos para cada formiga
Usa algoritmo 3 de escalonamento de tarefas e associação de recursos
Fim
Fim
Atualiza a melhor e pior solução dentre todos os projetos conforme a função objetivo (8) ou (37) do objetivo da lista de restrições e atualiza caso seja apropriado
Atualiza o feromônio dos projetos conforme Equação da restrição (16)
Avalia a melhor solução global conforme algoritmo de tomada de decisão conforme Equações das restrições de (22) a (28)
Fim

Fonte: Autor

Algoritmo 2. Algoritmo de ajuste de sequência dos projetos

Para cada projeto
Calcula a probabilidade de execução do projeto conforme probabilidade dada pela Equação da restrição (38) e associa na lista de projetos de forma ordenada, do maior para o menor
Cria-se uma lista com os projetos sem dependência, mantendo a ordem definida anteriormente
Associa-se os projetos com dependência no final da nova lista mantendo a ordem definida anteriormente
Inicializa a variável j com valor 0
Enquanto j < Número de Projetos
Se o projeto sendo processado na posição j da nova lista não tiver dependência
Se o projeto sendo processado não estiver na lista de saída
Insere o projeto na lista de saída
Incrementa j
Fim
Caso contrário
Para cada projeto
Se o projeto que possui dependência e o projeto atual estiverem na lista de saída
Se a posição do projeto que possui dependência for menor que a posição do projeto atual
Incrementa j
Fim
Fim
Se todos os projetos que o projeto atual possui dependência se encontram na lista de saída
Incrementa j
Caso contrário
Se projeto que o atual possui dependência não está na lista de saída
Insere projeto que o atual possui dependência na lista de saída
Caso contrário
Se projeto atual não está na lista de saída
Se posição na lista de saída do que o atual possui dependência for maior que o atual
Remove o dependente da lista de saída
Insere o dependente na posição do projeto atual da lista de saída (o atual ficará na posição + 1 em relação ao dependente)
Fim
Fim
Fim

```
                Fim
              Fim
              Se projeto atual não está na lista de saída
                Insere na lista de saída
              Fim
            Fim
          Fim
        Fim
```

Fonte: Autor

Algoritmo 3. Algoritmo de controle do processamento de toda a camada das tarefas

```
Enquanto nfe < maxnfe
  Inicializa lista de priorização de recursos possíveis por tarefa
  Inicializa lista de priorização de escalonamento de tarefas
  Para cada formiga de projeto
    Ajusta a sequência das tarefas, conforme Algoritmo 4
    Decodifica as duas listas (recursos possíveis por tarefa e de escalonamento de tarefas) para construir o escalonamento das tarefas com associação dos recursos avaliando conforme a função objetivo (7) ou (36) do objetivo da lista de restrições
    Atualiza a lista de priorização de recursos possíveis por tarefa
    Atualiza a lista de priorização de escalonamento de tarefas
  Fim
  Atualiza os feromônios conforme Equação (12) e (15) da lista de restrições
  Atualiza as probabilidades conforme Equação (14) e (15) da lista de restrições
  Atualiza a melhor e a pior solução conforme a função objetivo (7) ou (36) do objetivo da lista de restrições, e atualiza caso seja apropriado
  Atualiza a lista de soluções únicas encontradas para o modelo
  Incrementa nfe em 1
Fim
```

