UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE ENGENHARIA DE SOFTWARE

PEDRO FARIAS GÓES DE SOUZA

OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS APLICADA AO ESCALONAMENTO DE TAREFAS COM RESTRIÇÃO DE RECURSOS EM PROJETO DE SOFTWARE

Itacoatiara – Amazonas
Novembro - 2020

PEDRO FARIAS GÓES DE SOUZA

OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS APLICADA AO ESCALONAMENTO DE TAREFAS COM RESTRIÇÃO DE RECURSOS EM PROJETO DE SOFTWARE

Monografia apresentada ao Instituto de Ciências Exatas e Tecnologia da Universidade Federal do Amazonas como parte dos requisitos necessários para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Dr. Rainer Xavier de Amorim

Itacoatiara – Amazonas
Novembro – 2020

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Souza, Pedro Farias Góes de
S729o Otimização por Colônia de Formigas Aplicada ao Escalonamento de Tarefas com Restrição de Recursos em Projeto de Software / Pedro Farias Góes de Souza . 2020
100 f.: il. color; 31 cm.

Orientador: Rainer Xavier de Amorim
TCC de Graduação (Engenharia de Software) - Universidade Federal do Amazonas.

1. Otimização. 2. Escalonamento de tarefas. 3. Projeto de Software. 4. Colônia de formigas. I. Amorim, Rainer Xavier de. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Curso de Bacharelado de Engenharia de Software

FOLHA DE APROVAÇÃO

PEDRO FARIAS GÓES DE SOUZA

OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS APLICADA AO ESCALONAMENTO DE TAREFAS COM RESTRIÇÃO DE RECURSOS EM PROJETO DE SOFTWARE

Monografia apresentada ao Instituto de Ciências Exatas e Tecnologia da Universidade Federal do Amazonas como parte dos requisitos necessários para a obtenção do título de Bacharel em Engenharia de Software.

Aprovada em 24 de Novembro de 2020

BANCA EXAMINADORA

Prof. Dr. Rainer Xavier de Amorim, Presidente
Universidade Federal do Amazonas

Prof. Dr. Carlos Alberto Oliveira de Freitas, Membro
Universidade Federal do Amazonas

Prof. Me. Marcos Thomaz da Silva, Membro
Universidade Federal do Acre

Folha de Aprovação assinada pela Profa. Odette Mestrinho Passos, responsável pela disciplina de Trabalho de Conclusão de Curso (Período: 2020/ERE), onde atesta a defesa do(a) aluno(a) e a presença dos membros da banca examinadora.

Documento assinado eletronicamente por **Odette Mestrinho Passos**, **Professor do Magistério Superior**, em 04/12/2020, às 17:27, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0377573** e o código CRC **859659BC**.

Rua Nossa Senhora do Rosário - Bairro Tiradentes nº 3836 - Telefone: (92) (92) 99318-2549
CEP 69103-128 Itacoatiara/AM - ccesoicet@ufam.edu.br

AGRADECIMENTOS

Agradeço a todos que contribuíram ao decorrer desta grande e importante fase da minha vida, em especial ao meu pai, Adrianilson, e minha mãe, Rosana, que são os meus eternos amores e inspirações da minha vida, que sem eles eu nada seria, e faço tudo por eles. Ao meu orientador, Prof. Dr. Rainer Xavier, que me orientou neste trabalho e não mediu esforços para me ajudar. Aos meus amigos e colegas que de alguma forma e em algum momento me ajudaram e estenderam a mão quando precisei. A todos os professores pelo ensinamento repassado e motivação para o meu sucesso profissional. A Deus pela sua proteção, benção e seu grandioso amor comigo e minha família.

"Se cheguei até aqui foi porque me apoiei no ombro de gigantes."

Isaac Newton

RESUMO

Devido aos notáveis avanços tecnológicos que a área da tecnologia da informação vem obtendo com o passar dos anos, o mercado consumidor se mostra exigente com a indústria de produtos de software. Sendo assim, a conclusão de um projeto na área de software dentro do prazo estimado na fase inicial garante o entusiasmo da equipe, podendo dar início a outro em seguida, aumentando a experiência da empresa e sua visibilidade de competência no mercado. Com isso, o objetivo deste trabalho é investigar sobre e o Problema de Escalonamento de Tarefas com Restrição de Recursos adotando como estratégia heurística a Otimização por Colônia de Formigas, e ao final apresentar um algoritmo competitivo com as estratégias existentes na literatura. O problema em questão consiste em organizar várias atividades que podem ser atribuídas a vários recursos, podendo estes ser renováveis ou não, a fim de cumprir os objetivos do projeto. As metodologias de pesquisas adotadas neste trabalho serão o Mapeamento Sistemático, usado para identificar, classificar e analisar evidências que tem relação com determinadas questões de pesquisa, tópicos ou qualquer outro assunto relacionado e a outra metodologia será a utilização de abordagens empíricas de implementação de uma heurística, proporcionando a experimentação e análise empírica dos resultados obtidos. Dessa forma, como resultado, destacam 11 estratégias de otimização identificadas para as mais diversas instâncias propostas para o problema, incluindo instâncias de benchmark. Além da proposição de uma estratégia de otimização baseada em colônia de formigas, que apresenta resultados promissores para instâncias de 30, 60 e 120, considerando a minimização do *Makespan*.

Palavras-Chave: Escalonamento de Tarefas. Otimização por Colônia de Formigas. Projetos de Software.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACO	*Ant Colony Optmization*
ACS-SPSP	*Ant Colony Six - Software Project Scheduling Problem*
CPU	*Central Processing Unit*
ES	Engenharia de Software
GA	*Genetic Algorithm*
GQM	*Goal/Question/Metric*
GRASP	*Greedy Randomized Adaptive Search Procedure*
HAntCO	*Hybrid Ant Colony Optmization*
HC	*Heurística Construtiva*
ILS	*Iterated Local Search*
JDK	Java Development Kit
MAC	*Mode Assignment Compression*
MRCPSP	*Multi-mode-Resource-Constrained Project Scheduling Problem*
MS	Mapeamento Sistemático
MS-RCPSP	*Multi-skills-Resource-Constrained Project Scheduling Problem*
NSGA-II	*Non-dominated Sorting Genetic Algorihm II*
OC	Otimização Combinatória
OE	Objetivo Específico
PCV	Problema do Caixeiro Viajante
PETRR	Problema de Escalonamento de Tarefas com Restrição de Recursos
PICO	*Population, Intervention, Comparison e Outcomes*
PMBOK	*Project Management Body of Knowledge*
PMI	*Project Management Institute*
PPPRR	Problema da Programação de Projetos com Restrição de Recursos
PR	*Path-Relinking*
PRV	Problema de Roteamento de Veículos
PSP	*Problem Scheduling Project*
PSPLIB	*Library Problem Scheduling Project*
RAM	*Random Access Memory*
RCPSP	*Resource-Constrained Project Scheduling Problem*
RVND	*Randon Variable Neighborhood Descent*
SBSE	*Search-based Software Engineering*

SERPRO	Serviço Federal de Processamentos de Dados
SPSP	*Software Project Scheduling Problem*
SPT	*Shortest Processing Time*

SUMÁRIO

1. INTRODUÇÃO

Neste capítulo são apresentados alguns aspectos que compõem este trabalho, descrevendo a sua Contextualização, Justificativa, Objetivos e o Método adotado.

1.1 Contextualização

Para Nan e Harter (2009) o sucesso dos projetos nas empresas dependem da elaboração de planos de projetos eficientes para reduzir o custo de construção de software. Na área de gerenciamento de projetos, o gerente faz a estruturação de como o projeto vai se desenvolver, executa junto com sua equipe as atividades necessárias, com o intuito de obter um produto final competitivo no mercado, e ao fim de tudo isso, finaliza o projeto dentro do prazo pré-estabelecido. Nesse cenário que a otimização das tarefas do projeto se mostra como fator essencial.

Logo, devido aos notáveis avanços tecnológicos que a área da tecnologia da informação vem obtendo com o passar dos anos, o mercado consumidor se mostra exigente com a indústria de produtos de software. Com isso, a complexidade para se obter um produto com a qualidade exigida pelos consumidores é cada vez maior, e uma forma de garantir essa qualidade é dando uma atenção especial a otimização das atividades (LOPES e TOLFO, 2018).

Assim, a gestão de projetos se baseia, basicamente, em três pilares, que são: foco na necessidade do cliente, equipe engajada de forma produtiva e colaborativa e a administração de recursos. A abordagem de gerenciamento de projetos é relativamente moderna, caracterizada por métodos de reestruturação da administração e adaptação de técnicas especiais de gestão, com o objetivo de obter melhor controle e utilização dos recursos existentes (KERZNER, 2011).

Neste contexto, a gestão de projetos de software compreende atividades que tem o objetivo de assegurar que o produto seja entregue ao cliente no prazo pré-definido e de acordo com os requisitos. O *Project Management Body of Knowledge* (PMI, 2017) estabelece uma linguagem comum, servindo de referência básica para qualquer um que se interesse pelo Gerenciamento de Projetos e, como tal, não deve ser encarado como um documento que contemple a totalidade do conhecimento de Gerenciamento de Projetos (DUARTE, 2017).

Dessa forma, quanto mais cedo um projeto é concluído, o entusiasmo da equipe tende a aumentar. Quando um projeto é concluído dentro do prazo estimado, pode-se dar início a outro, aumentando a experiência das empresas e sua visibilidade de competência no mercado. O Gerenciamento do Cronograma é um dos processos na realização de um projeto, e é constituído por um conjunto de subprocessos que interagem com outras etapas do projeto, sendo uma área que não pode ser negligenciada, de acordo com PMBOK® *Guide* (PMI, 2017).

Portanto, se houver um tempo pré-estabelecido para se realizar os estudos técnicos e se for executado um controle rigoroso, quanto ao prazo de entregas e planejamento dos anteprojetos, deve-se ter uma previsão de custos temporais para um possível atraso cronológico do projeto. Um projeto que atrasa produz aumento de custos por conta dos custos fixos, podendo inviabilizar o projeto, além de contribuir para criar ou aumentar a insatisfação de todas as partes interessadas (NASCIMENTO et al., 2017).

Com isso, o problema investigado nesta pesquisa, pode ser solucionado como um Problema de Escalonamento de Tarefas com Restrição de Recursos – PETRR, também conhecido na literatura como Problema da Programação de Projetos com Restrição de Recursos - PPPRR (RCPSP – *Resource-Constrained Project Scheduling Problem*). Este problema consiste em organizar várias atividades que podem ser atribuídas a vários recursos, podendo estes ser renováveis ou não, a fim de cumprir os objetivos do projeto. A organização e execução de tarefas dentro de qualquer projeto demanda uma atenção especial em qualquer tipo de organização e equipe (CRAWFORD et.al, 2014).

Portanto, busca-se uma melhor maneira de definição de alocação de recursos dentro de uma dada ordem, de forma básica, os problemas de escalonamento de tarefas geralmente consideram alocações pré-estabelecidas e destinam-se apenas em listar a melhor ordem de execução das tarefas (ROCHA, 2011). De acordo com Dridi et al. (2014), o PETRR tem sido considerado um dos problemas de otimização combinatória mais intrigantes, e tem atraído a atenção considerável da comunidade de pesquisa operacional.

As metodologias de pesquisas adotadas neste trabalho, para coletar as informações de forma a cumprir o objetivo, estão fundamentadas nos princípios da Engenharia de Software Experimental que se baseia na condução de um estudo secundário: Mapeamento Sistemático (MS) Kitchenham e Charters (2007). E nas abordagens empíricas de implementação das meta-heurísticas, experimentação e análise empírica dos resultados baseadas nos trabalhos de Kramer e Subramanian (2015b) e Amorim (2017).

1.2 Justificativa

A demora da entrega das atividades estabelecidas em relação as datas planejadas é um dos maiores problemas enfrentados pelos projetos. A pesquisa do *Project Management Institute* (PMI) com 400 organizações em 9 países, relata que 69% das organizações tem grandes obstáculos relacionados ao cumprimento dos prazos, mostrando seu papel essencial na gestão organizacional. O gerenciamento do cronograma em projetos, nos métodos tradicionais considera, para a elaboração do cronograma, as variáveis de escopo e custos (MAIA, 2018).

Segundo Silva et al. (2017) os principais aspectos do problema de definição de equipes e alocação de recursos dentro de um projeto são: os recursos humanos, que é uma demanda intelectual e social essencial nos projetos; as tarefas, que são atividades que precisam ser executadas pelos recursos humanos; interdependências entre tarefas, que é comumente ilustrada nos cronogramas do projeto; e estimativas dos prazos que se divide em estimativa de esforços e alocação de recursos.

A otimização do sequenciamento da alocação de recursos sempre foi uma área relevante da área da administração de projetos que despertou muito interesse no meio científico, ainda mais na computação. A atração no assunto muito se dá devido a possibilidade de redução de tempo da realização dos projetos, e da dificuldade de otimização de problemas reais dentro do ambiente empresarial, resultando na particularidade de sua difícil análise. O que no final se torna um estímulo para a pesquisa de métodos matemáticos que possam viabilizar a solução desse problema (VIEIRA, 2010).

De acordo com Silva et al. (2017), o Problema de Escalonamento de Tarefas com Restrição de Recursos pertence à classe dos problemas combinatórios; os procedimentos para solucioná-lo por sua vez são oriundos da otimização combinatória. Uma gama de programas matemáticos tem sido proposta para solucionar esse problema, levando ao desenvolvimento de algoritmos para tal finalidade.

A Otimização por Colônia de Formigas (*Ant Colony Optmization*, ACO) está entre as principais meta-heurísticas usadas para problemas de otimização combinatória, obtendo resultados consistentes e satisfatórios. Tal otimização tem essa denominação devido a ser baseada no comportamento e estrutura do funcionamento de colônias de formigas reais e como elas encontram rotas mais curtas de sua “casa” até as suas fontes de alimentos (BARBOSA et al. 2015).

O presente trabalho se torna motivo de pesquisa pela oportunidade de lidar com dilemas de alocação de recursos dentro de uma dada ordem, envolvendo a minimização das penalidades da desorganização das execuções das tarefas com restrição de recursos, resultando em não atraso no desenvolvimento nos projetos de software. Isso através de um enfoque teórico-computacional, utilizando a otimização por meta-heurística colônia de formigas (AMORIM, 2017).

1.3 Objetivos

- **Geral**

  Aplicar a estratégia de otimização por colônia de formigas para o escalonamento de tarefas com restrições de recursos a fim de se obter cronogramas eficientes em projetos de software.

- **Específicos**

  Identificar as abordagens de otimização existentes na literatura, baseadas em escalonamento de tarefas, para projetos de software.

  Levantar as instâncias de teste e resultados da literatura para fins de comparação.

  Implementar uma estratégia de otimização para a obtenção de cronogramas eficientes em projetos de software e comparar os resultados com a literatura.

1.4 Metodologia

A seguir encontram-se detalhadas as metodologias a serem utilizadas para cada objetivo específico (OE) definido neste trabalho:

- OE1: Identificar as abordagens de otimização existentes na literatura, baseadas em escalonamento de tarefas, para projetos de software.
- OE2: Levantar as instâncias de teste e resultados da literatura para fins de comparação.

Para a execução dos dois primeiros objetivos, será utilizado o método Mapeamento Sistemático (MS), o qual foi desenvolvido por Kitchenham e Chartes (2007) e definido em três etapas: (a) Planejamento do Mapeamento: nesse passo, os objetivos da pesquisa são listados e o protocolo do mapeamento é definido; (b) Condução do Mapeamento: durante essa fase, as

fontes para o mapeamento são selecionadas, os estudos são identificados, selecionados e avaliados de acordo com os critérios estabelecidos no protocolo do mapeamento e (c) Resultado do Mapeamento: nessa fase, os dados dos estudos são extraídos e sintetizados para serem publicados.

O MS fornece uma visão geral de uma da pesquisa a ser realizada, identificando a quantidade, os resultados das pesquisas realizadas e disponíveis, além de estatísticas de publicações ao longo do tempo identificando as tendências (PETERSEN et al., 2008).

Kitchenham e Charters (2007), afirmam que estudos de MS em engenharia de software têm sido recomendados, sobretudo para áreas de pesquisa onde é difícil visualizar a gama de materiais, relevantes e de alta qualidade, que possam estar disponíveis. Assim sendo, a escolha do MS como proposta para a condução desta pesquisa, justifica-se pelo fato do objetivo ser apenas identificar e utilizar os resultados obtidos para futuras pesquisas. O MS foi baseado no *guidelines* desenvolvido e definido em três etapas, conforme a Figura 1:

Figura 1- Etapas do Mapeamento Sistemático

Fonte: O autor (2020).

- OE3: Implementar uma estratégia de otimização para a obtenção de cronogramas eficientes em projetos de software e comparar os resultados com a literatura.

Para realizar o terceiro objetivo específico serão utilizadas abordagens de implementação de uma estratégia baseada em otimização, experimentação e análise empírica dos resultados baseadas nos trabalhos de Krammer e Subramanian (2015) e Amorim (2017). As instâncias obtidas no segundo objetivo específico serão utilizadas nos experimentos computacionais que serão realizados. Os resultados obtidos serão analisados e comparados com a literatura.

1.5 Organização do Trabalho

Capítulo 1 – Introdução: apresenta alguns aspectos que compõem este trabalho, descrevendo a sua Contextualização, Justificativa, Objetivos e o Método adotado. Além deste Capítulo, outros cinco capítulos que compõem este trabalho, organizados da seguinte forma:

Capítulo 2 – Fundamentação Teórica: é apresentado o referencial teórico que fundamenta os conceitos básicos utilizados nesta pesquisa e os trabalhos relacionados, além de apresentar um comparativo entre a proposta com os trabalhos relacionados.

Capítulo 3 – Mapeamento Sistemático sobre o Escalonamento de Tarefas com Restrição de Recursos: é apresentado o protocolo do Mapeamento Sistemático realizado, incluindo as diretrizes, como foi realizado o processo de condução e levantamento das publicações para esta pesquisa, juntamente resultados obtidos após a pesquisa, que mostra as respostas às principais questões de pesquisa levantadas.

Capítulo 4 – Algoritmo de Otimização por Colônia de Formigas: é apresentada a estratégia de otimização estudada e implementada nesta pesquisa, incluindo pseudocódigos, esquemas e exemplos de implementação.

Capítulo 5 – Experimentos Computacionais e Resultados: neste capítulo são discutidos os experimentos computacionais com instâncias de *benchmark* realizados, juntamente com as comparações com os principais resultados existentes na literatura.

Capítulo 6 – Conclusão e Perspectivas Futuras: Apresenta as considerações finais, resultados obtidos, contribuições do trabalho, as limitações e futuras perspectivas para a continuidade desta pesquisa.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os principais conceitos relacionados a Gerenciamento de Projetos de Software, O Problema da Elaboração de Cronograma e Otimização Combinatória em Engenharia de Software, que serviram de base para esta pesquisa, além disso, serão apresentados os trabalhos relacionados.

2.1 Conceitos Relacionados

2.1.1 Gerenciamento de Projeto de Software

No século XX, surgiu a necessidade de maximizar a produção e minimizou-se o pensamento de aquisição de mão de obra e, consequentemente, as horas de serviço dessa mão de obra. Tiveram dois grandes pensamentos na época, o de “Frederick Taylor (1856-1915)” e o de “Henry Grantt (1861-1919)”; o primeiro aplicou a ideia de que o trabalho poderia ser analisado e melhorado focando em suas partes elementares; o segundo criou a técnica de traçar a sequência de tarefas e tal duração (SANTOS, 2018).

O Gerenciamento de Projetos envolve aplicação de habilidades de conhecimentos, matérias e técnicas às atividades de um projeto, que para atender as suas exigências precisa atingir as metas de qualidade e desempenho, na maioria das vezes, delimitado por cronograma, restrições de recursos e tendendo ao mais baixo custo possível. Trata-se da estratégia das organizações para que possam unir seus resultados dos projetos com os objetivos do negócio (PMI, 2017).

Não muito longe dessa definição, Pereira (2017) afirma que gerenciamento de projeto é um empreendimento não repetitivo caracterizado por uma sequência objetiva, clara e lógica de eventos, com início, meio e fim, que tendem a atingir um objetivo definido, sendo desenvolvido por recursos humanos dentro de parâmetros pré-estabelecidos de tempo, custos, recursos envolvidos e qualidade.

Dessa forma, o Gerenciamento de Projetos de Software compreende muitas atividades essenciais para o êxito de um projeto, como: disponibilidade e alocação de recursos, custos e estimativas, gestão de qualidade, planejamento dos segmentos do projeto. Tais atividades na maioria das vezes são concorrentes e conflitantes, por exemplo, custo e duração de um projeto;

além de serem complexos, percorrem todas as etapas de um processo de produção de um software (REZENDE, 2019).

Logo, um dos objetivos a serem alcançados como forma de pontos que garantem a obtenção do êxito no projeto é o poder de minimização do custo e duração de um projeto. Uma vez que a maior parte do tempo do planejamento é a delimitação do orçamento e definição do prazo para a conclusão do projeto. Sendo assim, surge a ideia de planejar o cronograma, que é um dos dez gerenciamentos listados no PMBOK, e um dos mais complexos para se controlar mediante as mudanças que ocorrem durante o desenvolvimento do projeto (VEGA-VELAZQUEZ et al., 2018).

O PMI (2017) define o cronograma do projeto como um documento técnico que apresenta todas as datas planejadas para a execução de cada atividade. É uma ferramenta de comunicação e gerenciamento que possui informações detalhadas que servem de guia para uma direção, onde o projeto está obtendo êxito e erros. Tal documento é composto por seis etapas, que são: planejar o gerenciamento do cronograma, definir as atividades, sequenciar as atividades, estimar duração de atividades, desenvolver e controlar o cronograma.

Segundo Pressman (2005), cronograma é uma atividade que distribui os esforços estimados em todo o prazo estipulado do projeto, alocando esforços e tarefas de engenharia de software de característica específica. Com as tarefas do projeto já definidas e os esforços necessários já estimados, a alocação de recursos é uma atividade que consiste em alocar o empenho dedicado às tarefas e arranjá-las num cronograma planejado para o projeto.

Com isso, o gerenciamento do cronograma envolve todos os processos indispensáveis para executar o projeto dentro de um tempo estimado. Os gerentes de equipes devem ter um controle eficiente das datas estipuladas para consertar possíveis problemas inesperados que possam comprometer os prazos determinados, impedindo que essas situações possam se agravar e se tornar algo que não possa mais se reverter na execução do projeto (DANTAS, 2019).

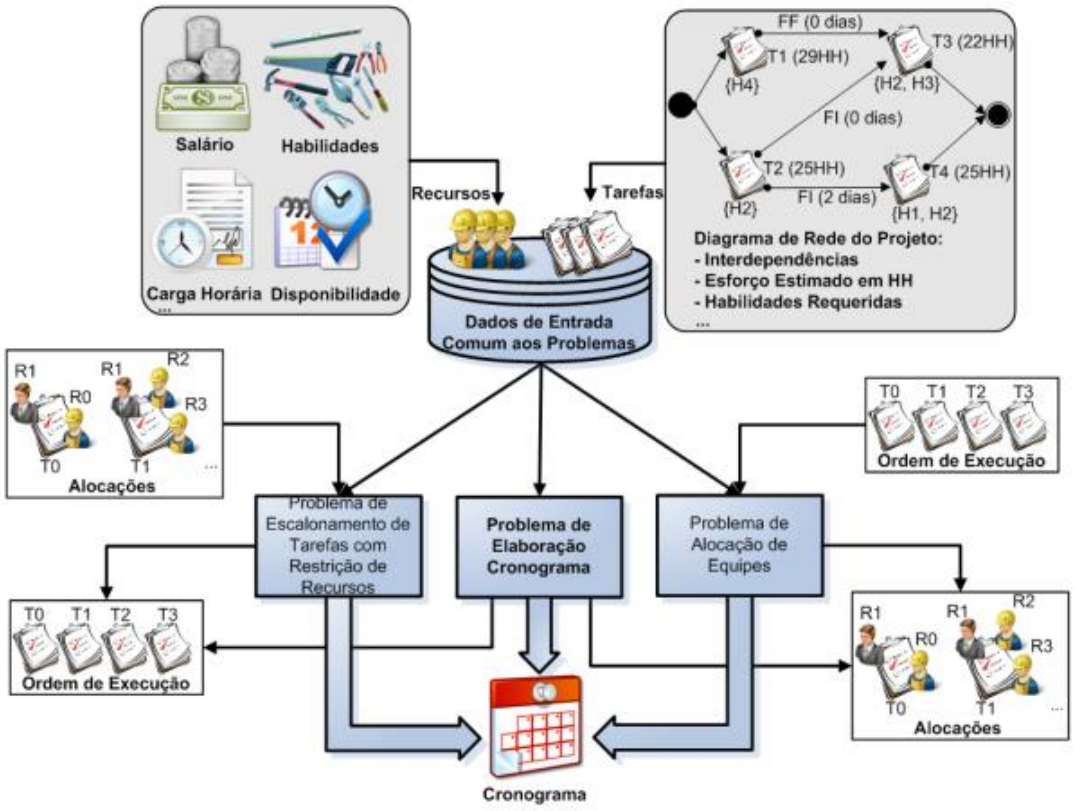
Segundo Andrade et al. (2019), o mal gerenciamento do cronograma causa incertezas e riscos, que por sua vez causam atrasos e estouros de orçamento durante a execução do projeto. Com isso, a elaboração de um cronograma é um aspecto importantíssimo na fase de controle do desenvolvimento de um projeto de software que se estende por toda a sua produção; e todo esse monitoramento tem o objetivo de prever o resultado do projeto e tomar ações corretivas quando necessário.

2.1.2 O Problema da Elaboração do Cronograma

O cronograma do projeto é um dos campos de planejamento de projetos mais importantes e amplamente utilizados. A aplicação de teorias em problemas reais mostra-se muito importante, especialmente quando os estudos nesse campo avançam e aumentam. O cronograma do projeto possui uma gama de questões e especificações, tornando-se complexo para cada tipo de problema de elaboração; uma vez que os problemas da elaboração do cronograma são conhecidos como NP-Difíceis, diferentes heurísticas e meta-heurísticas são apresentadas como forma de solução destes problemas (KHALILZADEH et.al, 2017).

O Problema da Elaboração do Cronograma, compreende a junção de dois problemas conhecidos na literatura, são eles: o Problema de Alocação de Equipes e o Problema de Escalonamento Tarefas com Restrições de Recursos. O problema de alocar uma esquipe se trata da atribuição das responsabilidades de uma tarefa a um conjunto de recursos humanos, e o escalonamento de tarefas determina a ordem da execução de cada tarefa, como pode ser observado na Figura 2 (TRICHES et al., 2015).

Figura 2 - O Problema da Elaboração de Cronograma



Fonte: Rocha (2011).

Chand et. al (2018) usam a nomenclatura de Problema de Agendamento de Projetos (PSP) para definir o conceito, que para ele é um conjunto de atividades não-preemptivas e relacionadas à precedência que precisam ser agendadas. A estrutura do projeto pode ser representada por um grafo de atividades, sendo que cada atividade tem sua representação por nós, e os arcos são as restrições de precedência; tais restrições impõem que determinadas atividades não sejam executadas antes que atividades anteriores tenham sido concluídas.

Segundo Rocha (2011), o Problema de Elaboração do Cronograma recebe os dados em comum como entrada, e os seus objetivos consistem em determinar a melhor forma de alocação de recursos vinculado ao melhor jeito de ordenar as atividades para sua execução. Com os valores de cada recurso, suas atribuições e a melhor ordem de execução das tarefas, pode-se estimar o instante de início de execução de cada tarefa separadamente e, com isso, tem-se o cronograma estabelecido por completo.

Muitas ferramentas e modelos são utilizados para a criação de cronogramas em projetos dos mais variados tipos. Ainda que vários artefatos ajudem no desenvolvimento de um cronograma, muitos deles não consideram particularidades significantes dos projetos de software, o que leva que esses cronogramas sejam elaborados conforme experiências e aprendizados de projetos passados de software (COLARES, 2010).

A análise dos atrasos no cronograma é um problema permanente e recorrente de aplicação prática em gerenciamento de projetos. Os atrasos influenciam diretamente no sucesso final dos projetos, ainda mais quando grandes valores financeiros estão em jogo, e o tempo e custo são variantes sensíveis e dependentes uns dos outros. Relações de atraso é uma questão muito conhecida, e que leva, a longas negociações e até mesmo casos jurídicos, resultando em problemas sérios a ambas as partes interessadas no projeto (GUIDA, 2019).

2.1.3 Otimização Combinatória em Engenharia de Software

Nesta seção serão apresentados os principais conceitos de Otimização Combinatória (OC), Engenharia de Software Baseada em Busca, que envolve a aplicação de OC para solucionar muitos problemas de Engenharia de Software e, por último, são apresentadas as estratégias de solução baseadas em Heurísticas e Meta-heurísticas.

2.1.3.1 Otimização Combinatória

Otimização Combinatória se encontra dentro do problema da otimização matemática, esta área de pesquisa trata de um tipo específico de problema em que procura um conjunto de

soluções factíveis para sua representação dentro de um espaço de busca discreto. Onde o objetivo se trata de encontrar, dentro das soluções possíveis, aquela solução ou conjunto soluções que o problema requer, podendo ser de maximização ou minimização de uma determinada função objetivo ou multiobjetivo, respeitando suas restrições (AMORIM, 2013).

Becceneri et al. (2007) diz que a otimização combinatória é uma área que engloba grandes quantidades de problemas e que busca por soluções que tragam o melhor resultado e uso dos recursos envolvidos. Esses problemas podem ser divididos em três categorias: aqueles cujas variáveis assumem valores reais (ou contínuos), aqueles cujas variáveis assumem valores discretos (ou inteiros) e aqueles em que há variáveis inteiras e contínuas, classificados, respectivamente, como problemas de Otimização Contínua, Otimização Combinatória ou Discreta, e Otimização Mista.

Nestes tipos de problemas não basta só ter um conjunto de soluções possíveis, mas também a otimização dos objetivos de interesse. Os problemas de otimização combinatória podem ser de minimização ou maximização, em ambos os casos há uma função aplicada a um domínio finito, que no geral é enumerável. Apesar de finito, o domínio da função é em geral grande, podendo ser aplicados algoritmos que testam solução por solução deste domínio, se tornando assim impraticáveis (MIYAZAWA e SOUZA, 2018).

Um grande e conhecido problema da otimização combinatória é o Problema do Caixeiro Viajante (PCV), que pode ser entendido como um viajante que procura a melhor forma de reduzir seus custos de mudanças, através da disposição das cidades que ele irá visitar, buscando a melhor rota para que possa visitar todas as cidades com o menor custo de tempo possível (BASTOS, 2017).

Outro grande problema dentro da área de otimização combinatória é o Problema de Roteamento de Veículos (PRV) que, segundo Junior (2017), o PRV consiste em projetar as rotas de veículos com o intuito de minimizar o custo total, sendo que cada veículo começa e termina em sua respectiva garagem, garantido assim que cada local seja visitado exatamente uma vez e a demanda total de qualquer rota não exceda a capacidade total de veículos da mesma. Este problema é tido como um dos mais complexos da área de otimização, embora seja um dos mais estudados na literatura.

Mais um problema muito conhecido na área de otimização combinatória é o Escalonamento de Tarefas, que trata da alocação eficiente de aplicações paralelas sobre os recursos disponíveis em um sistema distribuído. O objetivo mais comum do escalonamento é

minimizar o tempo de execução esperado de uma tarefa, ou de um conjunto de tarefas, geralmente esse problema se expande com a entrada da complexidade de restrição de recursos computacionais (CAIXETA, 2018).

2.1.3.2 Engenharia de Software Baseada em Busca

Dentro da Engenharia de Software (ES), existe a subárea de Engenharia de Software Baseada em Busca, do inglês *Search-based Software Engineering* (SBSE), que se dedica a resolver os problemas de otimização aplicando algoritmos de busca. O termo se intensificou em um trabalho de 2001 feito por Haman e Jones, onde foram apresentados pontos de pensamentos sobre a aplicação de meta-heurísticas em problemas inerentes ao desenvolvimento de software (NETO, 2016).

A SBSE surgiu como o objetivo de promover estudos científicos que abordassem algoritmos de busca para tratar questões na área de ES, através de técnicas matemáticas que poderiam ser empregadas em diversos problemas do ciclo de vida da engenharia de software. Sua aplicação seria em ferramentas e métodos de codificação e design; requisitos e especificação de software, se estendendo até manutenção (LIMA, 2018). Conforme a Tabela 1 abaixo:

Tabela 1 - Áreas e problemas em SBSE

ÁREAS E PROBLEMAS EM SBSE	
Área de Engenharia de Software	**Principais Problemas**
Engenharia de Requisitos	Análise de Requisitos
	Seleção de Requisitos
	Planejamento de Requisitos
Teste de Software	Geração de Dados de Teste
	Seleção de Casos de Teste
	Priorização de Casos de Teste
	Testes Não Funcionais
Estimativa de Software	Estimativa de Tamanho
	Estimativa de Custo
Planejamento de Projeto	Alocação de Recursos
	Alocação de Pessoal
Otimização de Código-Fonte	Paralelização
	Otimização para Compilação
Manutenção de Software	Reengenharia de Software
	Automated Maintenance
Otimização de Compilador	Alocação em *Heap*
	Tamanho de Código
Projeto de Software	Modularização

Fonte: Adaptado de Freitas et al. (2010).

Para Santos (2017) a Engenharia de Software Baseado em Busca, define o termo "busca", que é usado para se referir a otimização de meta-heurística, baseada em técnicas de busca que são usadas. O uso do termo "busca" para SBSE, é a busca de soluções ideais ou perto do ideal, que são procuradas em busca de uma função de espaço de soluções possíveis, guiado por um centro de parâmetros que distingue entre a melhor e pior solução.

Em SBSE, procura-se sempre obter uma solução exata para o problema, mas nem sempre é possível encontrá-la devido ao esforço computacional ser grande para identificá-la, torando-se inviável. Com isso, aplicam-se técnicas de busca para resolver os problemas de otimização em Engenharia de Software e, assim, definir tais problemas como problemas de SBSE. Logo, tem-se a definição: a) uma representação do problema; b) um conjunto de operadores de manipulação da solução; e c) uma função de aptidão (Ferreira et. al 2016).

2.1.3.3 Estratégias de Solução baseadas em Metaheurísticas

Nos problemas de otimização, existem estratégias exatas, que resolvem os problemas e retornam a sua solução ótima. Existem também os algoritmos aproximativos/heurísticos, que fornecem uma solução em tempo polinomial, mas nem sempre garantem a otimalidade. Com isso, os problemas NP-Difíceis são mais complexos de serem resolvidos de forma exata, como o problema da elaboração do cronograma investigado nesta pesquisa, tornando-se cada vez mais difícil de resolver à medida que a instância do problema cresce. Logo, os métodos heurísticos/metaheurísticos se mostraram promissores na resolução de problemas NP-Difíceis, mesmo com instâncias elevadas, retornando assim boas soluções em tempo computacional aceitável (KRAMER e SUBRAMANIAN, 2015a).

Dessa forma, Blum e Roli (2003) mencionam que as metaheurísticas podem ser classificadas de diferentes formas, que são apresentadas a seguir:

- Baseadas ou não na natureza: essas metaheurísticas são aquelas que utilizam metodologia para geração de soluções baseadas em comportamentos da natureza e/ou seres vivos, tais como: seleção natural, mutação, colônia de formigas, etc. Caso contrário, são não-baseadas na natureza;
- Baseadas em população ou indivíduos: essa é dependente do número de soluções utilizadas ao mesmo tempo para a geração de novas soluções. Algoritmos que atuam sobre uma única solução são chamados de métodos de trajetória, onde se enquadram as metaheurísticas baseadas em busca local. Já as metaheurísticas baseadas em

população desempenham um processo de busca que descrevem a evolução de um conjunto de pontos no espaço de busca;

- Baseadas em função objetivo estática ou dinâmica: depende de como a metaheurística se comporta mediante seu uso da função objetivo. Algumas modificam a função objetivo, utilizando-se de informações adquiridas durante o processo de busca para escapar de ótimos locais;
- Baseadas em uma ou mais estruturas de vizinhança: considera o número de estruturas de vizinhança utilizadas durante o procedimento de busca local.

Os tópicos a seguir destacam as heurísticas/metaheurísticas que servirão de base para o desenvolvimento da estratégia algorítmica a ser desenvolvida nesta pesquisa, que são: Busca Local Iterada e Algoritmo de Colônia de Formigas.

a) Busca Local Iterada

A estratégia de busca local é baseada em um mecanismo de filtragem muito simples que escolhe cientemente as soluções vizinhas serem utilizadas para a comparação da busca com características próximas a solução atual. Uma das formas de implementar um algoritmo de busca local é percorrer a vizinhança dessa solução em busca de outra de valor maior ou menor, dependendo se o problema é de minimização ou maximização. Se a solução na vizinhança for encontrada, torna-se a solução nova e o algoritmo continua (MAXWELL, 2015).

Dessa forma, a Busca Local possui um comportamento aleatório no início de seu processo de busca e afunila para um método de refinamento à medida que seus resultados se aproximam da solução final. A ideia fundamental da Busca Local é permitir movimentos que resultem em soluções piores que a solução vigente, objetivando escapar de ótimos locais. A probabilidade de realizar tais movimentos é diminuída durante a busca (CHAVES, 2009).

Logo, a heurística de Busca Local é intensificada no Algoritmo de Busca Local Iterada (ILS), que consiste em um processo iterativo, no qual uma solução inicial é perturbada, ou seja, sofre permutações aleatórias, gerando assim novas soluções de partida para um método de Busca Local. A perturbação da solução tem um papel importante, porque uma perturbação pequena pode fazer que o algoritmo não consiga escapar de uma região que tem uma atração para ótimos locais que já foram obtidos. Porém, com uma perturbação muito grande o algoritmo pode ter um reinício totalmente aleatório (CHAVES, 2009).

Assim, Amorim (2013) define ILS como uma solução inicial, onde a cada iteração, a heurística substitui a solução vigente por uma solução vizinha, sendo essa uma função objetivo melhor que a solução vigente. A busca acaba quando as soluções nas adjacências são piores que a solução vigente, e essas soluções adjacentes compõem um subconjunto do espaço de soluções. Tal técnica pode ser vista como uma busca de soluções em um grafo que representa um espaço de busca de soluções.

A forma geral do algoritmo permite dar um passo atrás para dar dois à frente. Porque com o retrocesso, a busca atual passa a procurar a melhor solução na nova área de busca após a perturbação. A justificativa desse retrocesso se dá para garantir que a busca seja em áreas de regiões "interessantes" ou ainda não exploradas dentro do espaço de soluções. Assim, esse recuo ajuda a superar decisões que direcionam a busca para locais que já foram trabalhados (CONGRAM et al., 2002), conforme representado no algoritmo da Figura 3 e ilustrado na Figura 4.

Figura 3 - Pseudocódigo da ILS

```
algoritmo ILS
    gere uma solução inicial s_0
    ŝ ← BuscaLocal (s_0)
    enquanto ( critério de parada não for satisfeito ) faça
        s' ← Perturbação (ŝ , histórico)
        ŝ' ← BuscaLocal (s')
        ŝ ← CritériodeAceitação (ŝ , ŝ', histórico)
    fim-enquanto
fim-algoritmo
```

Fonte: Adaptado de Lourenço et al. (2003).

Figura 4 - Funcionamento do algoritmo ILS em um determinado espaço de busca.

Fonte: Amorim (2013).

O funcionamento do algoritmo da Figura 3 parte gerando uma solução qualquer (S_0), em seguida é aplicada a busca local obtendo uma ótima solução local ($\hat{s}$). Daí se inicia uma nova iteração até que algum critério de parada seja satisfeito, e a cada iteração ocorre uma perturbação aleatória da solução $\hat{s}$ produzindo um novo ponto de partida para a busca local ($s´$). Então se aplica a heurística da busca local sobre $s´$ encontrando a solução ótima para tal região ($\hat{s}´$). A busca continua quando a aplicação de um critério de aceitação decidindo de qual solução será usada para nova busca (CHAVES, 2009).

b) Algoritmo de Colônia de Formigas

Otimização por colônia de formigas, do inglês *Ant Colony Optimization* (ACO) é uma metaheurística baseada no comportamento de formigas em busca de alimento. Proposto inicialmente por Colorni et al. (1992), essa estratégia pertence a uma classe de algoritmos que buscam as melhores soluções candidatas, usando a analogia de trilhas de feromônio artificial e informações heurísticas sobre o problema sob análise (SILVA, 2018).

A Otimização por Colônia de Formigas foi inspirada no comportamento de formigas reais e seu esquema de comunicação usando o rastro de feromônio. Ao procurar por comida, as formigas começam a se mover aleatoriamente, quando uma formiga encontra um caminho para a comida, ela volta reforçando esse caminho com feromônio, fazendo com que as outras formigas sigam esse novo caminho, e isso se repete até que outra formiga encontre um novo caminho mais curto entre a colônia e a comida, assim ela está otimizando a obtenção de comida da colônia (ALBANEZ et al., 2017).

No algoritmo ACO, cada formiga inicia em um ponto no espaço de busca, de forma aleatória, podendo todas as formigas serem colocadas no mesmo nó ou uma em cada nó. Após isso, todas as formigas seguem uma regra de transição até que se encontre uma solução que satisfaça o critério de parada da busca. Depois que todas as formigas tenham encontrado uma solução, dois procedimentos são realizados: o primeiro é baseado em busca local tentando melhorar a solução explorada; e o segundo é a atualização global de feromônios (TAVARES e GODINHO, 2013). O algoritmo apresentado na Figura 5 apresenta os passos de uma estratégia baseada em Colônia de Formigas.

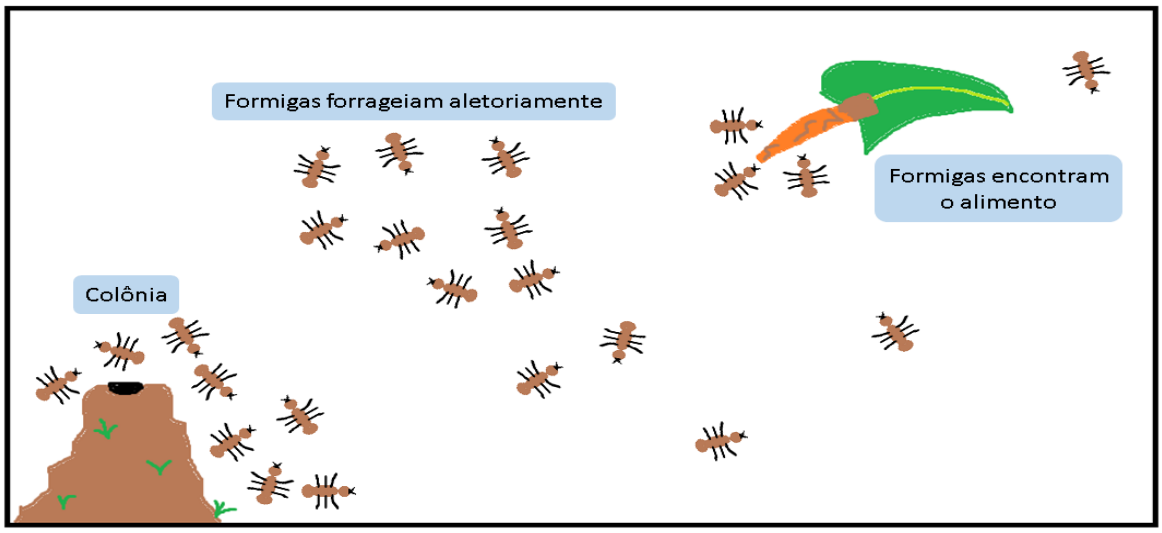
Figura 5 - Algoritmo ACO

```
1  Inicialize
2  Repita Neste nível, cada execução é chamada iteração
3      Repita Neste nível, cada execução é chamada passo
4          Cada formiga aplica uma regra de transição para construir a próxima etapa da solução
5          Aplica-se a atualização local de feromônios
6      Até que todas as formigas tenham criado uma solução completa
7      Aplica-se o procedimento de busca local
8      Aplica-se o procedimento de atualização global de feromônios
9  Até que o critério de parada seja satisfeito
```

Fonte: Dorigo et al. (1996).