Fonte: Autor

Algoritmo 4. Algoritmo de ajuste de sequência das tarefas

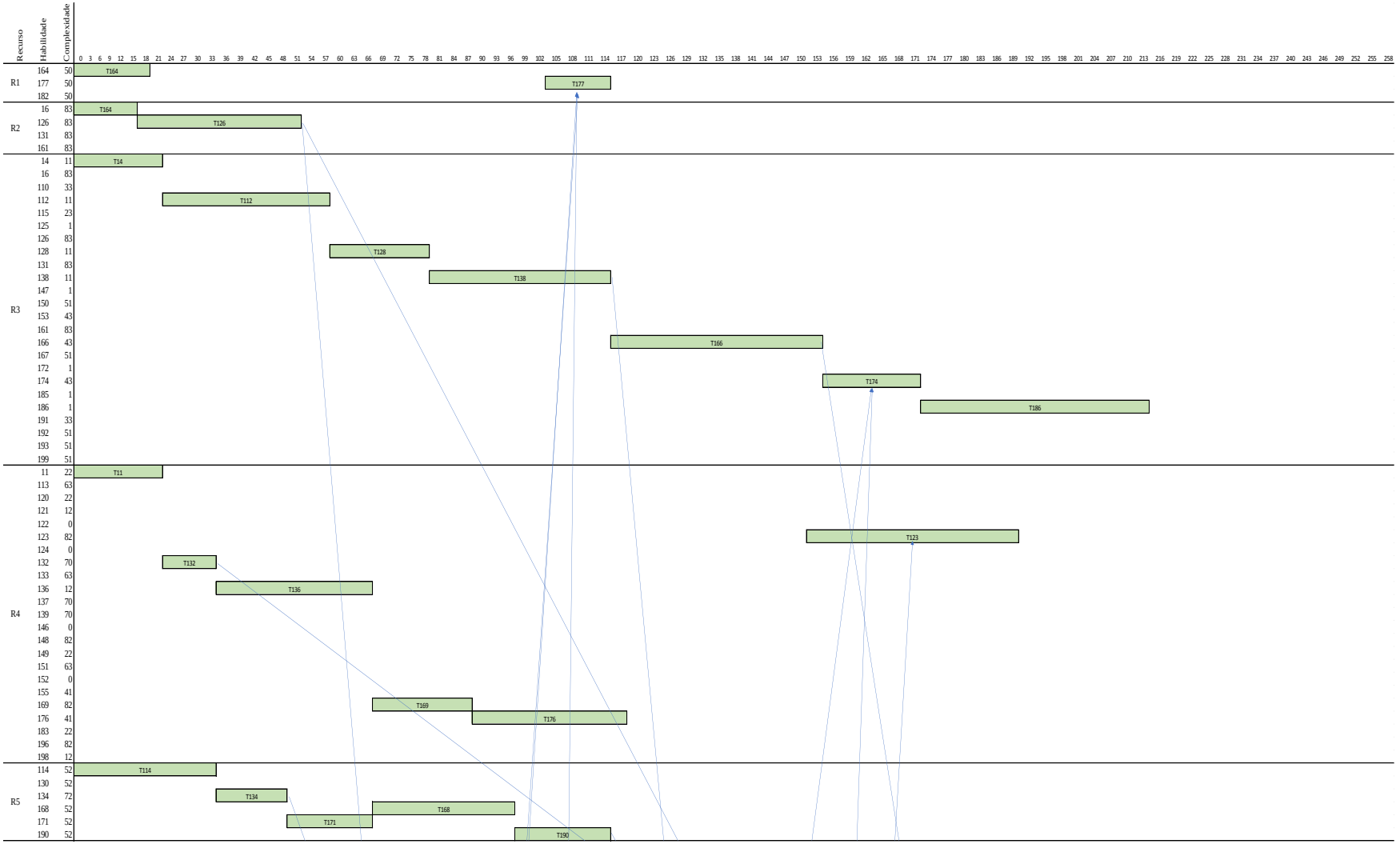
```
Para cada projeto
  Calcula a probabilidade de execução do projeto conforme probabilidade dada pela Equação da restrição (38) e associa na lista de projetos de forma ordenada, do maior para o menor
  Cria-se uma lista com os projetos sem dependência, com a ordem definida conforme item anteior
  Associa-se os projetos com dependência no final da nova lista, mantendo a ordem definida anteriormente
  Inicializa a variável j com valor 0
  Enquanto j < Número de Projetos
    Se o projeto sendo processado na posição j (projeto atual) da nova lista não tiver dependência
      Se o projeto sendo processado não estiver na lista de saída
        Insere o projeto na lista de saída
        Incrementa j
      Fim
    Caso contrário
      Para cada projeto que o projeto atual possua dependência
        Se o projeto que possui dependência e o projeto atual estiverem na lista de saída
          Se a posição do projeto que possui dependência for menor que a posição do projeto atual
            Incrementa j
          Fim
        Fim
        Se todos os projetos que o projeto atual possui dependência se encontram na lista de saída
          Incrementa j
        Caso contrário
          Se projeto que o projeto atual possui dependência não está na lista de saída
            Insere projeto que o projeto atual possui dependência na lista de saída
          Caso contrário
```

```
                    Se projeto atual não está na lista de saída
                      Se posição na lista de saída do projeto que o projeto atual possui dependência for
        maior que a atual
                        Remove o projeto dependente da lista de saída
                        Insere o projeto dependente na posição do projeto atual da lista de saída (o projeto
        atual ficará na posição + 1 em relação ao dependente)
                      Fim
                    Fim
                  Fim
                Fim
              Fim
              Se projeto atual não está na lista de saída
                Insere o projeto atual na lista de saída
              Fim
            Fim
          Fim
        Fim
```

Fonte: Autor

APÊNDICE III. Figuras de comparação dos problemas

Figura 31. Uma das soluções propostas pelo modelo para a base iMOPSE



a)

Figura 32. Uma das soluções propostas pelo modelo para a base iMOPSE

b)