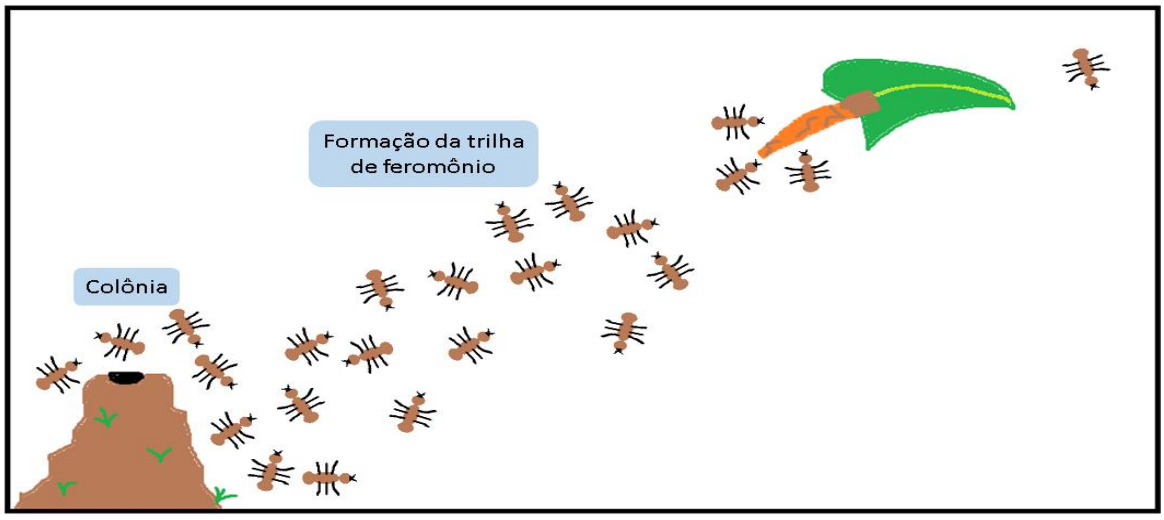
Grande parte das colônias de formigas possuem uma forma indireta e eficaz de se comunicar entre si que é um rastro, que leva as formigas aos alimentos. As formigas produzem uma essência natural, chamado feromônios, que eles deixam no caminho à alimentação a fim de marcá-lo para as demais formigas, tal trilha de feromônio some ao longo do tempo. Por outro lado, a evaporação dessa trilha pode ser reforçada pela passagem de mais formigas deixando seus feromônios. Assim, os caminhos certos que conduzem à comida são finalmente caracterizados por uma pista de feromônio forte, e são seguidos por mais formigas (ANGHINO e PALLOUCI, 2008). As Figuras 6, 7 e 8 ilustram este processo de construção da solução.

Figura 6 - Comportamento das formigas (a)



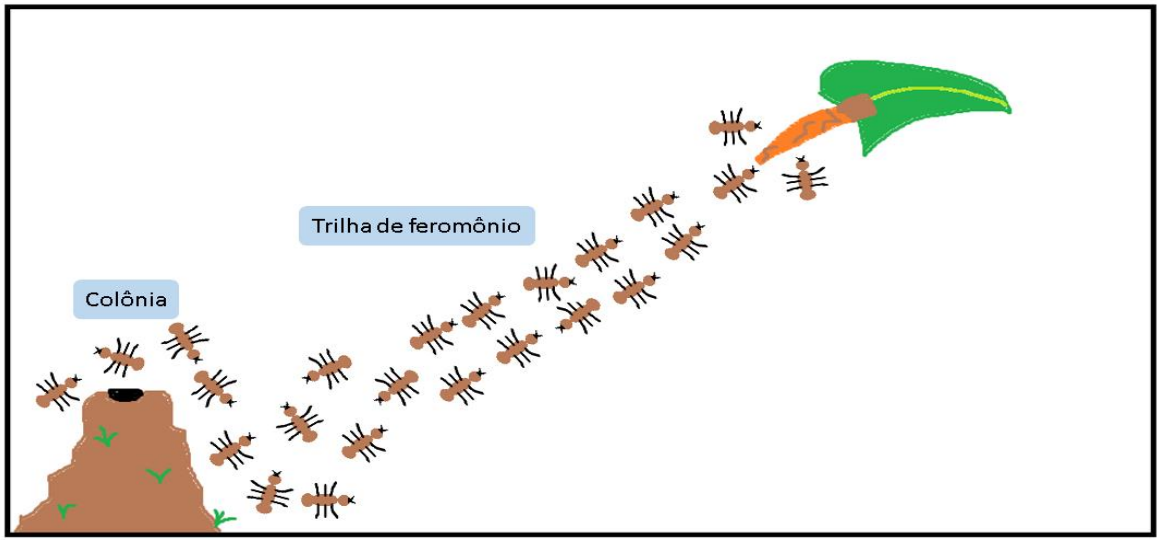
Fonte: O autor (2020).

Figura 7 - Comportamento das formigas (b)



Fonte: O autor (2020).

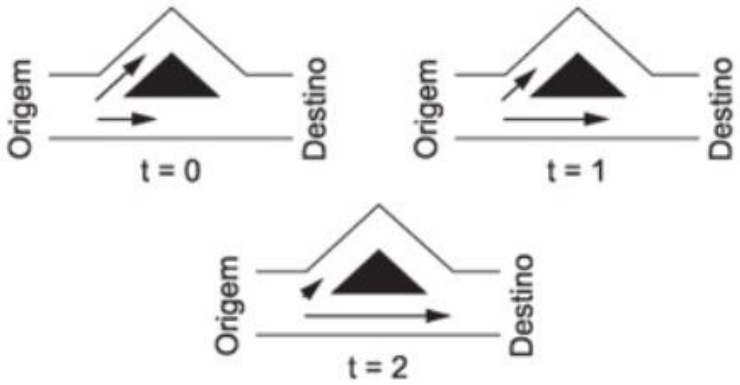
Figura 8 - Comportamento das formigas (c)



Fonte: O autor (2020).

Tavares e Godinho (2013) exemplificaram o caminho feito pelas formigas na Figura 9, onde pode-se notar que no instante $t = 0$, a formiga escolhe aleatoriamente um caminho qualquer disponível. Após chegar no objetivo, uma função pré-definida analisa o caminho realizado, obtendo a qualidade da solução gerada; com isso é realizado um depósito de feromônio artificial. Este depósito de feromônio faz parte da característica especial do algoritmo, que é o reforço positivo, que busca privilegiar as melhores soluções.

Figura 9 - O comportamento das formigas em diferentes momentos de tempo



Fonte: Tavares e Godinho (2013).

Não se pode deixar de falar que o ACO mesmo tendo sua convergência provada, não há estudos na literatura que dão a garantia de sua velocidade de convergência. Sendo assim, a única forma de medir o desempenho do algoritmo é pelo método da exaustão, executando extensos experimentos computacionais (DORIGO e STUTZLE, 2019).

2.1.4 Teoria de Escalonamento

2.1.4.1 Representação e Notação

O Problema do escalonamento pode ser entendido como uma função de decisão que distribui os recursos disponíveis pelas operações, durante uma linha de tempo. O objetivo mais comum do escalonamento é minimizar o tempo de execução de um conjunto de tarefas; outros exemplos podem ser minimização de custo, diminuição de possível atraso na comunicação (LEITE, 2017).

Devido ao grau de complexidade do problema de escalonamento, não é possível determinar uma solução ideal porque o tempo gasto para explorar todo o espaço de busca não é razoável. Sendo assim, é preciso métodos que cheguem próximo de uma boa solução com alta eficiência computacional, em vez de algoritmos que realizam buscas exaustivas e com grande custo computacional (CAIXETA, 2018).

A notação de três campos foi introduzida por Graham et al. (1979), cujo formato é $\alpha \mid \beta \mid \gamma$. Esta notação vem sendo utilizada para representar e classificar os problemas de otimização. A seguir é apresentado o detalhamento de cada campo desta notação (RODRIGUES, 2010; AMORIM, 2013; LEVER, 2017; BRUCKER e KNUST, 2006):

a) α é o ambiente de processamento, que possui somente uma entrada:

- **Ambiente com um processador:** há apenas um processador ou máquina disponíveis para todas as tarefas, quando se há esse valor a notação utilizada é 1;
- **Ambiente com múltiplos processadores ou máquinas idênticas:** existem *P* processadores paralelos idênticos disponíveis, onde *P* é um parâmetro de entrada de problema, e a anotação utilizada é P_m caso o número de processadores seja fixo;
- **Ambiente com máquinas de velocidades diferentes:** a notação usada é *Q* e Q_m para máquinas de processadores de velocidades diferentes;
- **Ambiente com máquinas paralelas não-relacionadas:** para casos onde se há m máquinas paralelas onde seu desempenho varia de acordo com a tarefa a ser processada. A notação usada é *R* e R_m;
- **Ambiente *Flow shop:*** quando se tem máquinas em sequência, e cada tarefa tem que ser executada por cada máquina na mesma ordem de disposição das máquinas, primeiro a máquina 1, depois a máquina 2 e assim sucessivamente. A notação usada é *F* e F_m;
- **Ambiente *Job shop:*** quando a ordem das tarefas é aleatória e as máquinas são de configurações diferente. A notação usada é *J* e J_m;
- **Ambiente *Open shop:*** as sequências são diferentes para cada tarefa, e essa por sua vez é executada apensas uma vez em determinada sequência. A notação utilizada é *O* e O_m.

b) β é o conjunto de características das tarefas e suas restrições (se houver):

- **Data de chegada ou de disponibilidade (*release date*):** indica o instante que a tarefa pode começar a ser executada. A notação utilizada é r_j;
- **Data de término sugerida (*due date*):** indica em que momento a tarefa tem que ser finalizada. A notação utilizada é d_j;
- **Data de término obrigatória (*deadline*):** indica que há tarefas que possuem tarefas de prazos de términos obrigatórios. A notação é D_j;
- **Preempção (*preemption*):** indica se uma tarefa pode ser interrompida e retomada posteriormente. A notação é *ptmn*;

- **Restrição de precedência (*precedence constraints*):** indica se há tarefas que precisam ser finalizadas antes do começo da execução de outras tarefas. A notação utilizada é *prec*;
- **Tempo de processamento (*processing time*):** indica se há um tempo de processamento para cada tarefa já definido. A notação é p_j;

c) γ é a função objetivo a ser otimizada (critérios de otimização), os critérios usados pelas funções nessa classe de problemas são:

- **Maior tempo de conclusão (*makespan*)**: [Notação: C_{max}] refere-se ao tempo de completude da última tarefa terminado sua execução, ou de outra forma, tempo de completude total dentre todas as tarefas executadas. $C_{max} = max\ \{C_1, ..., C_m\}$. O objetivo é minimizar $C_{max,}$ pois uma *makespan* mínimo geralmente significa uma maior utilização da máquina;
- **Tempo total de conclusão (*total completion time*)**: [Notação: ΣC_j ou $\Sigma w_j C_j$] soma de todos os tempos de completude das tarefas (ponderadas ou não), onde deseja-se minimizar ΣC_j ou $\Sigma w_j C_j$;
- **Máxima latência (*maximun lateness*):** [Notação: L_{max}] é definida como $L_j = C_j - d_j$, que será um valor positivo dependendo se a tarefa J_j for executada com atraso, e será um valor negativo se a tarefa for antecipada ou, senão, será um valor nulo. A função de máxima latência (L_{max}), então, retorna o valor do maior atraso possível ou menor antecipação se não houver atrasos, considerando todas as tarefas executadas. $L_{max} = max\{L_{1,} ..., L_n\}$, onde deseja-se minimizar o L_{max};
- **Tempo total de atraso (*tardiness*):** [Notação: ΣT_j ou $\Sigma w_j T_j$] refere-se a soma de todos os tempos de atraso das tarefas (ponderadas ou não). Dessa forma, o atraso assume o valor positivo se as tarefas forem atrasadas, caso não seja, o atraso assume o valor 0. Sendo $T_j = max\{L_j, 0\}$, onde deseja-se minimizar ΣT_j, ou $\Sigma w_j T_{j;}$
- **Número de tarefas tardias:** (Notação: ΣU_j ou $\Sigma w_j U_j$) é o número total de tarefas executadas com atraso (ponderadas ou não), onde U_j é uma penalidade unitária caso a tarefa esteja atrasada (e nula caso contrário);
- **Tempo total de antecipação (*earliness*):** [Notação: ΣE_j ou $\Sigma w_j E_j$] trata-se da soma de todas os tempos de antecipação das tarefas (ponderadas ou não). Sendo assim, a antecipação assume um valor positivo se as tarefas forem antecipadas, caso

contrário, o valor de antecipação é 0. Sendo $E_j = max d_j - C_j$, 0, onde deseja-se minimizar ΣE_j ou $\Sigma w_j E_j$.

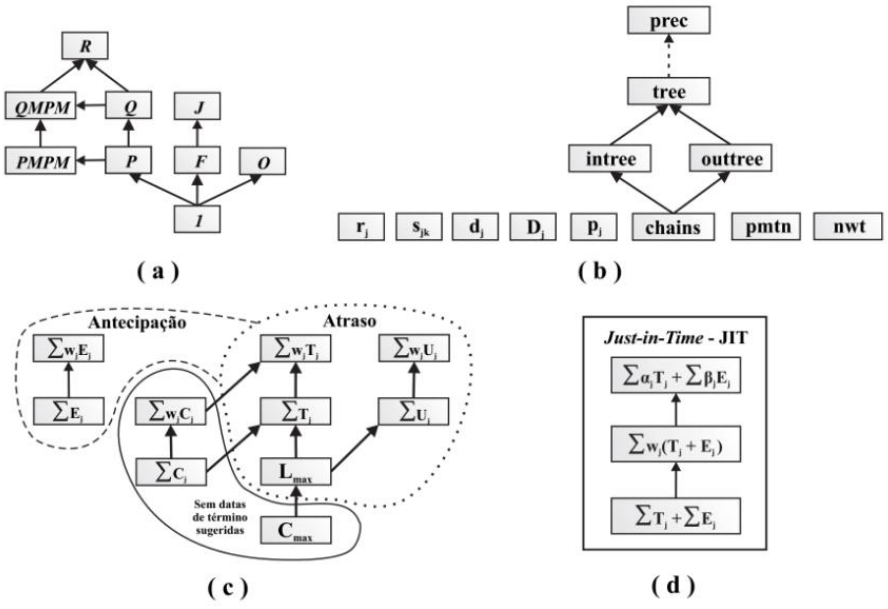
2.1.3.2 Hierarquia de Complexidade

Segundo Amorim (2013) os problemas de escalonamento de tarefas estão classificados dentro de diferentes classes de complexidade computacional. Essa complexidade determina a natureza do algoritmo a ser trabalhado, podendo que um algoritmo possa ser aplicado a diferentes problemas de escalonamento, pois pertencem a casos e generalizações de outros problemas.

Casavant e Kuhl (1988) dividem o escalonamento em duas grandes áreas: local (apenas um processador) e global (vários processadores); escalonamento global por sua vez pode ser dividido em estático ou dinâmico. No estático, o escalonamento é feito antes da execução do processo e sua definição é explícita no código-fonte ou durante sua compilação; já no dinâmico o escalonamento é realizado em tempo de execução e pode ser mudado em qualquer momento durante o processo em execução.

Na Figura 10 (a) são apresentadas as reduções elementares para os principais ambientes de processamento; na Figura 10 (b) são apresentados os diagramas de inclusão para algumas restrições de processamento; na Figura 10 (c) são apresentadas as reduções elementares para as funções objetivo; e na Figura 10 (d) é apresentada a relação de complexidade entre as funções

Figura 10 - Hierarquias de complexidade dos problemas de escalonamento



Fonte: Amorim (2013).

2.1.5 Problema do Escalonamento de Tarefas com Restrições de Recursos.

Fernandes et al. (2016) fala sobre o PETRR, como um problema clássico da literatura, em que consiste em sequenciar as atividades do projeto, respeitando a ordem de execução e precedência dessas tarefas que não podem ser paradas antes de seu término. A literatura aponta que é complexo obter soluções viáveis para esse tipo de problema, logo várias abordagens baseadas em heurísticas têm sido propostas para solucionar o PETRR.

Quando se fala em desenvolver um projeto de software, o processo de escalonar tarefas consiste em determinar a melhor ordem para a execução dentre um conjunto de tarefas com duração pré-estabelecidas, de acordo com suas tarefas antecessoras. Tudo isso na maioria dos casos, com o objetivo de minimizar o tempo total de execução de todas as tarefas, conhecido como ***makespan*** (ROCHA, 2011).

Azevedo (2017) dá a seguinte formulação matemática, seja $V = \{0, 1, ..., N, N + 1\}$, o conjunto de atividades que precisam ser escalonadas utilizando o conjunto de recursos $R = \{1, ..., m\}$. Cada tarefa j $\in V$ tem um tempo de execução de p_j e utiliza uma quantidade específica r_{ij} do recurso $i \in R$ a cada instante do seu processamento. O recurso $i \in R$ tem R_i unidades disponíveis durante todo o desenvolvimento do projeto. Seja a variável binária x_{jt}, cujo valor pode ser 1 se a tarefa for terminada no seu tempo de processamento e 0 se não for. A formulação é apresentada a seguir na Figura 11.

Figura 11 - Formulação Matemática sobre o PETRR

$$\min \sum_{t=1}^{T} t.x_{(N+1),t} \qquad \textbf{(a)}$$

$$\text{s.a.} \sum_{t=1}^{T} t.x_{kt} - p_j - \sum_{t=1}^{T} t.x_{jt} \geq 0 \qquad \forall\,(j,k) \in A \qquad \textbf{(b)}$$

$$\sum_{j=1}^{N} \left(r_{ij}. \sum_{u=t}^{t+p_j-1} x_{ju} \right) \leq R_i \qquad \forall i \in \mathcal{R}, 1 \leq t \leq T \qquad \textbf{(c)}$$

$$\sum_{t=1}^{T} x_{jt} = 1 \qquad \forall j \in V \qquad \textbf{(d)}$$

Fonte: Pinedo (2004).

A função objetivo em (a) representa o tempo de término da atividade $N + 1$, que representa o término do projeto, o *makespan*. O conjunto de restrições em (b) garante que todas as restrições de precedência entre as atividades sejam respeitadas. As restrições da inequação (c) certifica que a demanda total por cada recurso no instante *t* não exceda a disponibilidade

total do recurso. As restrições em (d) garantem que cada atividade seja processada exatamente uma vez. Essa formulação apresenta variáveis e restrições indexadas pelo tempo, portanto a quantidade de variáveis e restrições será pseudopolinomial (PINEDO, 2004).

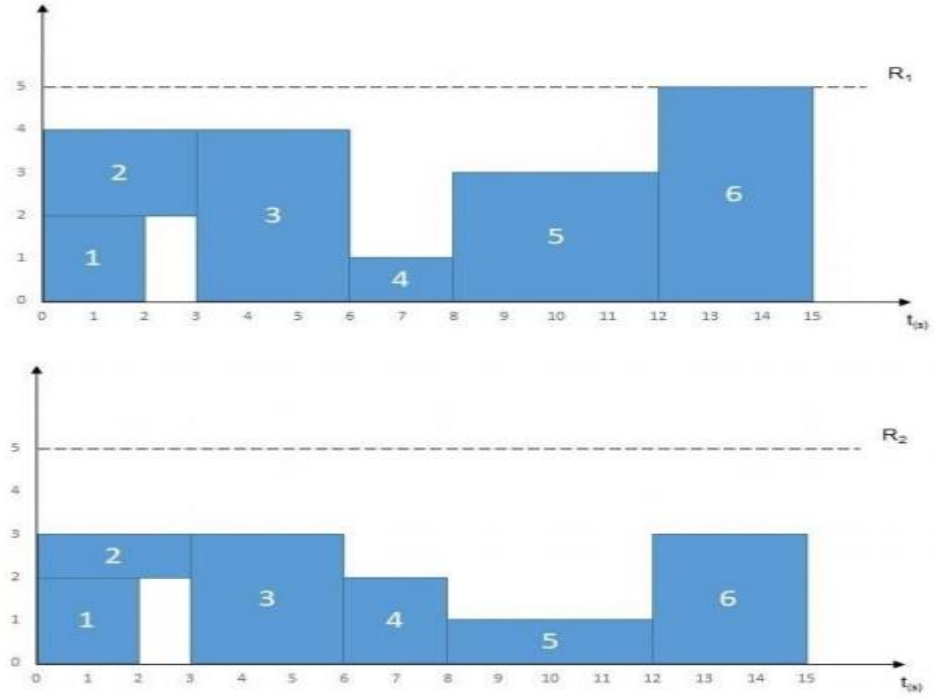
Melo (2018) apresenta um exemplo de instância para o PETRR e duas soluções viáveis (Figura 12) para cada quantidade de consumo de recurso por atividade, apresentada na Tabela 2. Onde coluna **j** é a atividade, a **d** é o tempo de processamento, e **C** é a quantidade de recurso que cada atividade consome e coluna **Predecessor** são as atividades que precisam ser finalizadas para que a atual comece. As atividades vão de 0 a 7, sendo que as atividades 0 e 7 são fictícias, logo, não possuem valor em nenhum campo do exemplo.

Tabela 2 - Exemplo de instância de PETRR

Atividade j	d_j	C_{j1}	C_{j2}	Predecessor
0	0	0	0	-
1	2	2	2	-
2	3	2	1	-
3	3	4	3	-
4	2	1	2	1,2
5	4	3	1	3
6	3	5	3	4
7	0	0	0	0

Fonte: Melo (2018).

Figura 12 - Soluções viáveis para o exemplo



Fonte: Melo (2018).

2.1.3.2 Tarefas

Uma tarefa é qualquer parte necessária para a execução de um projeto por completo. O tamanho da tarefa pode variar dependendo da maneira que o projeto é dividido pelos seus responsáveis; as tarefas variam também do número de participantes para sua execução, podendo ser um da equipe ou todos. E a tarefa também pode usar um número pequeno ou grande de recursos existentes no projeto (ROCHA, 2011).

Rocha (2011) também inclui outra divisão do tipo de tarefa, que pode ser: Concreta, Marco e de Duração Fixa. A tarefa Concreta é aquela que diminui o seu tempo de execução com o aumento de recursos disponíveis para ela. A tarefa do tipo Marco é uma tarefa que indica o ponto de partida de um evento expressivo no projeto. E a tarefa do tipo Duração Fixa é uma tarefa que sua duração não muda, mesmo adicionando ou removendo recursos, ela se mantém fixa em relação ao seu tempo de conclusão.

2.1.3.2 Recursos Humanos

O PMBOK (2017) define recursos como indivíduos com papéis e responsabilidades atribuídos, que podem trabalhar coletivamente na maioria das vezes, para alcançar um objetivo específico do projeto. Pode-se obter o recurso humano individual, o líder do projeto, que normalmente é o gerente, que fica com a função de delegar os demais recursos humanos a tarefas e atividades específicas. Vale ressaltar que o engajamento de todos os recursos humanos para o objetivo final do projeto é de suma importância para o seu êxito.

Rocha (2011) divide os Recursos Humanos em dois tipos: o empregado e o prestador de serviço. O empregado é assalariado, sendo assim o gasto com empregados fixos, podendo variar apenas a quantidade de horas trabalhadas por eles. Já o prestador de serviços tem seu gasto variante conforme seu tipo de serviço, número de horas trabalhadas e consequentemente sua influência no projeto.

2.1.5.3 Habilidades

O termo habilidade tem como origem a palavra *habilitate*, do latim, que significa saber fazer. É a capacidade do indivíduo de realizar algo, como classificar, montar, calcular, ler, observar e interpretar (CARDOSO et al., 2011). Para Gomes (2003) identificar variáveis, compreender fenômenos, relacionar informações, analisar situações-problema, sintetizar, julgar, correlacionar e manipular são exemplos de habilidades.

Uma pessoa para produzir com um recurso precisa possuir um nível mínimo de intensidade de cada habilidade exigida pela atividade. O nível da habilidade faz parte da função objetivo que pretende formar equipes com os maiores níveis de habilidades e assim formar equipes mais capacitadas e, consequentemente, produzindo um produto de maior qualidade (ROCHA, 2011).

2.2 Trabalhos Relacionados

2.2.1 Rocha (2011)

O trabalho de Rocha (2011) teve como principal objetivo a proposta de uma abordagem híbrida para otimizar uma problemática que envolve dois grandes problemas da área de otimização, que são: Problema de Alocação de Equipes e o Problema de Escalonamento de Tarefas com Restrição de Recursos. A fim de elaborar cronogramas eficientes que minimizasse o tempo e o custo, e que maximizasse a qualidade na formação das equipes nas mais variadas atividades que compõe um projeto.

Rocha (2011) também teve como objetivo implementar um ou mais algoritmos, e com isso desenvolver um aplicativo com interface gráfica que auxilia no cadastro de habilidades, recursos e tarefas, permitindo o ajuste dos parâmetros de entrada dos algoritmos. Indo mais além, permitindo a visualização gráfica e detalhada dos cronogramas gerados pelos algoritmos.

A metodologia utilizada por Rocha (2011) foi uma revisão da literatura, buscando trabalhos relevantes com assuntos similares ao seu trabalho, que poderiam trazer uma comparação a sua linha de pesquisa. O intuito de sua revisão da literatura foi encontrar trabalhos que apresentassem aspectos relativos ao Problema de Elaboração do Cronograma; entre as expressões de busca estavam: “alocação de recursos”, “escalonamento de tarefas”, “experiências profissionais adquiridas”, “tarefas com prioridades diferentes”. O autor em questão não apresentou em quais bases de dados ocorreu as pesquisas.

Os resultados de Rocha (2011) basearam-se em suas duas versões propostas para a modelagem, uma mono-objetiva e outra multiobjetiva. A mono-objetiva utiliza a base da implementação o Algoritmo Genético (GA) e a multiobjetiva ultiliza o NSGA-II (*Non-dominated Sorting Genetic Algorihm II*), que é a provavelmente a metaheurística mais utilizada para otimização de funções multiobjetivo. Quando comparados essas duas estratégias de solução, os resultados do NSGA-II foram bem superiores à solução construída com base na configuração planejada.

A modelagem proposta por Rocha (2011) foi aplicada no projeto Sigecom, um projeto do SERPRO - Serviço Federal de Processamentos de Dados. Na Sigecom, como em grande parte das empresas desenvolvedoras de software, a elaboração do cronograma é efetuada com base no conhecimento do gerente do projeto e nas experiências em projetos anteriores. A solução encontrada pelo GA possui um dos menores tempos, porém com um dos custos monetários mais elevado. Como o GA não se propõe a formar equipes de qualidade, seu desempenho nesse objetivo foi o pior dos algoritmos. O NSGA-II, pela sua própria natureza, gerou resultados bem diversificados, encontrando sempre os melhores objetivos. Conforme a Tabela 3 a seguir:

Tabela 3 - Resultados da proposta

Solução	Tempo	Custo	Qualidade
GA	56,56	7.876,50	77.066,00
NSGA-II	56,47	7.357,50	82.340,00

Fonte: Adaptado de Rocha (2011).

2.2.2 Tavares e Godinho (2013)

O trabalho de Tavares e Godinho (2013) trata do problema de sequenciamento de tarefas em um ambiente de máquina única com a possibilidade de terceirização. Esse trabalho tem o objetivo de minimizar a soma ponderada dos custos totais de terceirização e do somatório dos tempos de finalização de cada tarefa, representado como 1 | *Budget* | $(1 - \delta)\ \Sigma C_j + \delta.OC$, na notação de três campos, tratando-se de um problema NP-Difícil.

Dessa forma, o problema abordado na pesquisa de Tavares e Godinho (2013) pode ser definido como segue: seja J um conjunto de n tarefas a serem alocadas em uma máquina única ou realizadas por meio de terceirização. Cada tarefa é definida por: a) um tempo de processamento p_i; b) um custo de terceirização o_i; e c) um tempo de entrega de terceirização l_i. O término de uma tarefa i é definido pela seguinte equação:

$$C_i = \begin{cases} l_i & se\ i \in O_\pi \\ \sum_{k \in SP_k} pk + p_i & se\ i \in S_\pi \end{cases}$$

$$OC = \sum_{j \in O_\pi} O_j$$

em que:

- O conjunto O_π contém todas as tarefas terceirizadas;
- O conjunto S_π contém as tarefas a serem alocadas na máquina única;

- O conjunto SP_k contém todas as tarefas de S_π sequenciadas antes da tarefa k.

O objetivo é minimizar a soma dos tempos para a finalização de cada tarefa ($\sum C_j$) e do custo total de terceirização (OC). Adicionalmente, existe um limite para o custo total de terceirização, conhecido como $Budget$. A soma de $\sum C_J$ e OC é ponderada por meio de um parâmetro de custo $0 < \delta < 1$. Logo, a função objetivo é $\min z = (1-\delta).\sum_{j=1}^{n} C_j + \delta.OC$, onde existe uma condição de contorno que indica que $OC \leq Budget$ (TAVARES e GODINHO, 2013).

A metodologia usada por Tavares e Godinho (2013) para a resolução do referido problema, foi a proposta de usar um método formado por dois estágios: no primeiro modelo, propôs que as tarefas fossem sequenciais, utilizando a regra SPT (*Shortest Processing Time* - Menor Tempo de Processamento) para que se conseguisse a redução do espaço de busca no grafo gerado, enquanto que, no segundo modelo, foi proposto um algoritmo baseado em ACO (*Ant Colony Optimization* – Otimização por Colônia de Formigas).

O algoritmo proposto por Tavares e Godinho (2013) incorporou ao ACO quatro características específicas do problema estudado, a saber:

- Uma representação em forma de grafo para problemas de *scheduling* que envolvam terceirização, obtida por meio da aplicação do primeiro estágio do método;
- Uma regra de pré-seleção, que garante a viabilidade da solução;
- Uma nova regra de visibilidade específica para o problema;
- Uma estratégia de busca local.

Os resultados obtidos por Tavares e Godinho (2013) mostrado na Tabela 4, onde *Gap* é o resultado do esforço de busca encontrado dividido pelo esforço ótimo para a solução. O método proposto que foi desenvolvido usando a linguagem JAVA e executado em um *Pentium* D 2.80 GHz *dual core* com 2 Gb de memória, e testado em uma plataforma *Ubuntu Linux* com o gerenciador de janelas *Gnome.* Para cada número de tarefas (10, 20, 30, 40, 50, 60, e 70), 20 instâncias diferentes foram geradas, usando os seguintes parâmetros:

- Tempos de processamento em um intervalo [1, 10];
- Custos de terceirização no intervalo [1, 40];
- *Lead times* (tempo de espera) de terceirização em um intervalo [1, 30].

Tabela 4 - Gap médio usando método proposto

n	Sem busca local			Com busca local		
	β = **0 (%)**	β = **2.5 (%)**	β = **5 (%)**	β = **0 (%)**	β = **2.5 (%)**	β = **5 (%)**
10	10,45	0,00	0,00	2,97	2,97	0,92
20	13,96	1,08	2,15	6,01	5,89	7,47
30	18,61	2,24	4,17	4,95	0,91	1,40
40	28,87	4,91	8,81	5,41	1,15	5,16
50	22,81	11,69	15,13	8,16	3,95	4,49
60	28,32	18,12	19,84	11,17	6,91	7,45
70	37,70	23,21	28,49	13,69	9,21	10,42

Fonte: Tavares e Godinho (2013).

Na Tabela 4 acima os parâmetros do ACO foram estabelecidos como constantes, com exceção do parâmetro β = {0, 2.5, 5}, onde β indica a visibilidade do rastro dos feromônios, e no caso β = 0 aponta que a visibilidade não é considerada no processo de geração de soluções. E os demais parâmetros são: a) Níveis de feromônio: entre 15 e 20, iniciando em 20; b) 5 formigas por colônia e c) 10 iterações.

Tavares e Godinho (2013) conclui o trabalho citado afirmando que a estratégia de busca local proposta para o problema 1 | *Budget* | $(1 - \delta)\ \Sigma C_j + \delta.OC$ melhorou significativamente os resultados obtidos na resolução de problemas com 30, 40, 50 60 e 70 tarefas. Percebeu-se que a busca local aumenta o tempo computacional necessário para a execução do algoritmo. Porém, mesmo quando são tratados alguns problemas contendo 70 tarefas, o tempo computacional requerido é viável.

2.2.3 Xiao et al. (2013)

O trabalho de Xiao et al. (2013) fez um estudo sobre os GA e notou que embora tenha muitos algoritmos dessa estratégia, sua implementação em uma busca heurística não é fácil. Sendo assim, seu trabalho teve como objetivo tirar o máximo de proveito de outra estratégia de solução metaheurística, o ACO, para obter no final do trabalho uma ferramenta computacional de otimização eficaz e eficiente para resolver o Problema de Programação em Projetos de Software (*Software Project Scheduling Problem* - SPSP).

O método de solução do SPSP proposto por Xiao et al. (2013) foi a proposição de um algoritmo eficaz baseado no ACO, que é chamado de ACS-SPSP. O SPSP investigado na pesquisa de Xiao et al. (2013) consiste em uma alocação ótima de uma equipe de engenheiros de software para atender a uma ou mais tarefas há um custo de mínimo de salário e tempo de duração do projeto. Este problema é NP-Difícil e possui várias questões complexas de otimização combinatória.

Xiao et al. (2013) representaram a aplicação do ACO para o problema investigado através da construção de um grafo, representando os feromônios e construção da solução. A ideia de representar a solução através de um grafo é de descrever um escalonamento de tarefas real em projetos de software. As simbologias e notações consideradas na modelagem em grafos proposta é apresentado na Tabela 6.

Tabela 5 - Simbologias e Notações utilizadas no modelo SPSP

Símbolos	Definição
$SK = \{s_1, s_2, \dots, s_{\|SK\|}\}$	Conjunto de habilidades necessárias em um projeto de software
$TK = \{t_1, t_2, \dots, t_{\|TK\|}\}$	Conjunto de tarefas incluídas em um projeto de software
$EM = \{e_1, e_2, \dots, e_{\|EM\|}\}$	Conjunto de empregados envolvidos em um projeto de software
$V = TK = \{t_1, t_2, \dots, t_{\|TK\|}\}$	Conjunto de vértices do grafo de precedência de tarefas do projeto de software
$A = \{(t_1, t_2), (t_2, t_3), \dots, (t_m, t_n)\}$	Conjunto e arcos do grafo de precedência de tarefas
$G(V, A)$	Grafo de precedência de tarefas (GPT) para o projeto de software
$t_j^{esforço}$	Carga de trabalho da tarefa t_j, que é expressa em pessoa-mês
$t_j^{habilidades} \subseteq SK$	Conjunto de habilidades da tarefa t_j, que é um subconjunto de SK
$e_i^{habilidades} \subseteq SK$	Conjunto de habilidades do empregado e_i, que é um subconjunto de SK
$e_i^{salário}$	Salário do mês do empregado e_i
e_i^{maxded}	Dedicação máxima do empregado e_i no projeto
$M = (m_{ij})_{E \times T}$	Matriz de solução do SPSP, onde o elemento m_{ij} significa que o empregado b é designado à tarefa t_j
$t_j^{início}$	Tempo de início da tarefa t_j
t_j^{fim}	Tempo de término da tarefa t_j
t_j^{custo}	Custo da tarefa t_j
t_j^{dur}	Duração da tarefa t_j
p_{dur}	Tempo de duração do projeto de software
p_{cost}	Custo do projeto de software
$e_i^{excesso}$	Excesso de trabalho do empregado e_i
$p_{excesso}$	Total de excesso de trabalho do projeto de software

Fonte: Xiao et al. (2013).

Neste contexto, Xiao et al. (2013) apresentam alguns exemplos de representação da modelagem proposta, o primeiro deles é apresentado na Figura 13, que apresenta um exemplo das habilidades necessárias em um projeto de software. A Figura 14 apresenta um exemplo sobre um projeto de software, incluindo o significado de cada tarefa, esforço pessoa-mês e o conjunto de habilidades requeridas para cada tarefa. Assim, a relação de precedência entre as tarefas é descrita no grafo da Figura 14, que é formalizado como $GPT = G(V, A)$. Os parâmetros, definições e valores usados no ACS-SPSP por Xiao et al. (2013) são apresentados abaixo na Tabela 7.

Figura 13 - Significado de algumas habilidades de software

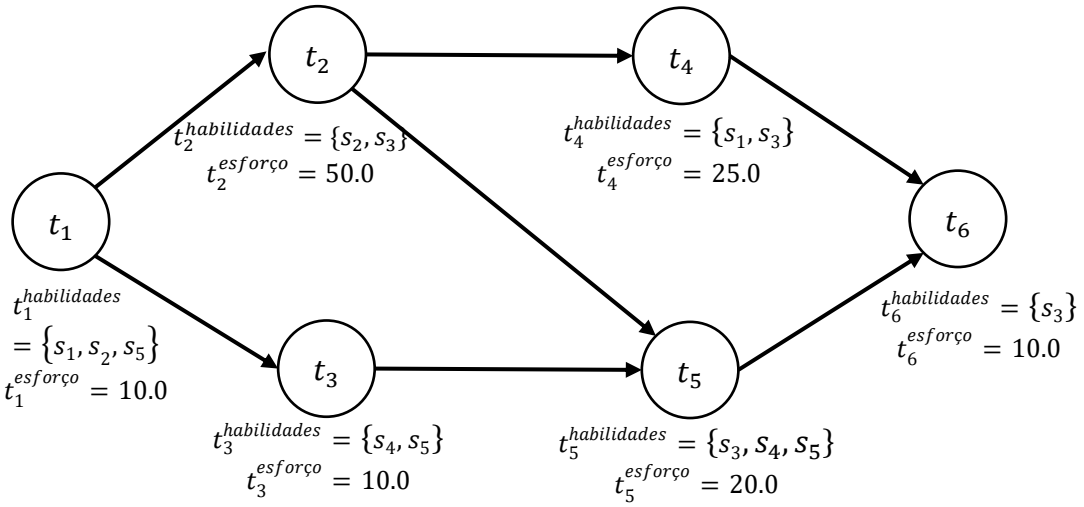
$SK = \{s_1, s_2, s_3, s_4, s_5\}$

s_1: designer de interface
s_2: profissional de UML
s_3: programador
s_4: testador
s_5: líder da equipe

Fonte: Adaptado de Xiao et al. (2013)

Figura 14 - Grafo de precedência de tarefas em um projeto de software

t_1: designer do sistema
t_2: implementação
t_3: testador
t_4: desenvolvedor web
t_5: testa e libera o software
t_6: documentação do sistema



Fonte: Adaptado de Xiao et al. (2013)

Tabela 6 - Parâmetros do ACS-SPSP

Parâmetro	Definição	Valor
N_{gen}	Números da geração	1000
N_{ant}	Número de formigas em uma geração	
β	A taxa de importância do feromônio	1
W_{const}	O peso do custo	10^{-6}
W_{dur}	O peso de duração	10^{-1}
W_{t_const}	O peso do custo da tarefa	10^{-6}
W_{t_dur}	O peso da duração da tarefa	10^{-1}
W_{over}	O peso do excesso de trabalho	10^{-1}
W_{penal}	Constante de penalidade	100
W_{reqsk}	O peso do número de competências necessárias	10
W_{undt}	O peso do número de tarefas não realizadas	10

Fonte: Xiao et al. (2013).

Os resultados obtidos por Xiao et al. (2013) foi que com o aumento da quantidade de tarefas não se pode obter uma solução viável se o número de tarefas for igual a 30. Dessa forma, se o número de tarefas aumentar, o espaço de busca cresce exponencialmente e fica mais difícil

de aplicar ACO para resolver o problema. O aumento de tarefas indica que se os funcionários irão possuir mais tarefas, além da dificuldade de lidar com horas extras no projeto, levando a queda da taxa de acerto. Logo, o aumento do número de tarefas levará ao crescimento da duração total do tempo de projeto, além do aumento das soluções viáveis. Portanto, a relação dos seguintes fatores: dedicação, salário dos funcionários, dificuldade das tarefas e aumento do custo do projeto também está relacionado ao aumento do número de tarefas. Os resultados da pesquisa de Xiao et al. (2013) estão resumidos na Tabela 7.

Xiao et al. (2013) concluíram a pesquisa comparando-o com a abordagem GA, o método proposto ACS-SPSP é promissor. A Estratégia algorítmica proposta baseada em Colônia de Formigas mostrou-se muito boa em lidar com problemas modelados em grafos.

Tabela 7 - Resultados do ACS-SPSP para diferentes números de tarefas

Instância	Tarefa	Taxa de acerto	Duração	Custo
5e_10t	10	100	21.4369	1,200,000
5e_20t	20	9	58.2710	2,400,000
5e_30t	30	0	-	-
10e_10t	10	97	12.0369	1,200,000
10e_20t	20	79	49.2427	2,400,000
10e_30t	30	0	-	-
15e_10t	10	97	8.1436	120,000
15e_20t	20	32	28.3261	120,000
15e_30t	30	0	-	-

Fonte: Xiao et al. (2013).

2.2.4 Myszkowski et al. (2015)

O trabalho Myszkowski et al. (2015) tem o objetivo de pesquisar a robustez da ACO e seus híbridos com regras de prioridade para resolver o Problema de Escalonamento de Projetos com Restrição de Recursos Multi-Habilidades (*Multi-skills-Resource-Constrained Project Scheduling Problem* - MS-RCPSP). Propondo uma nova abordagem a otimização de colônia de formigas de forma híbrida (HAntCO), o MS-RCPSP é o Problema de Escalonamento de Tarefas com Restrições de Recursos, mas se aprofunda em uma vertente desse problema que é com a nova problemática de *multi-skills*, que é a adição de várias habilidades nos funcionários para desempenhar as tarefas de um projeto.

O método proposto por Myszkowski et al. (2015) é uma abordagem híbrida que une as regras de priorização heurística clássica de programação de projetos aliado a otimização por

colônia de formigas. Além disso, uma nova abordagem para o valor de atualização de feromônio foi proposta com base nas melhores e piores soluções armazenadas por formigas.

Myszkowski et al. (2015) utiliza dois grupos de casos de projeto criados: um contém 100 tarefas e as e outra contém 200 tarefas. Dentro de cada grupo, os casos de projeto são variados por um número de recursos disponíveis e a complexidade da relação de precedência. O número de recursos para instâncias de ambos os grupos foi escolhido de forma a preservar a relação de carga média de recursos e relações de tarefas média constante para instâncias dadas. Assim, para instâncias do projeto com 200 tarefas, o número de possíveis recursos e relações de precedência é duas vezes maior do que para instâncias projeto contendo 100 tarefas. As instâncias utilizadas são apresentadas na Tabela 8, onde se mostram as colunas de Instâncias, Tarefas, Recursos, Relações e Habilidade.

Tabela 8 - Valores das instâncias para o HAntCO

Instâncias	Tarefas	Recursos	Relações	Habilidades
100_20_23_9_D1	100	20	23	9
100_20_22_15	100	20	22	15
100_20_47_9	100	20	47	9
100_20_46_15	100	20	46	15
100_20_65_9	100	20	65	9
100_20_65_15	100	20	65	15
100_10_27_9_D2	100	10	27	9
100_10_26_15	100	10	26	15
100_10_47_9	100	10	47	9
100_10_48_15	100	10	48	15
100_10_64_9	100	10	64	9
100_10_65_15	100	10	65	15
100_5_20_9_D3	100	5	20	9
100_5_20_15	100	5	20	15
100_5_48_9	100	5	48	9
100_5_48_15	100	5	48	15
100_5_64_9	100	5	64	9
100_5_64_15	100	5	64	15
200_40_45_9	200	40	45	9
200_40_45_15	200	40	15	15
200_40_90_9	200	40	90	9
200_40_91_9	200	40	91	9
200_40_130_9_D4	200	40	130	9
200_40_144_15	200	40	144	15
200_20_55_9	200	20	55	9
200_20_54_15	200	20	54	15
200_20_97_9	200	20	97	9
200_20_97_15	200	20	97	15
200_20_150_9_D5	200	20	150	9
200_20_145_15	200	20	145	15

200_10_50_9	200	10	50	9
200_10_50_15	200	10	50	15
200_10_84_9	200	10	84	9
200_10_85_15	200	10	85	15
200_10_135_9_D6	200	10	135	9
200_10_128_15	200	10	128	15

Fonte: Myszkowski et al. (2015).