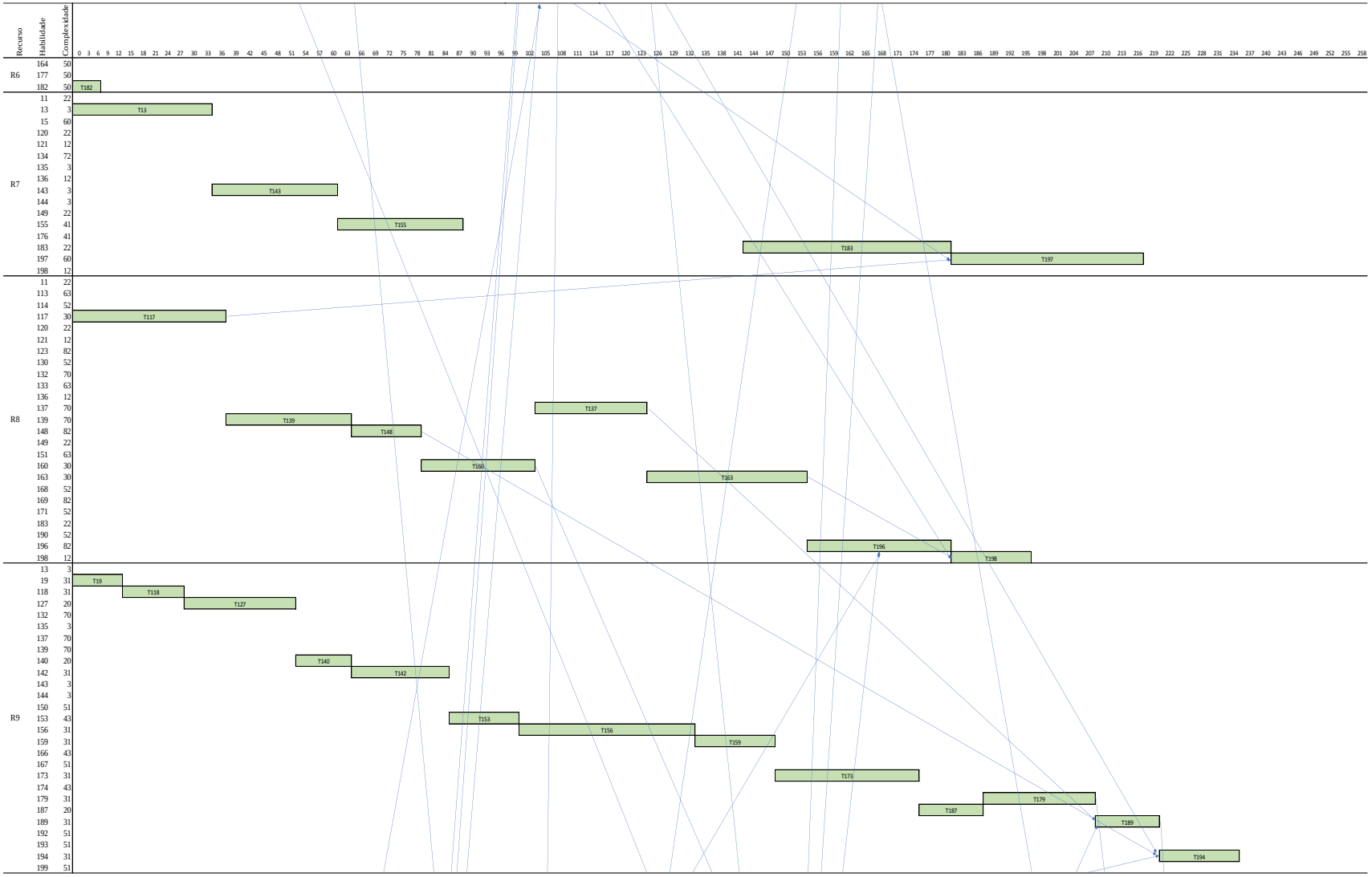


Figura 33. Uma das soluções propostas pelo modelo para a base iMOPSE

c)