Os resultados obtidos por Myszkowski et al. (2015) da abordagem HAntCO, em média foram considerados só os modos de otimização de custo. Os valores médios encontrados sofreram variações de desvio padrão que refletem a variabilidade dos resultados obtidos. Também foi contabilizado quantas vezes uma estratégia tornou-se melhor do que outra, também em resultados médios. Para entender os resultados é preciso entender dois métodos de atualizações de feromônios:

- ELITE: estratégia que apenas libera duas formigas elite para deixar rastro de feromônio, uma guarda a melhor solução atual e a outra formiga a melhor solução global;
- DIFF: nesta abordagem, a formiga com a solução pior ou melhor encontradas em uma determinada iteração é selecionado. Atualizando o valor do feromônio pelo o pior permite explorar o espaço de busca e consequentemente, escapar de ótimos locais.

Os resultados foram divididos em dois grupos de otimização, um de custos e outra de duração. Para a otimização de duração, estratégia ELITE, tornou-se melhor em 25 casos (69%), enquanto DIFF tornou-se melhor nas restantes. Para otimização de custos, a estratégia DIFF proporcionou melhores resultados em 14 casos (39%), enquanto que, apenas em um caso, a estratégia ELITE tornou-se melhor. Para os restantes, a média dos resultados obtidos tornou-se o mesmo, isso leva a conclusão de que HAntCO faz uma busca no espaço de soluções de forma muito dirigida, sendo incapaz de explorar outras partes da solução (MYSZKOWSKI et al. 2015).

Myszkowski et al. (2015) finaliza seu trabalho concluindo que após as comparações dos resultados obtidos por HAntCO e ACO com os dados de outros métodos heurísticos simples, como Busca Tabu e algoritmos evolutivos clássicos que são especializados operadores genéticos que tenham sido publicados antes. Os resultados fornecidos foram também suportados pelos testes de significância estatística. Os resultados obtidos levam à conclusão de

que, com base em abordagens ACO adequadas para a solução de MS-RCPSP, pode-se fornecer os melhores resultados de todos os métodos investigados.

2.2.5 Muritiba et al. (2018)

O trabalho de Muritiba et al. (2018) propõe um algoritmo de Reconexão por Caminhos (*Path-Relinking* – PR) que explora o espaço solução para o Problema de escalonamento de projetos com restrição de recursos multi-modo (*Multi-mode Resource-Constrained Project Scheduling Problem* - *MRCPSP*) de uma forma comparativa e eficiente. As principais contribuições deste trabalho são os seguintes: a) a introdução da Atribuição do Modo Compressão (MAC), procedimento que busca um método de inicial para concentrar o esforço da pesquisa para uma região promissora, b) o uso do limite inferior do tempo central integrado ao método de religação como filtro do espaço de pesquisa, c) o cálculo do tempo de processamento limites superiores (P^{max}) para cada atividade como uma forma de reduzir a busca na vizinhança local, d) o esquema de reconexão por caminhos discreto que permite a diversificação da pesquisa.

O método de Muritiba et al. (2018) destaca-se dos métodos de RP em geral, pois não atua como um método de intensificação pós-busca, em vez disso, é a espinha dorsal do espaço de solução, pois nenhum conjunto de soluções elite é usado para iniciar o algoritmo. Dessa forma, as etapas de reconexão das soluções são executadas aleatoriamente, sem utilizar nenhuma estratégia gulosa. Além disso, o algoritmo integra um esquema de busca local, que é guiado pelo custo da solução de referência, dessa forma, o resultado da busca local pode ser tomado como nova solução de referência, caso apresente uma solução com um custo melhor que a solução de referência corrente.

Muritiba et al. (2018) implementou o método PR proposto usando Java no Oracle JDK versão 1.8.0_20, rodando sobre a versão JRE 1.8.0_20-b26 64 bits. O hardware utilizado foi um Dell Optiplex 990 - Intel i7-2600 (3.40 GHz), 4 GB de RAM. Foi resolvido cada instância de referência uma única vez com uma semente aleatória constante. Os procedimentos de solução metaheurísticas são executados nas instâncias do conjunto de dados $J10$, $J20$, $J30$, $Boct50$ e $Boct100$. E o quadro geral das instâncias utilizadas foram baseadas nas instâncias $PSPLIB$ e $Boctor$, que são muito utilizadas na literatura para essa estratégia heurística.

Tabela 9 - Visão geral das instâncias de PSPLIB e Boctor

	PSPLIB	Boctor
Número de subconjuntos	7	2
Número de casos / subconjunto	640	120
Número de casos viáveis / subconjunto		120 (méd.)
549 (méd.)		
Número de atividades	10, 12, 14, 16, 18, 20, 30, 50, 100	
Número de recursos renováveis	2	1, 2, 4
Número de recursos não renováveis	2	0
Número de modos	3	1, 2, 4

Fonte: Muritiba et al. (2018).

Muritiba et al. (2018) teve para cada conjunto de instâncias, a coluna **#Act** que mostra o número de atividades, **#Feas** aponta o número de instâncias possíveis, **#Opt** aponta o número de soluções ótimas encontradas e a coluna **%Dev** que mostra o desvio médio entre as soluções ótimas. O número de horários gerados foi (1.000, 5.000 e 50.000). O critério de parada utilizado é o limite de tempo de CPU, que foi de 0.15 segundos por atividade, por exemplo, casos em conjunto $j30$ obteve (30 × 0,15) = 4,5 segundos como limite de tempo de CPU. Os resultados de Muritiba et al. (2018) estão apresentados na Tabela 10.

Tabela 10 - Resultado do PR para instâncias de PSPLIB e Boctor

Dados	**#Act**	**#Feas**	**1.000 horários**		**5.000 horários**		**50.000 horários**		**0.15 s/Act**	
			#Opt	**%Dev**	**#Opt**	**%Dev**	**#Opt**	**%Dev**	**#Opt**	**%Dev**
j10	10	536	532	0.03	534	0.01	535	0.01	535	0.01
j12	12	547	540	0.05	547	0.00	547	0.00	547	0.00
j14	14	551	532	0.14	543	0.05	549	0.01	549	0.01
j16	16	550	532	0.11	544	0.03	548	0.01	549	0.01
j18	18	552	515	0.23	537	0.09	549	0.02	550	0.01
j20	20	554	522	0.19	542	0.06	552	0.01	552	0.01
j30	30	552	-	13.19	-	12.85	-	12.55	-	12.55
Boc50	50	540	-	23.97	-	23.22	-	22.85	-	22.90
Boc100	100	540	-	26.10	-	24.51	-	23.42	-	24.01

Fonte: Muritiba et al. (2018).

Em comparação com o estado de métodos de arte na literatura do MRCPSP, foi apresentado que os resultados do método PR fornecido no trabalho em questão obteve desvio melhor do que os métodos da literatura anteriores dentro quadro computacional semelhante para os dados estabelecidos nos conjuntos de instância $PSPLIB$ e $Boctor$, com a exceção da instância $j30$ (MURITIBA et al. (2018).

2.2.6 Comparativo dos Trabalhos Relacionados com a Proposta

A Tabela 11 apresenta os principais fatores dos Trabalhos Relacionados com o objetivo de mostrar a relação com a proposta do trabalho apresentado:

Tabela 11 - Comparativo dos Trabalhos Relacionados

Trabalhos Relacionados	Comparativo com a Proposta
Rocha (2011)	O trabalho investigou dois grandes problemas da área de otimização, que são: (i) Problema de Alocação de Equipes e o (ii) Problema de Escalonamento de Tarefas com Restrição de Recursos. Onde foi proposto duas estratégias de otimização para os dois problemas, uma multiobjetiva (aplicada ao problema i) e outra mono-objetiva (aplicada ao problema ii), sendo os métodos NSGA-II (*Non-dominated Sorting Genetic Algorihm II*) e Algoritmo Genético (*Genetic Algorithm - GA*), respectivamente, a fim de encontrar boas soluções em um tempo computacional aceitável para os problemas investigados. Este trabalho utilizou a Metodologia de Mapeamento Sistemático para levantar as principais publicações relacionadas ao tema, especificamente Escalonamento de Tarefas com Restrições de Recursos. Também é proposta uma metaheurística baseada em Otimização por Colônia de Formigas, cujo resultados de experimentos computacionais são comparados com literatura.
Tavares e Godinho (2013)	O trabalho de Tavares e Godinho (2013) trata do problema de escalonamento de tarefas em um ambiente de máquina única com possibilidade de terceirização. O problema apresentado busca minimizar a soma ponderada dos custos totais de terceirização e do somatório dos tempos de finalização de cada tarefa, utilizando a estratégia ACO, com algumas características específicas trabalhadas no método usado para atender aos critérios de um Projeto de Software real. Esta pesquisa também trata do problema de escalonamento de tarefas em projetos de software, mas sem a possibilidade de terceirização, e a estratégia ACO utilizada é tratada sem características específicas, cujos resultados são comparados com os benchmarks disponíveis na literatura, não sendo aplicado especificamente em um projeto real.
Xiao et al. (2013)	O trabalho de Xiao et al. (2013) propõe uma abordagem diferente da otimização por colônia de formigas (ACO), que é chamado algoritmo ACS-SPSP. Que trabalha uma tarefa em projetos de software que envolve vários funcionários, onde várias tarefas são distribuídas de acordo com as habilidades dos funcionários. O problema foi modelado em um grafo de precedência de tarefas ($GPT = G(V, A)$), onde seis heurísticas de aprimoramento foram aplicadas a fim de direcionar as formigas a obter as melhores soluções, como: fatores de custos das tarefas, habilidades alocadas dos funcionários e importâncias das tarefas no escalonamento. Este trabalho se assemelha ao de Xiao et al. (2013) em relação a heurística escolhida ACO, assim como o problema de escalonamento investigado, que envolve a divisão das tarefas para os funcionários, mas se diferencia no quesito de habilidades dos funcionários, não havendo a adição dessa problemática chamada de multi-habilidades (multi-skills).
Myszkowski et al. (2015)	O trabalho Myszkowski et al. (2015) tem o objetivo de pesquisar sobre ACO e estratégias híbridas com busca local e regras de prioridade para resolver o problema de MS-RCPSP. Propondo uma nova abordagem a otimização de Colônia de Formigas de forma Híbrida (HAntCO), se aprofundando assim em uma vertente deste problema, chamada de multi-habilidades (multi-skills). Este trabalho não aborda a problemática de multi-habilidades, mas considera a aplicação de uma estratégia híbrida do ACO, aplicado ao escalonamento de tarefas em projetos de software.

Muritiba et al. (2018)	O trabalho de Muritiba et al. (2018) propõe um algoritmo de Reconexão por Caminhos (Path-Relinking – PR) que explora o espaço de busca a fim de se encontrar a melhor solução para o Problema de Escalonamento de Projetos com Restrição de Recursos Multi-Modo (Multi-mode Resource-Constrained Project Scheduling Problem - MRCPSP). O método de Muritiba et al. (2018) destaca-se dos métodos de RP em geral, pois não atua como um método de intensificação pós-busca, em vez disso, é a espinha dorsal do espaço de solução, pois nenhum conjunto de soluções elite é usado para iniciar o algoritmo. Esta pesquisa diferencia-se do de Muritiba et al. (2018) porque, além do Mapeamento Sistemático executado, outra contribuição desta pesquisa é um método de otimização baseado em Colônia de Formigas. Além da comparação dos resultados obtidos com outras instâncias da literatura, diferente de Muritiba et al. (2018).

Fonte: O autor (2019).

3. MAPEAMENTO SISTEMÁTICO SOBRE O ESCALONAMENTO DE TAREFAS COM RESTRIÇÃO DE RECURSOS

Neste capítulo será apresentado todo o desenvolvimento do Mapeamento Sistemático, com as fases de Planejamento, Condução, Resultado do Mapeamento contendo todas as informações necessárias para a fomentação do estudo deste trabalho.

3.1 Planejamento (Protocolo)

Com o intuito de se obter conhecimento sobre o tema proposto, foi planejado um MS, esta técnica foi baseada no guidelines desenvolvido por Kitchenham e Charters (2007), que são elaborados para dar uma visão geral da área a ser investigada e fornecer evidências sobre a pesquisa. Para alcançar os objetivos são definidos 3 etapas (a) Planejamento do Mapeamento: nesse passo, os objetivos da pesquisa são listados e o protocolo do mapeamento é definido; (b) Condução do Mapeamento: durante essa fase, as fontes para o mapeamento são selecionadas, os estudos são identificados, selecionados e avaliados de acordo com os critérios estabelecidos no protocolo do mapeamento e (c) Resultado do Mapeamento: nessa fase, os dados dos estudos são extraídos e sintetizados para serem publicados

3.1.1 Definição do Objetivo e Questões de Pesquisa

O MS foi realizado no período de Setembro/2019 à Novembro/2019 e seu objetivo foi estruturado segundo o paradigma GQM (Goal/Question/Metric) (BASILI et al., 1994), o qual é apresentado na Tabela 12.

Tabela 12 - Aplicação do paradigma GQM

Analisar	Publicações Científicas
Com o propósito de	Levantar os tipos de abordagens de otimização, baseadas em estratégias algorítmicas heurísticas e metaheurísticas existentes na literatura
Com relação	A sua publicação em problemas de escalonamento de tarefas com restrições de recursos aplicados em projetos de software
Do ponto de vista do	Pesquisador
No contexto de	Problemas de otimização combinatória

Fonte: O autor (2019).

- **QP1** – Quais são as abordagens de otimização existentes na literatura que foram aplicadas ao escalonamento de tarefas com restrição de recursos?

- **QP2** – Quais são os principais resultados e instâncias de teste ou benchmarks para experimentos computacionais relatados nas publicações identificadas?

3.1.2 Fontes, Idioma e Expressões de Busca

A busca foi restringida usando-se palavras-chave específicas para encontrar as publicações de interesse. A expressão de busca foi definida de acordo com dois dos quatro aspectos indicados em Petersen et al. (2008): População e Intervenção, conforme a Tabela 13.

Tabela 13 - Expressão de busca utilizada para identificar as publicações

Para investigação por busca manual (no idioma Português):

- **População:** problemas de escalonamento de tarefas com restrições de recursos aplicados a projetos de software (e sinônimos):
  - *Palavras-chave:* "problemas de agendamento de projetos" OU "problemas de cronogramas eficientes para projetos" OU "sequenciamento de tarefas" OU "escalonamento de atividades em projetos de software" OU "otimização para agendamento de projetos" OU "escalonamento de tarefas" OU "escalonamento de tarefas com recursos limitados".
- **Intervenção:** publicações que fazem referências a abordagens de otimização, baseadas em estratégias algorítmicas heurísticas e metaheurísticas e instâncias de teste para experimentos computacionais existentes na literatura (e sinônimos):
  - *Palavras-chave:* "soluções heurísticas e metaheurísticas" OU "estratégias metaheurísticas" OU "estratégias heurísticas" OU "algoritmos de busca local" OU "otimização por colônia de formigas" OU "solução por ACO".

Para investigação por expressão de busca (no idioma Inglês):

- **População:** problemas de escalonamento de tarefas com restrições de recursos aplicados a projetos de software (e sinônimos):
  - *Palavras-chave:* "project scheduling issues" OR "efficient scheduling problems for projects" OR "task sequencing" OR "scheduling of activities in software projects" OR "optimization for project scheduling" OR "scheduling tasks" OR "task scheduling with limited resources".
- **Intervenção:** publicações que fazem referências a abordagens de otimização, baseadas em estratégias algorítmicas heurísticas e metaheurísticas e instâncias de teste para experimentos computacionais existentes na literatura (e sinônimos)
  - *Palavras-chave:* "heuristic and metaheuristic solutions" OR "metaheuristic strategies" OR "heuristic strategies" OR "local search algorithms" OR "ant colony optimization" OR "solution by ACO".

Fonte: O autor (2019).

Os locais de busca definidos para esta pesquisa serão feitos no Google Acadêmico, anais de conferências da Sociedade Brasileira de Pesquisa Operacional (SBPO), que possui relação com o tema a ser pesquisado e bibliotecas digitais, conforme listados abaixo:

1. Anais de conferências
   - SBPO: Simpósio Brasileiro de Pesquisa Operacional
   - ENEGEP: Encontro Nacional de Engenharia de Produção
   - SBES: Simpósio Brasileiro de Engenharia de Software
   - WESB: Workshop de Engenharia de Software Baseada em Busca

2. Bibliotecas digitais

- IEEE *Computer Science Digital Library*: <http://ieeexplore.ieee.org>
- Science Direct: <https://www.sciencedirect.com/>

3.1.3 Critérios de Seleção

Esta pesquisa se restringe à análise de publicações disponíveis até a data presente da execução do estudo. Foram utilizados critérios de inclusão listados a seguir:

1. As publicações devem estar em Português ou Inglês;
2. As publicações devem estar disponíveis na Web ou por meio de contato com os autores;
3. As publicações devem possuir informações sobre a utilização de heurísticas e metaheurísticas que detém relação ou tratam sobre o ACO para problemas de escalonamento de tarefas com restrições de recursos para projetos de software. com datas de término sugeridas;
4. Apresentar uma identificação das instâncias de teste utilizadas nos experimentos computacionais para o problema de escalonamento investigado;
5. As publicações que não satisfaçam tais critérios serão excluídas.

3.1.4 Formulário de Extração dos Dados

Serão extraídas informações de publicações relevantes para a pesquisa, que serão registradas em tabelas, conforme os campos abaixo, descritos da 3.1.4 abaixo.

Tabela 14 - Campos de coleta de dados

A) Dados da publicação	
Título:	Indica o título do trabalho
Autores:	Nome dos autores
Fonte de Publicação:	Local de publicação
Ano da Publicação:	Ano de Publicação
Resumo:	Texto contendo uma descrição do resumo
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Estratégia algorítmicas aplicadas a solução do PETRR.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	Definição do problema investigado e representação na notação de três campos
Instâncias de Teste ou *Benchmarks* utilizados:	Instâncias propostas ou instâncias da literatura utilizadas
Ambiente de testes:	Características da(s) máquina(s) utilizada(s)
Principais resultados e contribuições apresentados:	Principais contribuições, limites superiores e quantitativo de instâncias resolvidas

Fonte: O autor (2020).

3.2 Condução do Mapeamento Sistemático

A execução do Mapeamento Sistemático se deu entre os meses de Março a Abril de 2020, as publicações foram avaliadas e selecionadas de acordo com os critérios estabelecidos na primeira etapa do mapeamento, o protocolo. As publicações duplicadas ou inacessíveis na internet não foram usadas.

Foram encontradas 238 publicações nas fontes de pesquisa entre os anos de 2012 a 2019, e foram realizados os filtros após isso (ver Apêndice A, com as publicações identificadas após o primeiro filtro). A Tabela 15, apresenta a quantidade de dados encontrados em cada etapa.

Tabela 15 - Publicações Encontradas

Fonte	Início	Após o 1º Filtro	Após o 2º Filtro
SBPO	45	15	9
ENEGEP	25	6	3
SBES	10	0	0
IEEE	89	10	1
Science Direct	116	15	7
TOTAL	**238**	**46**	**20**

Fonte: O autor (2020).

Após essa fase, serão apresentados os resultados referentes as questões de pesquisa abordadas no mapeamento. As tabelas com a extração dos dados das publicações selecionadas podem ser consultadas no Apêndice B.

3.3 Análise dos Resultados do Mapeamento Sistemático

3.3.1 Análise e Discussão da QP1

A primeira questão a ser respondida é *"Quais são as abordagens de otimização existentes na literatura que foram aplicadas ao escalonamento de tarefas com restrição de recursos?"*, foram identificados 11 tipos de estratégias para a solução desse problema, conforme pode ser observado na Tabela 16.

Após a análise da Tabela 16 foram identificadas 11 abordagens de otimização utilizadas na literatura para a solução de problemas de escalonamento de tarefas e problemas de escalonamento de tarefas com restrição de recursos. A abordagem mais recorrente encontrada foi a Metaheurística ILS, representando quase 20% do total das recorrências encontradas, seguida de perto pelo Algoritmo Genético com três recorrências, as demais abordagens tiveram duas recorrências, exceto Evolução Diferencial, Otimização por Colônia de Formigas e

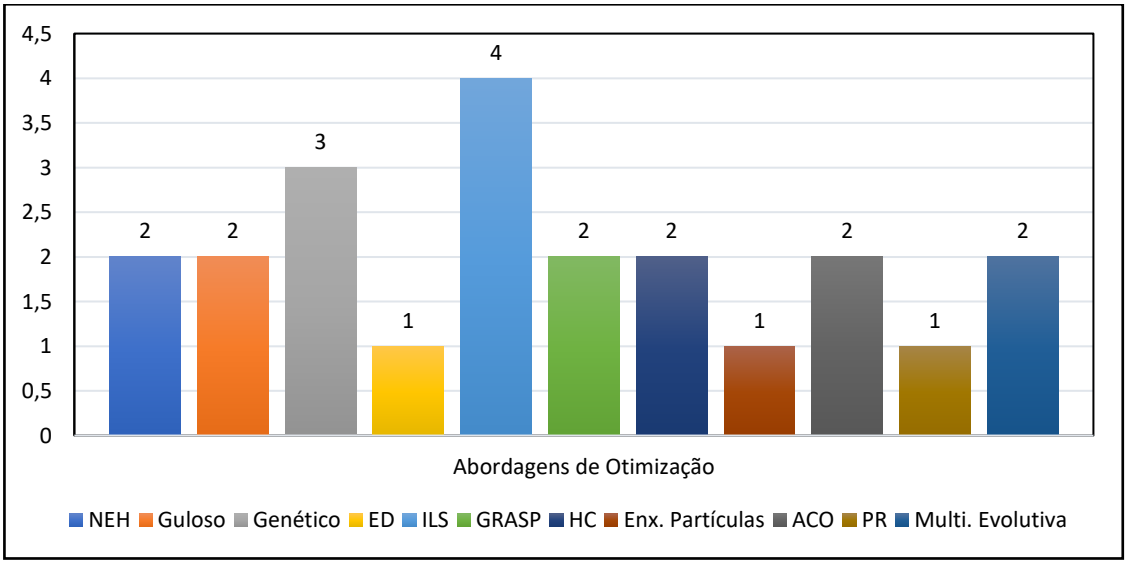
Algoritmo *Path-Relinking* que obtiveram só uma recorrência. O Gráfico 1, apresenta essas quantidades de recorrências.

Tabela 16 - Resultados da QP1

ID	Abordagens de Otimização Identificadas	# de Publicações	Identificação das publicações
1	Heurística de Nawaz, Enscore Jr e Ham (NEH)	2	[P01] e [P06]
2	Algoritmo Guloso	2	[P04] e [P42]
3	Algoritmo Genético	3	[P40], [P06] e [P27]
4	Evolução Diferencial (ED)	1	[P42]
5	Metaheurística ILS	4	[P17], [P21], [P22] e [P31]
6	Metaheurística GRASP	2	[P18] e [P31]
7	Heurística Construtiva	2	[P23] e[P29]
8	Otimização por Enxame de Partículas	1	[P36]
9	Algoritmo de Otimização por Colônia de Formigas (ACO)	2	[P28] e [P46]
10	Algoritmo *Path-Relinking* (PR)	1	[P28]
11	Multitarefa Evolutivo	2	[P14] e [P43]

Fonte: O autor (2020).

Gráfico 1 - Quantidade de Recorrências das Abordagens



Fonte: O autor (2020).

As publicações [P04], [P06], [P28] e [P31], cerca de 25% das publicações encontradas, optaram por resolver seus problemas de escalonamento ou de combinação por soluções híbridas, trabalhando na junção de dois métodos. Essas publicações acreditaram que somando dois métodos, a resposta ao final do trabalho com os resultados encontrados seriam melhor comparados aos da literatura, que abordaram só um método pra solução dos problemas.

A Heurística de NEH, pouco conhecida e utilizada na literatura, se baseia em priorizar as tarefas com maior tempo de processamento em todas as máquinas; o Algoritmo Genético

também parte do mesmo princípio, onde os indivíduos da população inicial são gerados a partir de uma ordenação prévia das *n* tarefas em ordem decrescente do tempo total de processamento, em seguida há uma perturbação nos indivíduos para que haja uma aleatoriedade, formando assim a população a ser trabalhada na busca.

Os métodos Evolução Diferencial e Otimização por Enxame de Partículas poucos mencionados, precisam de uma breve explicação: a Evolução Diferencial segundo Myszkowski (2018) avalia indivíduo por indivíduo de uma população, ao final de cada fase desse processo os melhores indivíduos passam para a próxima geração, e seguindo assim geração após geração, restando apenas um apenas indivíduo que é também o melhor da população. O Enxame por Partículas conforme Jena (2015) trabalha de forma que cada indivíduo (partícula) representa uma solução e também conhece a melhor experiência global encontrada pelo enxame, após a atualização e conhecimento da melhor partícula e sal velocidade, o processo é repetido para todas as dimensões no espaço pesquisado.

A representação da solução das publicações foi bastante diversificada, a publicação [P28] foi dada em uma lista encadeada, já das publicações [P43] e [P46] foi grafos, a da publicação [P42] foi representada em um vetor. As publicações [P01], [P04], [P06] e [P18] escolheram sua representação da solução por matrizes, como está exemplificado na Figura 15, na qual as colunas representam a ordem de (*m*) máquinas e as linhas apresentam as (*n*) tarefas a serem processadas; as demais publicações optaram por usarem listas sequencias de tarefas para representarem suas soluções.

Figura 15 - Representação solução matricial

$$C_{mn} = \begin{vmatrix} C_{11} & C_{12} & \cdots & C_{1i} & \cdots & C_{1n} \\ \cdot & \cdot & \cdot & C_{2i} & \cdots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ C_{k1} & C_{k2} & \cdots & C_{ki} & \cdots & C_{kn} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ C_{m1} & C_{m2} & \cdots & C_{mi} & \cdots & C_{mn} \end{vmatrix}$$

Fonte: O autor (2020).

3.3.2 Análise e Discussão da QP2

A segunda questão a ser respondida é *"Quais são os principais resultados e instâncias de testes ou benchmarks para experimentos computacionais relatados nas publicações*

identificadas?”, a qual resposta se encontra na Tabela 17, com os dados encontrados nas publicações escolhidas.

Tabela 17 - Resultados da QP2

ID	Descrição do Problema	Instâncias Utilizadas (Tarefas)	Principais Resultados
P01	O problema *flowshop* consiste em organizar o processamento de *n* tarefas em um conjunto de *m* máquinas distintas	7,8,10,11, 12,13,14, 20,30,50,75	A variante da NEH que apresentou menor desvio relativo médio, logo o melhor resultado foi a NEHsep
P04	Minimização do tempo de término das tarefas (*makespan*) a serem realizadas por uma máquina de inserção automática de componentes	15,16,17, 20,22	Optar por um caminho com uma lógica simplista (seguir direto para o imediatamente mais próximo), faz o percurso aumentar em 51%
P06	Minimização do tempo total de produção em sistemas produtivos *flow shop* permutacional	20,50,100, 500	A meta-heurística desenvolvida através do algoritmo genético mostrou-se uma ferramenta eficaz para conseguir bons resultados
P14	Investigar a alocação dinamicamente dos recursos computacionais de acordo com as complexidades computacionais das tarefas	Não foi possível encontrar as instâncias no trabalho	O método proposto de Multitarefa Evolutiva mostrou-se superior aos demais algoritmos de última geração em problemas de otimização multitarefa
P17	Neste problema as tarefas são agrupadas em famílias de acordo com características de similaridade e um tempo de preparação é necessário entre o processamento de duas tarefas de famílias distintas	60,80,100	A heurística proposta tem performance superior ao AG com respeito a qualidade das soluções encontradas
P18	O sequenciamento de tarefas *Flowshop* Híbrido e Flexível com tempo de setup dependente da sequência, este problema é estudado em sua formulação dinâmica, ou seja, com eventos de quebra de máquinas	20,50,80, 120	O tempo médio em segundos que o algoritmo gastou para sequenciar as tarefas foi menor que o tempo gasto para sequenciar o problema estático
P21	Problemas de escalonamento da produção em máquinas paralelas com penalidades por antecipação e atraso	40,50,100, 200	A abordagem proposta por Ergun e Orlin (2006) foi adaptada com sucesso e que os ganhos, em termos de tempo de execução foram consideráveis
P22	Problema de sequenciamento de tarefas em máquinas paralelas considerando que as tarefas causam certos desgastes das máquinas. Estes desgastes irão diminuir o desempenho das máquinas aumentando os tempos de	20,35,50	Os testes computacionais mostraram que o ILS apresentou desempenhos superiores em comparação da metaheurística *Simulated Annealing*

	processamento das tarefas ao longo do tempo		
P23	Problema de sequenciamento de tarefas em máquinas paralelas não-relacionadas considerando restrição de precedência entre as tarefas e tempos de preparação dependentes da sequência	6,8,10,12, 15,20,25,30	Foi observado que a heurística obteve resultados significativamente melhores que soluções geradas aleatoriamente e possui um tempo computacional próximo de zero
P27	Problema *job shop* sem espera (PJSSE), cujo objetivo é minimizar o *makespan* entre todas as tarefas	15,20,30,50	A abordagem proposta mostrou desempenho superior em instâncias de grande porte, no que se refere à qualidade das soluções, se comparado com as demais abordagens
P28	Problema de Sequenciamento de Tarefas em Máquinas Paralelas não Relacionadas, onde são considerados os tempos de preparação dependentes da sequência de tarefas, bem como a possibilidade de inclusão de tempo de ociosidade nas máquinas	20,30,50, 60,70,80, 90,100	Os resultados indicam que a abordagem e competitiva quando comparados aos obtidos por um trabalho recente da literatura, que propôs um algoritmo baseado em GRASP e PR
P29	Problema de sequenciamento de tarefas com tempo de processamento distintos em máquinas paralelas idênticas que processam mais de uma tarefa simultaneamente	20,30,40, 60,80,100	Os resultados alcançados comprovaram a eficiência dos métodos heurísticos quando comparados ao limite superior
P31	Problema de escalonamento em *m* máquinas paralelas idênticas com servidor único e tempos de setup dependentes da sequência	6,8,9,10,12, 14,15,20,21, 25,28,30,35, 40,49,50,70, 100	Os resultados obtidos sugerem que os métodos exato e heurístico desenvolvidos foram capazes de encontrar soluções competitivas quando comparadas com as aquelas determinadas por outras abordagens propostas na literatura
P32	Este modelo constitui o problema central entre a classe de recursos com problemas tensos de agendamento de projetos. Basicamente, enquanto minimizamos o *makespan* do projeto, temos que observar a precedência e as restrições de recursos.	Não foi possível encontrar as instâncias no trabalho	Como essa literatura clássica foi um comparativo de vários autores sobre o problema, os principais resultados se divergem entre os mais diversos autores.
P36	O agendamento de tarefas é um problema que pretende reduzir o consumo de energia e melhorar o lucro dos provedores de serviços, reduzindo o tempo de processamento.	180-360	Os resultados experimentais ilustraram que os métodos propostos superaram os métodos comparados existentes na literatura.
P38	O RCPSP pode ser definido como segue. Um conjunto de atividades *N*, numerado de um nó inicial fictício 0 a um nó final fictício *n* + 1, deve ser agendado sem preempção em um conjunto *R* de recursos renováveis. O objetivo do tipo de problema é encontrar um cronograma	10,12,14,16, 18,20,30,60, 90,120	Resultados computacionais em dois conjuntos de dados bem conhecidos da literatura mostram que o algoritmo pode competir com os algoritmos multimodo da literatura quando nenhuma restrição lógica é levada em consideração. Quando as restrições lógicas são levadas em consideração, o algoritmo pode relatar

	viável dentro do menor *makespan* possível do projeto		grandes reduções no *makespan* do projeto para a maioria das instâncias dentro de um tempo razoável
P40	Os problemas de agendamento de projetos com restrição de recursos, nos quais a duração total do projeto, frequentemente referida como o *makespan* do projeto, é minimizada. O *makespan* é a duração total da programação (ou seja, quando todas as atividades têm fim terminado), e deve ser minimizado dentro das relações de precedência entre as atividades do projeto e a disponibilidade limitada de recursos renováveis	10,12,14,16, 18,20,30,60, 90,120	Como resultado obtivemos um novo conjunto de dados com 4 subconjuntos de instâncias de recursos gerados sob um design controlado; e a proposta de uma nova ferramenta fácil para fazer upload de novos conjuntos de dados e novas soluções ou para fazer download das soluções mais conhecidas existentes, que estimulará os pesquisadores a se concentrarem nas instâncias de dados do projeto que podem ser melhoradas
P42	Todas as tarefas no RCPSP não são preventivas e têm tempo de duração (como dados de entrada). Além disso, no cronograma (como solução), cada tarefa tem horário de início e término	100, 200	Os resultados mostraram o domínio do método proposto, onde o método ganhou o menor cronograma mais conhecido para instâncias
P43	Otimização da latência da rede, à disseminação do serviço e ao uso dos recursos	100, 200	Os resultados mostraram que o método obteve as mais altas otimizações dos objetivos e a maior diversidade do espaço da solução
P46	O escalonamento de tarefas é um problema típico de otimização de combinação e o problema de alocação de tarefas em sistemas distribuídos com vários processadores refere-se a como usar os recursos do sistema com mais eficiência em um ambiente de computação distribuído para concluir um conjunto limitado de tarefas	200-4000	Os resultados da simulação mostram que o modelo de algoritmo proposto aprimora a capacidade de busca local e melhora a qualidade do problema de agendamento de tarefas, além de ter boa eficácia, estabilidade e adaptabilidade

Fonte: O autor (2020).

Na Tabela 18 é apresentado em quais os ambientes de processamentos que as publicações se propuseram a solucionar seus problemas, nota-se que os ambiente se diversificaram bastante, mas, a concentração dos ambientes se ateve em maior quantidade em processamento em máquina única e *flow shop*. Uma particularidade que vale ressaltar é a escolha das publicações [P21], [P23] e [P28] em ambiente de processamento de máquinas paralelas não-relacionadas, o que é interessante, ao analisar que essa forma de processamento não se atém a uma sequência pré-definida de ordem da seleção das máquinas, e sim só a disponibilidade das máquinas em tempo de produção.

As publicações [P32], [P38] e [P40] são leituras clássicas sobre o RCPSP e também são levantamentos das soluções mais conhecidas sobre este problema, por isso não entram nas tabelas a seguir.

Tabela 18 - Ambientes de Processamento das Publicações

ID	Ambientes de Processamento			
	Máquina única	***Flow Shop***	***Job Shop***	**Máquinas Paralelas Não-Relacionadas**
P01		X		
P04	X			
P06		X		
P14	X			
P17	X			
P18		X		
P21				X
P22		X		
P23				X
P27			X	
P28				X
P29	X			
P31		X		
P36	X			
P42	X			
P43	X			
P46	X			

Fonte: O autor (2020).

Dentre as publicações identificadas, e apresentadas na Tabela 17, surgiu a necessidade de explorar quais as categorias de problemas relacionados ao tempo de execução e finalização das tarefas as publicações se preocuparam a solucionar, conforme pode ser visualizado na Tabela 19. Sendo obrigatório a resolução desses problemas menores para a solução dos problemas gerais de otimização das publicações.

Conforme a Tabela 19, a variabilidade dos problemas das execuções da tarefa dividiu em minimização do *makespan*, antecipação e atraso, e atraso. A maioria das publicações, cerca de 80%, teve como objetivo a minimização do *makespan*, onde o resultado é uma maior rapidez da realização das tarefas para que haja o aumento de produtividade e um ganho de tempo nos seus tempos de execução.

Tabela 19 - Problemas das Publicações

ID	Problemas		
	Minimização do *Makespan* (*Cmax*)	**Antecipação e Atraso**	**Atraso**
P01	X		
P04	X		
P06	X		
P14	X		
P17			X
P18	X		
P21		X	
P22	X		
P23	X		
P27	X		
P28		X	
P29	X		
P31	X		
P36	X		
P42	X		
P43	X		
P46	X		

Fonte: O autor (2020).

4. ALGORITMO DE OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

Neste capítulo será apresentado a estrutura do algoritmo de Otimização por Colônia de Formigas, com a representação do pseudocódigo e sua implementação.

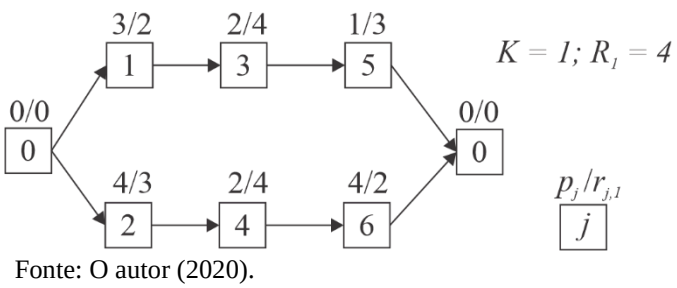
4.1 Definição do Problema investigado

O problema de escalonamento de tarefas com restrição de recursos em projeto de software (PETRR), investigado nesta pesquisa, pode ser exemplificado como segue: um único projeto consiste em um conjunto de $\mathcal{J} = \{0, 1, \dots, n, n+1\}$ de atividades que devem ser processadas. Atividades fictícias 0 e $n+1$ corresponde ao "início do projeto" e ao "fim do projeto", respectivamente. As atividades são interrelacionadas por dois tipos de restrição. Primeiro, as restrições de precedência forçam que a atividade *j* não seja inicializada antes que todas as atividades predecessoras imediatas pertencentes ao conjunto $\mathcal{P}_j$ sejam finalizadas. E segundo, realizar as atividades requer recursos com capacidades limitadas.

Dessa forma, tem-se K tipos de recursos, dados pelo conjunto $\mathcal{K} = \{1, \dots, K\}$. Enquanto está sendo processada, a atividade *j* requer $r_{j,k}$ unidades do tipo de recurso $k \in \mathcal{K}$ durante cada período de seu tempo de processamento não preemptivo p_j. O tipo de recurso k tem uma capacidade limitada de R_k a qualquer instante de tempo. Os parâmetros p_j, $r_{i,k}$ e R_k são determinísticos, e para o início do projeto e as atividades finais tem-se $p_j = 0$ e $r_{j,k} = 0$ para todo $k \in \mathcal{K}$. O objetivo do PETRR é encontrar a precedência e os tempos de completude viáveis para todas as atividades de forma que o *Makespan* do projeto seja minimizado.

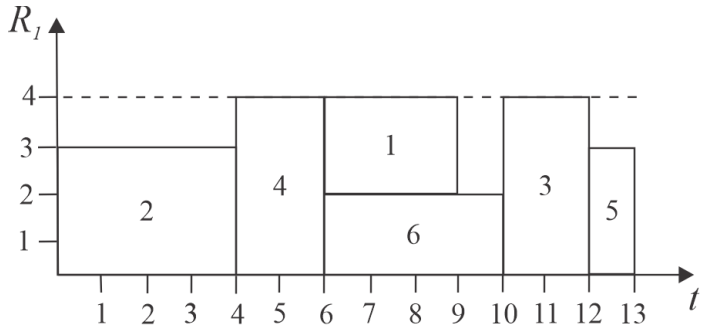
Um exemplo de projeto pode ser consultado na Figura 16, compreendendo $n = 6$ atividades que devem ser escalonadas, sujeitas a $K = 1$ tipo de recurso renovável com capacidade de 04 unidades. Um escalonamento viável com um *Makespan* ótimo de 13 períodos é apresentado na Figura 17.

Figura 16 - Exemplo de instância do problema



Fonte: O autor (2020).

Figura 17 - Escalonamento do exemplo das instâncias



Fonte: O autor (2020).

4.2 Estratégia Algorítmica

Um procedimento de otimização baseado em colônia de formigas é uma meta-heurística baseada em uma população de agentes (formigas) que faz uso de mecanismos de adaptação, cooperação e paralelismo visando a obtenção de um procedimento para resolução de problemas de otimização combinatória (ver Figura 18).

Figura 18 - Pseudocódigodo ACO

```
Pseudo-código:
        Para cada iteração
            Para cada formiga
                job = escolheTrabalho();                          (p1)
                agente = escolheAgente(job);                      (p2)

                Se solução tornou-se infactível
                    Reinicializa a formiga;                       (p3)
                Senão
                    Decai feromônio da aresta job-agente escolhida;   (p4)
                    Se todos os jobs foram alocados
                        Busca Local;                                      (p5)
                        Se a solução corrente é a melhor até agora
                            Decai a qtd de feromônio de todas as arestas;  (p6)
                                Aumenta a qtd de feromônio das
                                        arestas da solução corrente;       (p7)
                            Reinicializa a formiga;                        (p8)
```

Fonte: O autor (2020).

O algoritmo sugerido por Randall (2004), apresentado na Figura 18, possui funções que estão descritas a seguir:

- **p1**: determina qual a tarefa que será alocada, a tarefa que mais consumir recursos será alocada primeiro;
- **p2:** escolhe o agente (formiga) para alocar a tarefa.

Quando o total de recursos disponíveis são consumidos, a preocupação é em obter uma solução factível, mesmo com um custo mais alto. Fazendo com que esse número de recursos disponíveis seja igual ao custo, objetivando a obtenção de uma solução com menor custo, assumindo que a factibilidade não é tão difícil para àquela instância do problema.

- **p3:** se durante a execução do algoritmo, um agente não conseguir uma solução factível, ela é reinicializada. Logo, observa-se que no formigueiro existem formigas de características diferentes, sendo que algumas formigas são pouco alocadas as tarefas, portanto estão distantes de uma solução, enquanto outras formigas estão prestes a completar a solução.
- **p4:** Quando se aloca um agente a uma tarefa, diminui-se a quantidade de feromônio associada àquela aresta assim diminuindo a probabilidade da próxima formiga se ocupar com a mesma tarefa, criando assim, uma maior diversidade das soluções.
- **p5:** melhora o custo total da solução encontrada, decaindo o feromônio de todas as arestas;
- **p6:** aumenta o feromônio das arestas que estão localizadas a solução encontrada;
- **p7:** uma continuação da **p6** para aumentar a probabilidade das outras formigas optarem pelas mesmas arestas da melhor solução encontrada até o momento;
- **p8:** reinicialização da formiga pois seu objetivo já foi cumprido.

4.3 Implementação da Estratégia Algorítmica

A implementação do Algoritmo ACO foi desenvolvida em Python na versão 3.7, o código lê um conjunto de instâncias, onde esse conjunto de instâncias fornece: o número de totais de tarefas; a quantidade de tipo de recursos renováveis; a quantidade máxima em que cada recurso pode disponibilizar para cada execução de uma tarefa, como por exemplo, o projeto possui 12 engenheiros de softwares, 13 programadores, 4 administradores dos bancos de dados e 12 técnicos em suporte e manutenção de computadores; a instância também disponibiliza o quanto cada tarefa vai exigir de cada tipo de recurso; e por último ela fornece quais tarefas são sucessoras de cada tarefa na ordem de execução.

Características importantes do problema que refletiram na implementação do código foi: que uma equipe só pode pegar uma nova tarefa após concluir a tarefa atual; e uma tarefa só pode ser iniciada se suas tarefas antecessoras estiverem concluídas. Essa implementação proporcionou o paralelismo das tarefas, tendo disponibilidade da equipe e tarefas antecessoras finalizadas.