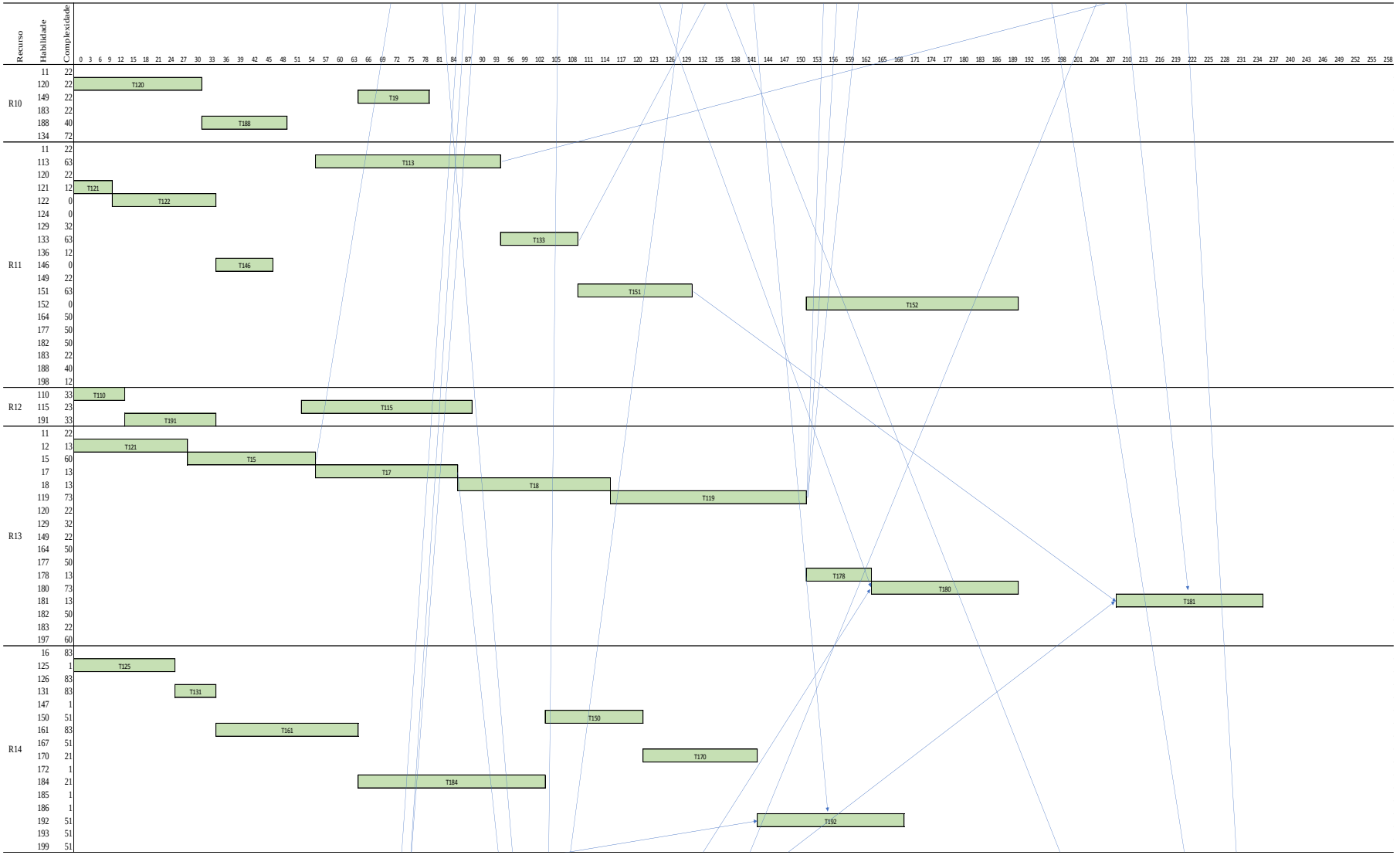
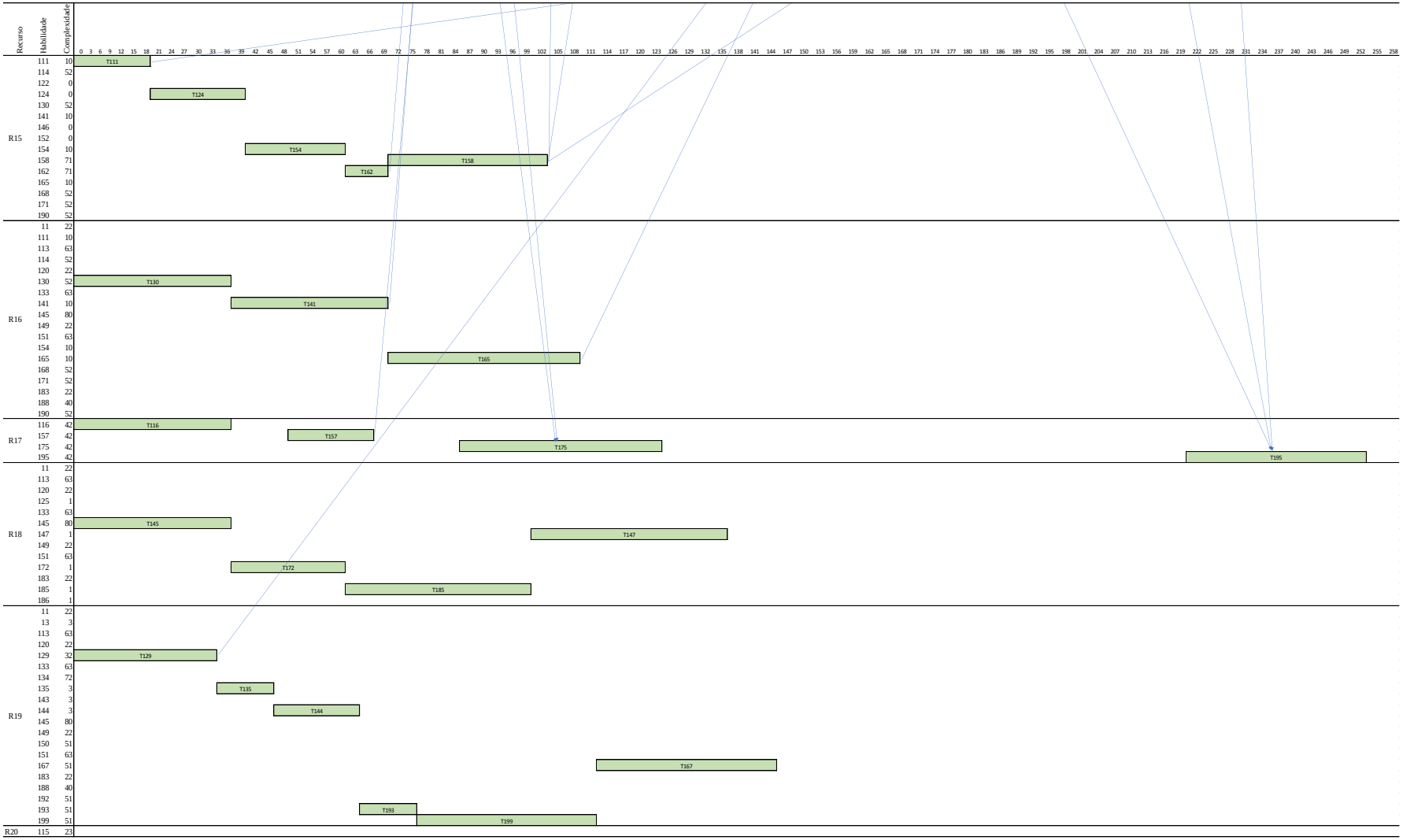