O código implementado possui funções necessárias para sua execução dentro das diretrizes da Otimização por Colônia de Formigas, como a função de obter a posição da formiga, o caminho a ser percorrido por ela, a atualização da melhor solução encontrada até então, a redução e aquisição do valor do feromônio. A Figura 19 e a Figura 20, apresentadas abaixo, são partes do código implementado.

Figura 19 - Código do Algoritmo ACO (a)

```
import random
import os

def le_instancia(arq):
        le_valor = lambda linha, posicao : int(linha[posicao:posicao+8].strip())
        linhas = open(arq).readlines()

        instancia = arq[3:].split('.')[0].replace('_', '.')
        qtd_jobs = le_valor(linhas[0], 0)
        qtd_recursos = le_valor(linhas[0], 8)

        pos = 0
        cap_recursos = []
        for i in range(qtd_recursos):
                cap_recursos.append(le_valor(linhas[1], pos))
                pos += 8
```

Fonte: O autor (2020).

Figura 20 - Código do Algoritmo ACO (b)

```
    def _pega_maquina_formiga(self, iformiga):
        maquinas = self.formigas[iformiga][FORMIGA_MAQUINAS]
        return maquinas.index(min(maquinas))

    def _pega_posicao_formiga(self, iformiga):
        if len(self.formigas[iformiga][FORMIGA_CAMINHO]) == 0:
            return 0
        return self.formigas[iformiga][FORMIGA_CAMINHO][-1]

    def _pega_caminho_formiga(self, iformiga):
        return self.formigas[iformiga][FORMIGA_CAMINHO]

    # alterar aqui para trabalhar com Q||Cmax ou R||Cmax
    def _pega_tempo_job_maquina(self, job, maquina):
        return self.jobs[job]['duracao']

    def _pega_maior_caminho_acumulado_formiga(self, iformiga):
        maquinas = self.formigas[iformiga][FORMIGA_MAQUINAS]
        return max(maquinas)

    def _adiciona_caminho_maquina_formiga(self, iformiga, maquina, caminho):
        p = self._pega_tempo_job_maquina(caminho, maquina)
        self.formigas[iformiga][FORMIGA_MAQUINAS][maquina] += p
        self.formigas[iformiga][FORMIGA_ACUMULADO] = self._pega_maior_caminho_acumulado_formiga(iformiga)

    def _atualiza_recursos_formiga(self, iformiga, caminho):
        for i in range(self.instancia['qtd_recursos']):
            self.formigas[iformiga][FORMIGA_RECURSOS][i] -= self.jobs[caminho]['recursos'][i]

    def _adiciona_trajeto_formiga(self, iformiga, caminho):
        self.formigas[iformiga][FORMIGA_CAMINHO].append(caminho)
        self.delta_caminho[caminho] += self.cresc_feromonio
        maquina = self._pega_maquina_formiga(iformiga)
        self.formigas[iformiga][FORMIGA_ACUMULADO]
        self._atualiza_recursos_formiga(iformiga, caminho)
        self._adiciona_caminho_maquina_formiga(iformiga, maquina, caminho)
```

Fonte: O autor (2020).

5. EXPERIMENTOS COMPUTACIONAIS E RESULTADOS

5.1 Testes Computacionais

Os testes foram feitos em um computador Intel Core i5, CPU (3.4GHz), com 12 GB de RAM, sistema operacional Windows 10.0, 64 bits. O algoritmo ACO foi implementado na linguagem de programação Python e compilados na IDLE Python 3.7 64bits.

5.2 Instâncias do Problema

Os conjuntos de dados utilizados para avaliar o desempenho do algoritmo ACO, foram J30, J60 e J120, extraídos da PSPLIB (*Project Scheduling Problem Library)* baixados de http://www.om-db.wi.tum.de/psplib/, onde foram produzidas por Kolish e Sprecher (1996), tais instâncias são de modo único, devido a implementação do algoritmo ser função objetivo.

5.3 Resultados

Os resultados obtidos pelo algoritmo ACO são comparados com os ótimos resultados disponíveis na literatura sobre o PETRR presente no trabalho de Vanhoucke e Coelho (2016). É importante destacar que o algoritmo aqui proposto é totalmente novo na literatura, produzido e testado com a proposta de encontrar solução do problema de PETRR mais eficaz do que as presentes na literatura.

Entendendo mais um pouco sobre as restrições usadas em Vanhoucke e Coelho (2016), elas são restrições lógicas, onde cada atividade é duplicada em várias atividades de acordo com seu número de modos possíveis. Com o objetivo de construir cronogramas ideais para o PETRR foram usadas restrições OR e BI; a restrição OR é relativamente fácil, ela consiste na duplicação de cada tarefa em várias tarefas, onde essas tarefas duplicadas tem a mesma duração e demanda de recursos; já a restrição BI, são restrições bidirecionais com a duplicação de tarefas em várias tarefas, mas com a adição de atividades extras entre essas atividades duplicadas recém-criadas.

Foram rodas os conjuntos de instâncias J30, J60 e J120, sendo extraído o melhor resultado, com menor tempo de *Makespan*, o tempo médio de *Makespan* e o resultado maior encontrado, logo o pior de cada conjunto de instâncias. O resumo destes resultados está apresentado na Tabela 20 a seguir.

Tabela 20 – Resumo dos Resultados comparados com a literatura

Instâncias	*Makespan*	*Vanhoucke e Coleho (2016)*		*Algoritmo ACO*
		OR	**BI**	
J30	**Min.**	32	32	28
	Méd.	35,1	39,8	39,8
	Máx.	38	44	74
J60	**Min**	62	66	33
	Méd.	67,4	76.9	49,1
	Máx.	72	86	85
J120	**Min.**	124	130	50
	Méd.	132,2	151,6	70,1
	Máx.	140	166	112

Fonte: O autor (2020)

Os resultados presentes na Tabela 20 demonstram que os tempos de *Makespan* do algoritmo ACO proposto são positivamente expressivos, principalmente nos dois conjuntos maiores de instâncias: J60 e J120, apresentando na comparação de menor *Makespan* na instância J120 um resultado menor que a metade que as dois valores encontrados em Vanhoucke e Coelho (2016), mostrando ser um algoritmo altamente competitivo com os existentes na literatura.

Foi perceptível em tempo de teste que algumas sub-instâncias no conjunto de instâncias J30 deram um resultado do *Makespan* muito maior que a média, que está representado no *Makespan* máximo do Algoritmo ACO, demonstrando que o algoritmo teve algumas dificuldades nessas instâncias específicas, mas no geral se mostrou bem melhor aos resultados comparados. Os experimentos computacionais detalhados podem ser consultados no Apêndice C.

6. CONCLUSÃO E PERSPECTIVAS FUTURAS

6.1 Considerações Finais

Neste trabalho foi abordado o Problema de Escalonamento de Tarefas com Restrições de Recursos por meio da Otimização por Colônia de Formigas, buscando com o objetivo de minimizar o *Makespan*, e também propôs a solucionar este problema com um algoritmo que atendesse os critérios da otimização e fosse competitivo com os já existentes na literatura.

Esta pesquisa contou com a condução do Mapeamento Sistemático, que foi conduzido em dois meses, em que foram consultadas publicações que atendessem aos critérios de seleção, dentro das bibliotecas e simpósios mais significativos da área de otimização combinatória e pesquisa operacional. Após a primeira busca foram encontradas 238 publicações e, ao final das filtragens, foram reduzidas a 20 publicações que tratavam do tema desta pesquisa, dentro dessas 20 publicações encontram-se as literaturas clássicas sobre o tema de pesquisa, além de estratégias algorítmicas que podem adaptadas ao problema PETRR.

Como resultado do Mapeamento Sistemático foram identificadas 11 abordagens de otimização que podem ser aplicadas ao problema, incluindo Heurística de Nawaz, Enscore Jr e Ham (NEH - pouco conhecido na literatura), Algoritmos Gulosos, Algoritmo Genético, Evolução Diferencial, Metaheurísticas ILS, GRASP, Heurísticas Construtivas, Otimização por Enxame de Partículas, Algoritmo de Otimização por Colônia de Formigas, Reconexão de Caminhos (*Path Relinking*) e Algoritmo Multitarefa Evolutivo. E relação ao quantitativo de instâncias, a literatura cita instâncias geradas pelos autores, incluindo instâncias de tamanho menor, incluindo 7 a 120 tarefas. Mas também são consideradas instâncias de teste de tamanho maior, variando de 200 a 4000 tarefas. A instância de benchmark identificada na literatura, utilizada em muitas pesquisas da área foi a PSPLIB (*Project Scheduling Problem Library*), disponíveis no seguinte endereço: http://www.om-db.wi.tum.de/psplib/.

Esta pesquisa contou com a implementação de uma estratégia algorítmica baseada em Otimização por Colônia de Formigas, na linguagem de programação Python, por ser uma linguagem de alto nível, além de possuir várias funções implementadas. As instâncias da literatura testadas foram tiradas do benchmark da PSPLIB, que são: J30, J60 e J120, elas são de modo único e usadas com frequência na literatura.

Diante do exposto, pode-se concluir que o objetivo geral e específicos desta pesquisa foram atingidos. Os resultados apresentados mostram que a abordagem produzida neste

trabalho foram um sucesso em ganhos, em termos de minimização do *Makespan*, principalmente para as instâncias maiores. Em instâncias menores, em geral, o algoritmo proposto se saiu bem, mas obteve alguns resultados específicos acima da média.

6.2 Limitações

As limitações deste trabalho estão relacionadas a busca das publicações relacionadas ao tema da pesquisa, tendo em vista que nem todos os trabalhos estão disponibilizados de forma gratuita. Outra limitação está relacionada a complexidade do problema, o que demanda um esforço computacional considerável na execução das tarefas.

Para a instância J90, presente no trabalho de Vanhoucke e Coelho (2016), ainda não foi possível a sua execução devido ao formato das instâncias, que necessitam ser estudadas e o código deve ser corretamente adaptado para a leitura dessas instâncias de teste.

6.3 Trabalhos Futuros

Como trabalhos futuros pretende-se testar outros parâmetros do algoritmo, incluindo o quantitativo de formigas, incremento do feromônio, evaporação do feromônio, número de iterações e feromônio inicial. Também pode-se resolver o problema como uma generalização para problemas com múltiplos modos de execução. Além da proposição de outras estratégias algorítmicas baseadas em metaheurísticas e híbridas.

Em relação as variantes do Problema de Escalonamento de Tarefas com Restrições de Recursos, pode-se pensar em outras variações do problema e comparação dos resultados com mais trabalhos relacionados.

REFERÊNCIAS

ALBANEZ, D.; LINO, S.; e SILVA, S. **Otimização por Colônia de Formigas Aplicada ao Sistema Dinâmico com Absorvedor Linear.** Tecnologias em pesquisa: engenharias, p. 333-347, 2017.

AMORIM, R. **Estratégias Algorítmicas Exatas e Híbridas para Problemas de Escalonamento em Máquinas Paralelas com Penalidades de Antecipação e Atraso.** Tese (Doutorado em Informática) – Universidade Federal do Amazonas, 2017.

AMORIM, R. **Um Estudo sobre Formulações Matemáticas e Estratégias Algorítmicas para Problemas de Escalonamento em Máquinas Paralelas com Penalidades de Antecipação e Atraso.** Dissertação (Mestrado em Informática) – Universidade Federal do Amazonas, 2013.

ANDRADE, P.; MARTENS, A. e VANHOUCKE, M. **Using Real Project Schedule Data to Compare Earned Schedule and Earned Duration Management Project Time Forecasting Capabilities.** Elsevier, vol. 99, p. 68-78, 2019.

ANGHINOLFI, D. e PAOLUCCI, M. **A New Ant Colony Optimization Approach for the Single Machine Total Weighted Tardiness Scheduling Problem.** Internacional Journal of Operations Research, vol. 5, n. 1, p. 44-60, 2008.

AZEVEDO, G. **Escalonamento de Projetos com Restrição de Recursos e Precedências Generalizadas: Um Método Exato de Resolução**. Tese (Doutorado em Engenharia de Produção) – Universidade Federal Fluminense, 2017.

BARBOSA, D.; SILLA, C. e KASHIWABARA, A. **Aplicação da Otimização por Colônia de Formigas ao Problema de Múltiplos Caixeiros Viajantes no Atendimento de Ordens de Serviço nas Empresas de Distribuição de Energia Elétrica.** Brazilian Symposium on Information Systems (SBIS), Goiânia, 2015.

BASILI, V.R.; CALDIERA, G. e ROMBACH, H.D. **Goal Question Metric Approach**. Encyclopedia of Software Engineering, vol. 2, p. 528-532, 1994.

BASTOS, R. **O Problema do Caixiero Viajante com Passageiros e Lotação.** Dissertação (Mestrado em Sistemas e Computação) – Universidade Federal do Rio Grande do Norte, 2017.

BECCENERI, J. C. YANASSE, H. H. e SOMA, N. Y. **Um Algoritmo Exato com Ordenamento Parcial para Solução de um Problema de Programação da Produção: Experimentos Computacionais**. Gestão e Produção (UFSCar), vol. 14, p. 353-361, 2007.

BLUM, C. e ROLI, A. **Metaheuristics in Combinatorial Optimization: Overview and Con-Ceptual Comparison**. ACM Computing Surveys, vol. 35, n. 3, p. 268–308, 2003.

BRUCKER, P. e KNUST, S. **Complex Scheduling.** New York: Springer, 2006.

CAIXETA, C. **Implementação de modificações em um Simulador de Escalonamento de Tarefas em Sistemas Distribuídos**. Monografia (Bacharel em Sistemas da Informação) – Universidade Federal de Uberlândia, 2018.

CARDOSO, R. L.; RICCIO, E. L.; MENDONÇA NETO, O. R. e OYADOMARI, J. C. **Entendo e Explorando as Competências do Contador Gerencial: Uma Análise feita pelos Profissionais**. Advances in Scientific and Applied Accounting, vol. 3, n. 3, p. 353-371, 2011.

CASAVANT, T.L. e HUHL, J.G. (1988). **A Taxonomy of Scheduling in Genereal-Purpose Distributed Computing Systems**. IEEE Transactions on Software Engineering, vol. 14, n. 2, p.141-154, 1988.

CHAND, S.; HUYNH, Q.; SINGH, H.; RAY, T. e WAGNER, M. **On the use of Genetic Programming to Evolve Priority Rules for Resource Constrained Project Scheduling Problems.** Elsevier, vol.98, p. 146-163, 2018.

CHAVES, A. **Uma Meta-Heurística Híbrida com Busca por Agrupamentos Aplicada a Problemas de Otimização Combinatória.** Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, 2009.

COLARES, F. **Alocação de Equipes e Desenvolvimento de Cronogramas em Projetos de Software Utilizando Otimização.** Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Minas Gerais, 2010.

COLORNI, A.; DORIGO, M. e MANIEZZO, V. **An Investigation of Some Properties of an "Ant Algorithm"**. Conferência Parallel Problem Solving from Nature 2 (PPSN), Bruxelas, v. 92, p. 509–520, 1992.

CONGRAM, R.; POTTS, C. e VELDE, S. **An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem.** Institute for Operations Research and the Management Sciences (INFORMS), vol. 14, n.1, p. 52-67, 2002.

CRAWFORD, B.; JOHNSON, F.; SOTO, R.; MONFROY, E. e PAREDES, F. **A Max-Min Ant System Algorithm to solve the Software Projects Scheduling Problem.** Expert Systems with Applications, vol. 41, n.15, p. 6634-6645, 2014.

DANTAS, C. **Aplicação de Ferramentas de Gerenciamento de Riscos e Cronograma em Um Curso de Ensino Educacional.** Monografia (Bacharel em Engenharia de Produção) – Universidade Federal Rural do Semi-Árido, 2019.

DORIGO, M. e STÜTZLE, T. **Ant Colony Optimization: Overview and Recent Advances**. Revista International Series in Operations Research & Management Science, p. 227-263, 2019.

DORIGO, M. **Ottimizzazione, Apprendimento Automatico, Ed Algoritmi Basati Su Metafora Naturale**. Tese (Pós Doutorado em Learning, and Natural Algorithms) – Universidade Politecnico di Milano, 1992.

DORIGO, M.; MANIEZZO, V. e COLORNI, A. **The Ant System: Optimization by a Colony of Cooperating Agents**. IEEE Transactions on Systems, Man, and Cybernetics-Part B, vol. 26, n. 1, p. 29-41, 1996.

DRIDI, O.; KRICHEN, S. e GUITOUNI, A. **A Multiobjective Hybrid Ant Colony Optimization Approach Applied to the Assignment And Scheduling Problem.** International Transactions in Operation Research, vol. 21, n. 6, p. 935-953, 2014.

DUARTE, L. **Influências do Guia PMBOK, da Metodologia Prince2 e do Framework Scrum em um Projeto de Desenvolvimento de um Software de Gestão.** Paramétrica, vol. 9, p. 492-508, 2017.

FERNANDES, G.; SOUZA, S. e FRANÇA, M. **Uma Abordagem Baseada em Iterated Local Search para o Problema de Escalonamento em Projetos com Restrição em Recursos e Múltiplos Modos De Execução.** XVIII Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM), Rio de Janeiro, 2016.

FERREIRA, T.; VERGILIO, S. e SOUZA, J. **Engenharia de Software Baseada em Busca e em Preferência uma Visão Geral.** VII Workshop de Engenharia de Software Baseada em Busca (WESB), Maringá, 2016.

FREITAS, F.; MAIA, C.; CAMPOS, G. e SOUZA, J. **Otimização em Teste de Software com Aplicação de Metaheurísticas.** Revista de Sistemas de Informação da FSMA, n. 5, p.3-13, 2010.

GOMES, D. M.. **Competências e Habilidades do Diretor**. Campo Grande, MS: UCDB, 2003.

GRAHAM, R.; LAWLER, E.; LENSTRA, J. e KAN, A. **Optimization and approximation in deterministic sequencing and scheduling: a survey**. Anais de Discrete Mathematics, vol 5, p. 287-326, 1979.

GUIDA, P. e SACCO, G. **A Method for Project Schedule Delay Analysis.** Elsevier, vol. 128, p. 346-157, 2019.

JENA, R. K. **Multi objective task scheduling in cloud environment using nested PSO framework**. Procedia Computer Science, v. 57, p. 1219-1227, 2015.

JUNIOR, A. **Modelagem do Problema de Roteamento no Planejamento do Inventário Florestal.** Tese (Doutorado em Ciências Florestais) – Universidade Federal do Espírito Santo, 2017.

KERZNER, H. **Gerenciamento de Projetos: Uma Abordagem Sistêmica para Planejamento, Programação e Controle.** São Paulo: Edgard Blucher, 2011.

KHALILZADEH, M.; SHAKERI, H.; GHOLAMI, H. e AMINI, L. **A Heuristic Algorithm for Project Scheduling with Fuzzy Parameters.** Elsevier, vol. 121, p. 63-71, 2017.

KITCHENHAM, B. e CHARTERS, S. **Guidelines for Performing Systematic Literature Reviews in Software Engineering**. Citeseer, 2007.

KOLISCH, R. e SPRECHER, A. PSPLIB - **A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program**. European Journal of Operational Research, Volume 96, 1997.

KRAMER, A. e SUBRAMANIAN, A (a). **A Unified Heuristic and an Annotated Bibliography for a Large Class of Earliness-Tardiness Scheduling Problems**. Journal of Scheduling, vol. 22, n. 1, p. 21-57, 2015.

KRAMER, R. e SUBRAMANIAN, A (b). **A matheuristic Approach for the Pollution-Routing Problem.** European Journal of Operational Research, vol. 243, n. 2, p. 523-539, 2015.

LEITE, M. **Abordagens de Otimização para o Planeamento e Escalonamento Integrado de Operações.** Dissertação (Mestrado em Engenharia de Sistemas) – Universidade do Minho, 2017.

LEVER, E. **Casos Especiais Ótimos de Algoritmos Aproximativos para Problemas de Escalonamento com Restrições de Precedência em Processadores Paralelos Idênticos.** Dissertação (Mestrado em Informática) – Universidade Federal do Amazonas, 2017.

LIMA, G. **Uma Abordagem para Seleção Automática de Bugs em Projetos de Software Baseados em Componentes.** Dissertação (Mestrado em Informática) – Universidade Federal da Paraíba, 2018.

LOPES, J. e TOLFO, C. **Teste de Software: Um Mapeamento Sistemático de Trabalhos Publicados do Siepe. 2018**. X Salão Internacional de Ensino, Pesquisa e Extensão (SIEPE), Santa Catarina, 2018.

LOURENÇO, H.; MARTIN, O. e STUTZLE, T. **Iterated Local Search.** Handbook of Metaheuristic, p. 321-353, 2003.

MAIA, D. **Abordagens e Perspectivas para Gestão do Cronograma em Projetos: Análise dos Métodos Tradicionais e Ágeis.** Iberoamerican Journal of Project Management, vol. 9, n. 2, p. 48-70, 2018.

MAXWELL, L. **Algoritmo de Busca Local**. Rio de Janeiro: Rio, 2015.

MELO, L. **Modelos de Programação Linear Inteira para Variantes do Problema de Programação de Projetos com Restrição de Recursos**. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Goiás, 2018.

MIYAZAWA, F. e SOUZA, C. **Introdução à Otimização Combinatória**. Jornadas de Atualização em Informática, 2018.

MURITIBA, A.; RODRIGUES, C. e COSTA, F. **A Path-Relinking Algorithm for the Multi-Mode Resource-Constrained Project Scheduling Problem.** Computers & Operations Research, vol. 92, p. 145-154, 2018.

MYSZKOWSKI, P; MACIEJ, P e SKOWRONSKI, M. **Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem**. Applied Soft Computing, v. 62, p. 1-14, 2018.

MYSZKOWSKI, P.; SKOWRONSKI, M; OLECH, L. e OSLIZIO, K. **Hybrid Ant Colony Optimization in Solving Multi-Skill Resource-Constrained Project Scheduling Problem.** Soft Computing, vol. 19, n. 12, p. 3599-3619, 2015.