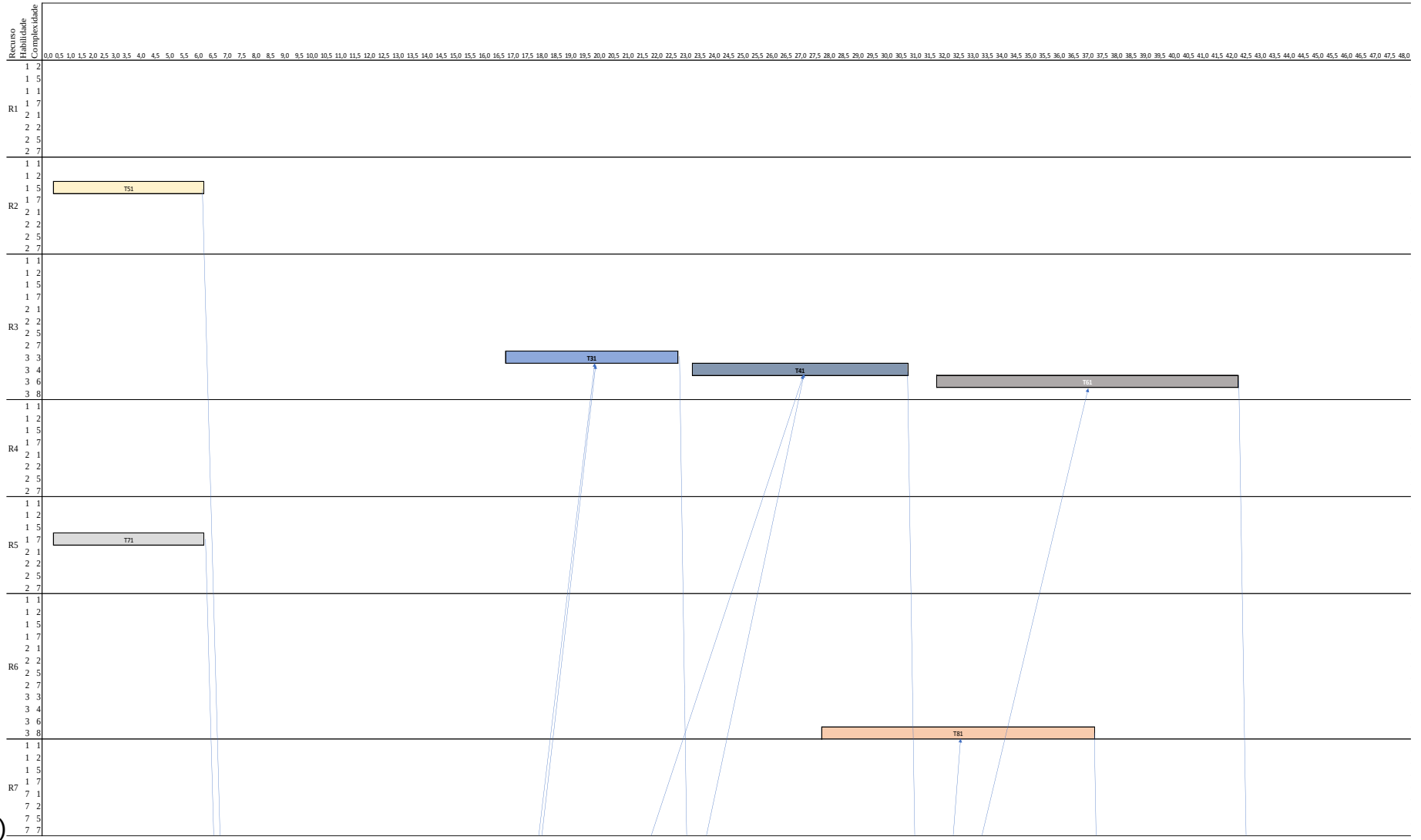
Figura 34. Uma das soluções propostas pelo modelo para a base iMOPSE



d)

Fonte: Autor

Figura 35. Uma das soluções propostas pelo modelo para a base LFCM



a)

Figura 36. Uma das soluções propostas pelo modelo para a base LFCM

b)

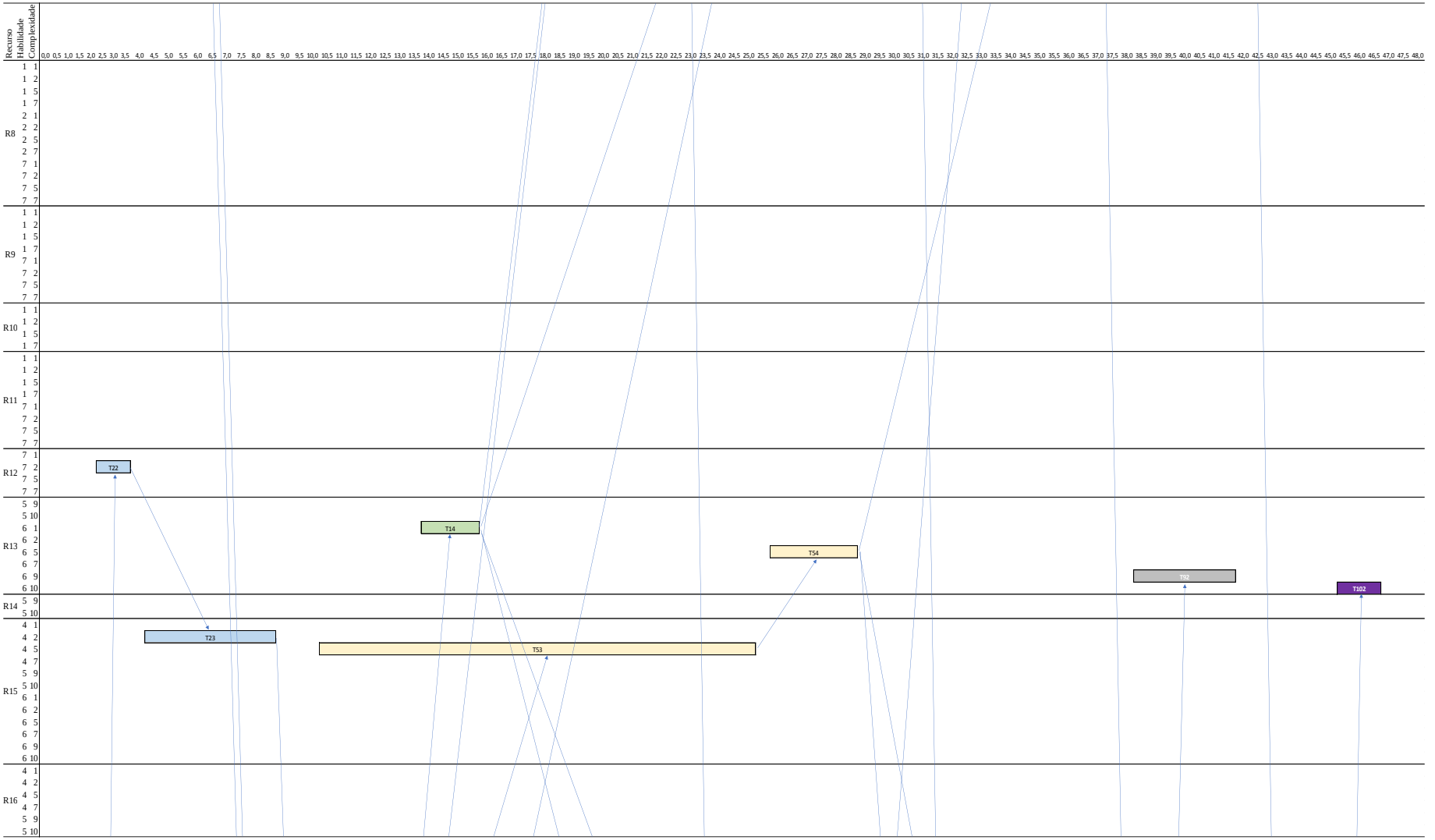


Figura 37. Uma das soluções propostas pelo modelo para a base LFCM

c)

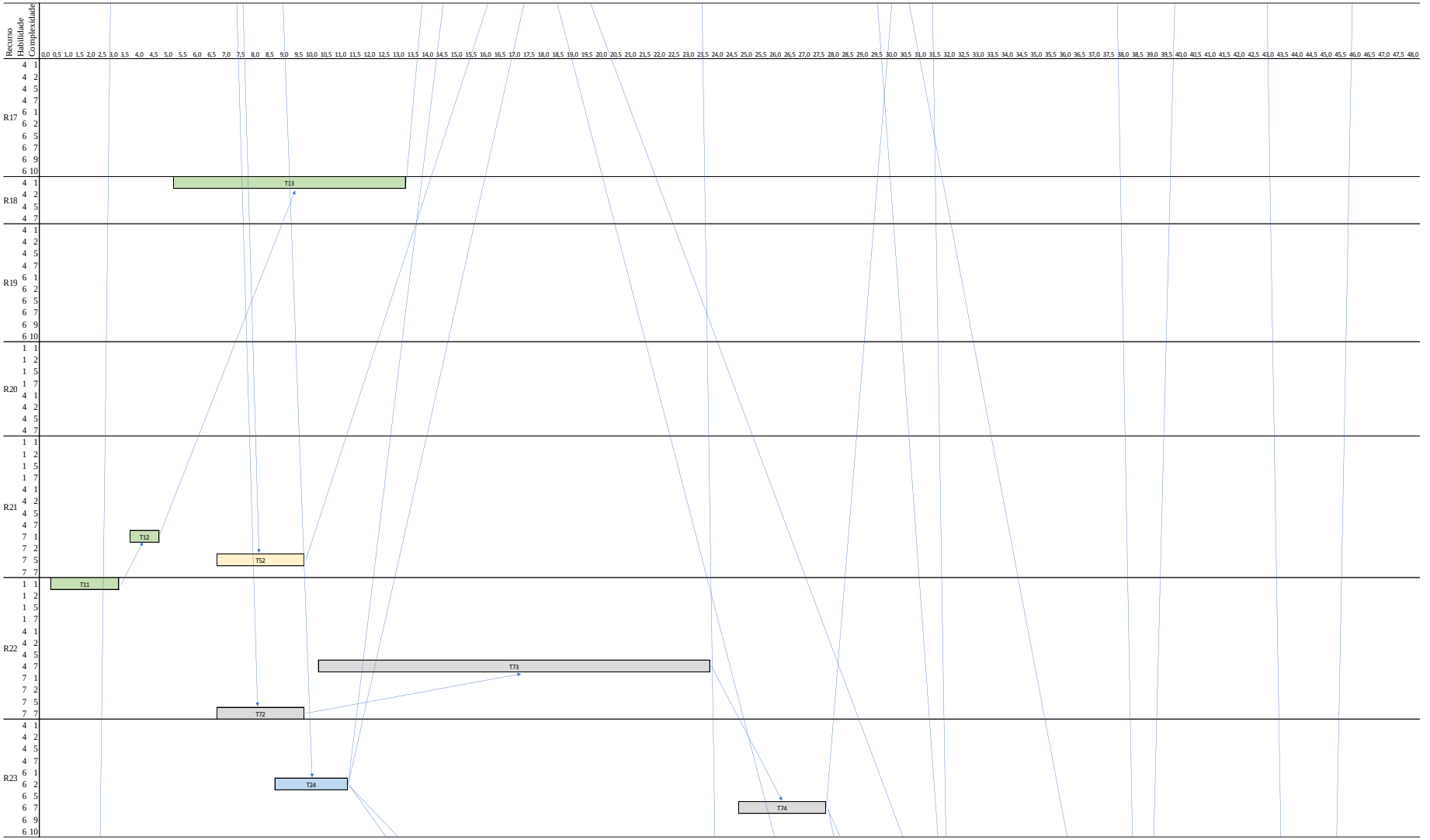


Figura 38. Uma das soluções propostas pelo modelo para a base LFCM

d)

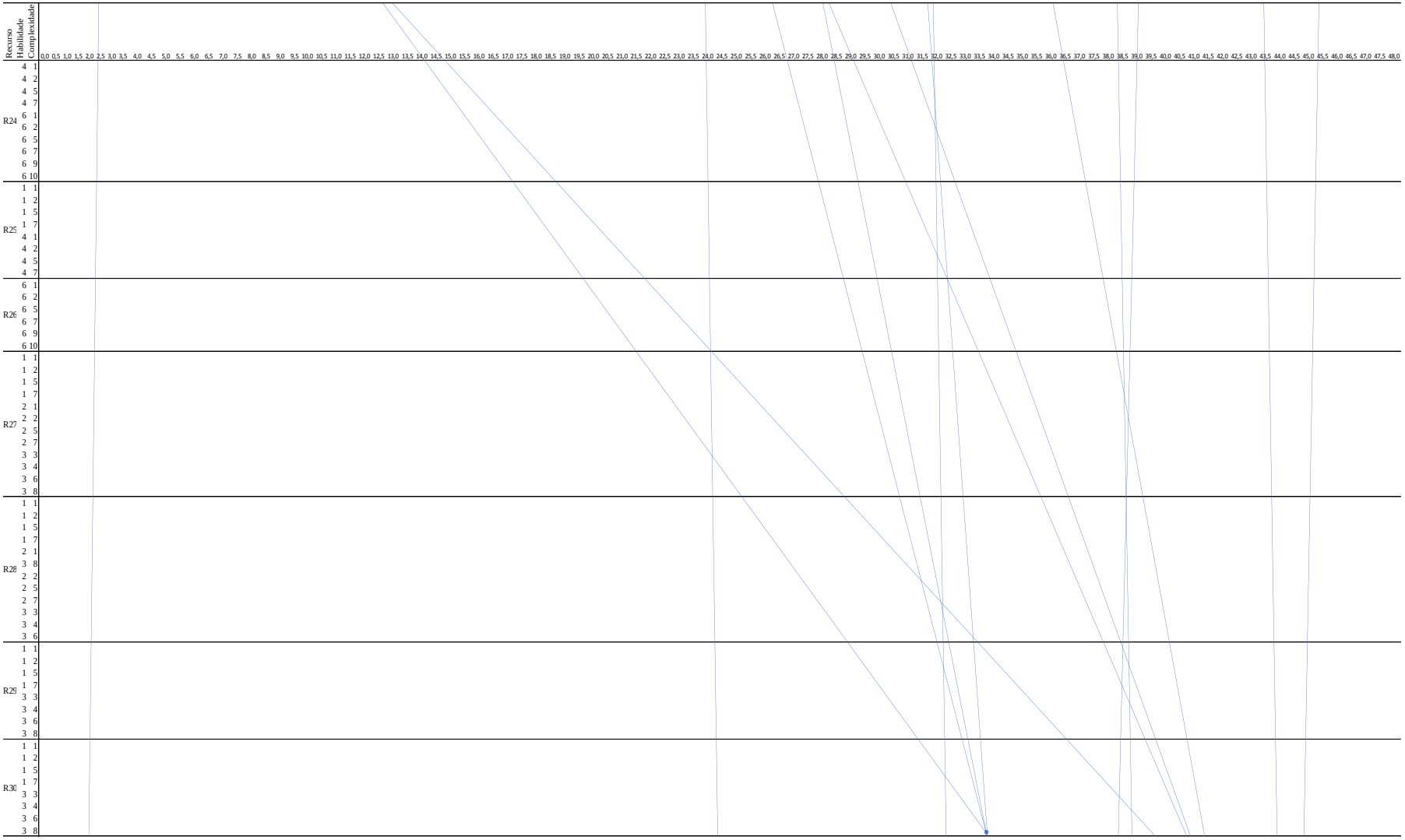
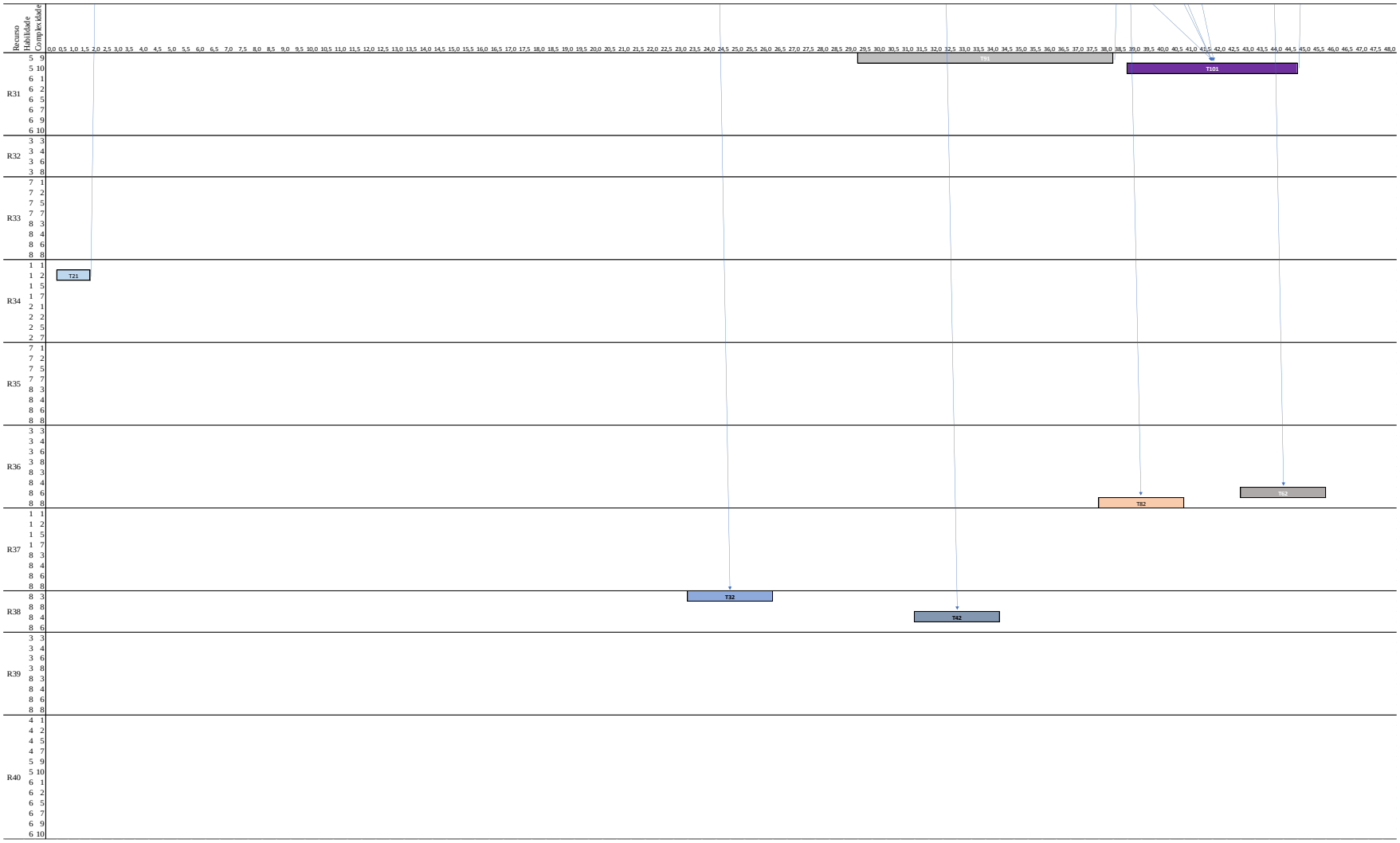


Figura 39. Uma das soluções propostas pelo modelo para a base LFCM

e)



Fonte: Autor