NAN, N. e HARTER, D. **Impact of Budget and Schedule Pressure on Software Development Cycle Time and Effort**. IEEE Transactions on Software Engineering, vol. 35, n. 5, p. 624-637, 2009.

NASCIMENTO, R.; COSTA, S. e COUTINHO, I. **Gestão de Tempo: Porque os Projetos Atrasam e como Garantir o Cumprimento de Prazos.** VI Simpósio Internacional de Gestão de Projetos, Inovação e Sustentabilidade (SINGEP), São Paulo, 2017

NETO, A. **Planejamento de Release Baseado em Otimização Interativa Através da Formalização das Preferências do Tomador de Decisão.** Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual do Ceará, 2016.

PEREIRA, T. **Metodologia para Gerenciamento de Projetos de Simulação a Eventos Discretos Baseada no PMBOK: Pesquisa-Ação em uma Empresa de Alta Tecnologia.** Tese (Doutorado em Ciências em Engenharia de Produção) – Universidade Federal de Itajubá, 2017.

PETERSEN, K.; FELDT, R.; MUJTABA, S. e MATTSSON, M**. Systematic Mapping Studies in Software Engineering.** Ease, vol. 8, p. 68-77, 2008.

PINEDO, M. L. **Planning and Scheduling in Manufacturing and Services**. New Iorque: Springer, 2004.

PMI – Project Management Institute. **A Guide to the Project Management Body of Knowledge (PMBOK Guide) (6th ed.).** Newton Square: Project Management Institute, 2017.

PRESSMAN, R. **Software Engineering: A Practitioner's Approach.** London: Palgrave Macmillan, 2005.

RANDALL, M. **Heuristics for Ant Colony Optimisation using the Generalised Assignment Problem**. IEEE, 2004.

REZENDE, A.; SILVA, L.; BRITTO, A. e AMARAL, R. **Software Project Scheduling Problem in the Context of Search-Based Software Engineering: A Systematic Review.** Elsevier, vol. 155, p. 43-56, 2019.

ROCHA, I. **Uma Abordagem Otimizada para o Problema de Alocação de Equipes e Escalonamento de Tarefas para a Obtenção de Cronogramas Eficientes.** Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual do Ceará, 2011.

RODRIGUES, R. **Caracterizações e Algoritmos para Problemas Clássicos de Escalonamento.** Tese (Doutorado em Engenharia de Sistemas e Computação) – Universidade Federal do Rio de Janeiro, 2010.

SANTOS, G. **Engenharia de Software Baseada em Busca para a Otimização Multiobjetivo de Requisitos Utilizando o Algoritmo NSGA-II.** Monografia (Bacharel em Ciência da Computação) – Universidade Federal de Roraima, 2017.

SANTOS, H.; TOFFOLO, T.; CARVALHO, M e SOARES, J. **Modelos e Metodos de Resolução para Problemas de Escalonamento de Projetos.** XLV Simpósio Brasileiro de Pesquisa Operacional (SBPO), Natal, 2013.

SANTOS, I.; MNEZES, D.; COSTA, A. e ALMEIDA, B. **Panorama dos Registros de Software de Gerenciamento de Projetos no Brasil.** Cadernos de Prospecção, vol. 11, p. 420-430, 2018.

SILVA, J. **Aplicação da Meta-Heurística Otimização por Colônia de Formigas ao Problema de Análise de Segurança de Sistemas de Energia Elétrica.** Dissertação (Mestrado em Computação Aplicada) – Universidade do Vale do Rio dos Sinos, 2018.

SILVA, P.; VIEIRA, C. e SILVA, A. **Modelagem e Solução de Problemas de Sequenciamento em Projetos com Restrição de Recursos.** Produto e Produção, vol. 18, n.1, p. 12-24, 2017.

TAVAREZ, R. e GODINHO, F. **Otimização por Colônia de Formigas para o Problema de Sequenciamento de Tarefas em uma Única Máquina com Terceirização Permitida.** Gestão e Produção, vol. 20, n.1, p.78-86, 2013.

TRICHES, J.; KRIPKA, M. e BOSCARDIN, I. **Otimização Aplicada ao Problema de Alocação de Equipes em uma Panificadora.** Exacta, vol. 13, n. 3, p. 377-388, 2015.

VEGA-VELAZQUEZ, M.; GARCIA-NÁJERA, A. e CERVANTES, H. **A Survey on the Software Project Scheduling Problem.** International Journal of Production Economics, vol. 202, p. 145-161, 2018.

VIEIRA, C. **Modelagem e Solução de Problemas de Sequenciamento de Atividades em Projetos com Restrição de Recursos**. Tese (Doutorado em Engenharia Mecânica) – Universidade Federal de Minais Gerais, 2010.

XIAO, J.; AO, X. e TANG, Y. **Solving Software Project Scheduling Problems with Ant Colony Optimization.** Elsevier, vol. 40, n.1, p. 33-46, 2013.

APÊNDICE A – PUBLICAÇÕES ENCONTRADAS APÓS O PRIMEIRO FILTRO

ID	Título	Autores	Ano	Local de Busca
P01	O Problema de Sequenciamento da Produção em um Ambiente Flowshop com Linhas Semiparalelas e Operação e Sincronização Final	A. Guimarães; M. Cardoso; F. Yalaoui.	2015	ENEGEP
P02	Aplicação do Escalonamento de Projetos com Restrições de Recursos e um Único Método de Processamento na Produção de Tubos Flexíveis	J. Ferreira; D. Duarte; L. Lorenzoni; R. Santolin.	2016	ENEGEP
P03	Otimização do Sequenciamento de Produção com Abordagem Justin-Time e Tempos de Setup Dependentes da Sequência em uma Usina Siderúrgica	M. Domingues; S. Santana; R. Bueno; R. Gonçalves; A. Paiva.	2016	ENEGEP
P04	Análise dos Tempos de Setup dependentes da Sequência através da Regra de Liberação e de Programação Dinâmica em uma Empresa do Polo Industrial de Manaus	L. Bentes; W. Rocha; N. Begnini; R. Onety;	2017	ENEGEP
P05	Uma Heurística Híbrida para Problemas Open-Shop de Sequenciamento de Tarefas	A. Freitas; E. Senne.	2017	ENEGEP
P06	Minimização do Tempo Total de Produção em Sistemas Produtivos Flow Shop Permutacional utilizando Otimização Bio-Inspirada	R. Oliveira; E. Souza; C. Santos.	2019	ENEGEP
P07	Multiprocessor Real-Time Systems with Shared Resources: Utilization Bound and Mapping	J. Han; D. Zhu; X. Wu; L. Yang; H. Jin.	2014	IEEE
P08	Variable Dwell Time Task Scheduling for Multifunction Radar	H. Mir; A. Guitouni.	2014	IEEE
P09	Distributed Task Allocation of Multiple Robots: A Control Perspective	L. Jin; S. Li.	2016	IEEE
P10	Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System	D. Zeng; L. Gu; S. Guo; Z. Cheng; S. Yu.	2016	IEEE
P11	Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing	L. Xin; J. Luo; H. Luo.	2018	IEEE
P12	DQN Inspired Joint Computing and Caching Resource Allocation Approach for Software Defined Information-Centric Internet of Things Network	F. Xu; F. Yang; S. Bao; C. Zhao.	2019	IEEE
P13	Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing	H. Alameddine; S. Sharafeddine; S. Sebbah; S. Ayoubi; C. Assi.	2019	IEEE
P14	Evolutionary Multitasking with Dynamic Resource Allocating Strategy	M. Gong; Z. Tang; H. Li; J. Zhang.	2019	IEEE
P15	Optimizing Resources Allocation for Fog Computing-Based Internet of Things Networks	X. Li; Y. Liu; H. Ji; H. Zhang; V. Leung.	2019	IEEE

P16	Task Scheduling for Smart City Applications Based on Multi-server Mobile Edge Computing	Y. Deng; Z. Chen; X. Yao; S. Hassan; J. Wu.	2019	IEEE
P17	Heurística Busca Local Iterada para o Sequenciamento de Tarefas em uma Máquina com Tempos de Preparação Dependentes da Família.	V. Jacob; J. Arroyo; A. Santos.	2013	SBPO
P18	Heurísticas para Minimização do *Makespan* de Problemas Dinâmicos de Sequenciamento ˆ Flowshop Híbrido e Flexível com Tempo de Setup Dependente com Eventos de Quebra de Máquinas	N. Moreira; S. Souza; M. Filho.	2014	SBPO
P19	Rotação de tarefas em linhas de produção com trabalhadores deficientes e estações em paralelo	F. Araújo; A. Costa; C. Miralles.	2014	SBPO
P20	Aplicação da Metaheurística AGC para o Problema de Alocação de Aulas à Salas.	R. Cirino; M. Santos; A. Delbem.	2015	SBPO
P21	Busca Local Eficiente para Problemas de Escalonamento da Produção em Máquinas Paralelas com Penalidades por Antecipação e Atraso	A. Krammer; A. Subramanian.	2015	SBPO
P22	Sequenciamento de Tarefas em Máquinas Paralelas Considerando Desgastes Dependentes da Sequência	V. Santos; J. Arroyo.	2015	SBPO
P23	Sequenciamento de Tarefas em Máquinas Paralelas com Tempos de Preparação e Precedência entre as Tarefas: Modelagem e Heurísticas Construtivas	F. Faêda; J. Arroyo; A. Santos.	2015	SBPO
P24	Uma Estratégia de Busca Local Eficiente para o Problema De Minimização do Atraso Total Resultante do Sequenciamento da Produção em uma Máquina com Tempos de Setup	K. Farias; A. Krammer; A. Subramanian.	2015	SBPO
P25	Heurística Iterated Greedy para o Problema de Sequenciamento de Lotes de Tarefas em Máquinas Paralelas	M. Fidelis; J. Arroyo.	2016	SBPO
P26	Heurísticas para o Problema de Sequenciamento de Lotes de Tarefas em Máquinas Paralelas	M. Fidelis; J. Arroyo.	2016	SBPO
P27	Um Algoritmo Genético Com Inserção Gulosa Para O Problema De Escalonamento De Tarefas Job Shop Sem Espera	F. Paes; A. Pessoa.	2016	SBPO
P28	Uma Abordagem ACO com Reconexão Por Caminhos para o Problema De Escalonamento De Tarefas Em Maquinas Paralelas Não Relacionadas Com Penalização por Adiantamento e Atraso	C. Larcher; S. Soares; L. Gonçalves.	2016	SBPO
P29	Sequenciamento de Tarefas em Máquinas Paralelas de Processamento em Lotes com Entregas	G. Faria; J. Arroyo; A. Santos; T. Nogueira; J. Chagas.	2016	SBPO
P30	Algoritmo Genético de Chaves Aleatórias Viciadas Aplicado ao Escalonamento de Tarefas em Máquinas Flexíveis Paralelas Idênticas com Restrições de Ferramentas	L. Soares; M. Carvalho.	2017	SBPO

P31	Formulação matemática e algoritmo híbrido para o problema de escalonamento em máquinas paralelas idênticas com servidor único de setup	J. Silva; E. Teixeira; A. Subramanian.	2017	SBPO
P32	Resource-constrained project scheduling: Notation, classification, models, and methods	P. Brucker; A. Drexl; R. Mohring; K. Neumann; E. Pesch.	1999	Science Direct
P33	Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds	S. Abrishami; M. Naghibzadeh; D. Epema.	2013	Science Direct
P34	Investigation of optimal resource allocation by heuristic combination rules	I. Stogniy; V. Taratoukhine; K. Kabitzsch; A. Fedotova.	2013	Science Direct
P35	A Robust Optimization Approach to Volunteer Management in Humanitarian Crises	K. Lassiter; A. Khademi; K. Taaffe.	2015	Science Direct
P36	Multi objective Task Scheduling in Cloud Environment Using Nested PSO Framework	R. Jena.	2015	Science Direct
P37	A taxonomy for task allocation problems with temporal and ordering constraints	E. Nunes; M. Manner; H. Mitiche; M. Gini.	2016	Science Direct
P38	An approach using SAT solvers for the RCPSP with logical constraints	M. Vanhoucke; J. Coelho.	2016	Science Direct
P39	Workflow scheduling algorithms for hard-deadline constrained cloud environments	A. Visheratin; M. Melnik; D. Nasonov.	2016	Science Direct
P40	A tool to test and validate algorithms for the resource-constrained project scheduling problem	M. Vanhoucke; J. Coelho.	2018	Science Direct
P41	An Agent-based Simulation System for Multi-Project Scheduling under Uncertainty	W. Soung; D. Kang; J. Zhang.	2018	Science Direct
P42	Hybrid Differential Evolution and Greedy Algorithm (DEGR) for Solving Multi–Skill Resource–Constrained Project Scheduling Problem	P. Myszkowski; L.Laszczyk; M. Skowronski.	2018	Science Direct
P43	Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures	C. Guerrero; I. Lera; C. Juiz.	2019	Science Direct
P44	Load balance-based workflow job scheduling algorithm in distributed cloud	C. Li; J. Tang; T. Ma; X. Yang; Y. Luo.	2019	Science Direct
P45	Multi-worker multi-task selection framework in mobile crowd sourcing	M. Abouolf; R. Mizouni; S. Singh; H. Otrok; A. Ouali.	2019	Science Direct
P46	Task optimization and scheduling of distributed cyber–physical system based on improved ant colony algorithm	N. Yi; J. Xu; L. Yan; L. Huang.	2020	Science Direct

APÊNDICE B – FORMULÁRIOS DE EXTRAÇÃO DE DADOS

Identificador	[P01]
A) Dados da publicação	
Título:	O problema de sequenciamento da produção em um ambiente *flowshop* com linhas semiparalelas e operação de sincronização final
Autores:	Irce Guimarães, Maurício Souza e Farouk Yalaoui.
Fonte de Publicação:	XXXV Encontro Nacional de Engenharia de Produção
Ano da Publicação:	2015
Resumo:	Neste artigo é abordado uma variante do problema de sequenciamento *flowshop* centrado em uma indústria de material eletroeletrônico. A decisão deste problema consiste em obter a programação das tarefas de forma a otimizar o *makespan*. Foram propostos um modelo de programação linear inteira mista e variantes da heurística de NEH.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Três variantes algoritmo NEH considerando: a média dos tempos de operação das tarefas nas máquinas paralelas, o maior dos tempos de operação das tarefas nas máquinas paralelas, e cada semi-linha separadamente incluindo a máquina de sincronização (NEHav, NEHhi e NEHsep)
Variação do(s) Problema(s) a ser(em) resolvidos(s):	O problema clássico de sequenciamento de tarefas em um ambiente de produção *flowshop* consiste em organizar o processamento de n tarefas em um conjunto de m máquinas distintas, configuradas em série. A principal característica deste problema é que as tarefas devem ter a mesma sequência tecnológica sendo que, cada tarefa tem um tempo de operação específico em cada uma das máquinas.
Instâncias de Teste ou *Benchmarks* utilizados:	7,8,10,11, 12,13,14, 20,30,50,75.
Ambiente de testes:	Intel iR Core TM 3.1GHz com 4GB de RAM.
Principais resultados e contribuições apresentados:	– A variante da NEH que apresentou menor desvio relativo médio, logo o melhor resultado foi a NEHsep. – Maior tempo de processamento gasto para uma instância foi de 359,1 segundos. – Todas as instâncias propostas foram resolvidas, totalizando 30 instâncias.

Identificador	[P04]
A) Dados da publicação	
Título:	Análise dos tempos de setup dependentes da sequência através da regra de liberação e de programação dinâmica em uma empresa do polo industrial de Manaus.
Autores:	Luiz Bentes, Wesley Rocha, Nara Begnini e Renata Onety.
Fonte de Publicação:	XXXVII Encontro Nacional de Engenharia de Produção
Ano da Publicação:	2017

Resumo:	Este trabalho analisa o sequenciamento da produção de uma linha de montagem de placas eletrônicas de uma indústria do Polo Industrial de Manaus. A situação-problema apresenta-se como a busca pela minimização do tempo de término das tarefas (*makespan*) a serem realizadas por uma máquina de inserção automática de componentes.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Algoritmo Guloso usando a regra MTS (menor tempo de *setup*).
Variação do(s) Problema(s) a ser(em) resolvidos(s):	$1\|s_{ij}\|C_{max}$ O problema acima descrito é equivalente ao famoso Problema do Caixeiro Viajante (TSP), porém diferentemente do clássico TSP não é necessário retornar à cidade (tarefa) inicial.
Instâncias de Teste ou *Benchmarks* utilizados:	15,16,17, 20,22
Ambiente de testes:	-
Principais resultados e contribuições apresentados:	− Os resultados indicam ser possível minimizar o tempo despendido com *setup* através de um método exato e este é recomendável em detrimento de uma regra como a MTS. − Maior tempo de processamento gasto para uma instância foi de 16,303 segundos. − Todas as instâncias propostas foram resolvidas, porém na instância com 22 tarefas não foi encontrado uma solução ótima.

Identificador	[P06]
A) Dados da publicação	
Título:	Minimização do tempo total de produção em sistemas produtivos *flow shop* permutacional utilizando otimização bio-inspirada
Autores:	Roberta Oliveira, Edmar Souza e Camila Santos.
Fonte de Publicação:	XXXIX Encontro Nacional de Engenharia de Produção
Ano da Publicação:	2019
Resumo:	Visando a necessidade de estratégias no ambiente industrial e a relevância do sequenciamento de ordens e tarefas, para a melhoria de resultados, esse trabalho se motiva ao estudo e desenvolvimento de sistemas computacionais baseados em heurísticas evolutivas, para a programação de tarefas em ambiente *flow shop* permutacional.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Algoritmo NEH e Algoritmo Genético.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	*Flow Shop* Permutacional $F\|\|C_{max}$
Instâncias de Teste ou *Benchmarks* utilizados:	20,50,100, 500
Ambiente de testes:	

Principais resultados e contribuições apresentados:	− A meta-heurística desenvolvida através do algoritmo genético mostrou-se uma ferramenta eficaz para conseguir bons resultados. − Maior tempo médio de processamento gasto para uma instância foi de 417 segundos, que no caso foi feita pelo Algoritmo Genético. − Todas as instâncias propostas foram resolvidas.

Identificador	[P14]
A) Dados da publicação	
Título:	Evolutionary Multitasking with Dynamic Resource Allocating Strategy
Autores:	Maguo Gong, Zedong Li e Jun Zhang.
Fonte de Publicação:	IEEE Transactions on Evolutionary Computation
Ano da Publicação:	2019
Resumo:	A multitarefa evolutiva é uma proposta recentemente paradigma para resolver simultaneamente várias tarefas usando uma única população. Neste estudo, projetamos um novo algoritmo evolutivo multitarefa com uma estratégia de alocação dinâmica de recursos *on-line*.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Multitarefa Evolutiva.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	No campo da cooperação coevolução, o método de alocação de recursos em aloca os recursos de acordo com a contribuição da subpopulação para a tarefa gera
Instâncias de Teste ou *Benchmarks* utilizados:	
Ambiente de testes:	-
Principais resultados e contribuições apresentados:	− Os resultados experimentais demonstram a superioridade do método proposto em comparação com os algoritmos de última geração em problemas de benchmark da otimização multitarefa.

Identificador	[P17]
A) Dados da publicação	
Título:	Heurística busca local iterada para o sequenciamento de tarefas em uma máquina com tempos de preparação dependentes da família
Autores:	Vinícius Jacob, José Arroyo e André Santos.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2013
Resumo:	Este artigo aborda um problema de programação da produção em uma máquina com tempos de preparação (*setup*) dependente da sequência das famílias de tarefas. Neste problema as tarefas são agrupadas em famílias de acordo com características de similaridade e um tempo de preparação é necessário entre o processamento de duas tarefas de famílias distintas
B) Dados derivados do objetivo	

Tipos de Algoritmos Utilizados:	Metaheurística ILS
Variação do(s) Problema(s) a ser(em) resolvidos(s):	$1\|ST_{sd,b}\|\sum T_j$ Neste trabalho é desenvolvido um algoritmo heurístico baseado na metaheurística *Iterated Local Search* (ILS).
Instâncias de Teste ou *Benchmarks* utilizados:	60,80,100
Ambiente de testes:	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz e 48 GB de RAM
Principais resultados e contribuições apresentados:	− Os resultados experimentais demonstram a superioridade do método proposto em comparação com os algoritmos de última geração em problemas de benchmark da otimização multitarefa. − Maior tempo médio de processamento gasto para uma instância foi de 6,24 segundos, que no caso foi ocasionada pela instância 80. − Todas as instâncias propostas foram resolvidas.

Identificador	[P18]
A) Dados da publicação	
Título:	Heurísticas para minimização do *makespan* de problemas dinâmicos de sequenciamento *flowshop* híbrido e flexível com tempo de *setup* dependente com eventos de quebra de maquinas.
Autores:	Neuma Moreira, Sérgio Souza e Moacir França.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2014
Resumo:	Este trabalho trata do problema de sequenciamento de tarefas *Flowshop* Híbrido e Flexível com tempo de *setup* dependente da sequência denominado HFFS-SDST (*Hybrid Flexible Flowshop With Sequence Dependent Setup Time*). O problema HFFS-SDST tratado neste trabalho é estudado em sua formulação dinâmica, ou seja, com eventos de quebra de máquinas
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Metaheurística GRASP.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	O problema HFFS-SDST Dinâmico abordado pertence à classe de Sequenciamento Dinâmico Estocástico, pois considera a quebra de forma inesperada durante a execução do processamento. A função objetivo proposta para resolver o problema foi a minimização do *Makespan*, denominado de C_{max}.
Instâncias de Teste ou *Benchmarks* utilizados:	20,50,80, 120
Ambiente de testes:	Intel (R) Core (TM) i7-3632QM CPU @ 2.2 GHz, Memória RAM de 8 GB
Principais resultados e contribuições apresentados:	− Os resultados obtidos mostram que o tempo médio em segundos que o algoritmo gastou para sequenciar as tarefas faltantes no instante em que ocorreu a

	quebra foi menor que o tempo gasto para sequenciar o problema estático. - Maior tempo médio de processamento gasto para uma instância foi de 188 segundos, que no caso foi ocasionada pela instância 120. - Todas as instâncias propostas foram resolvidas.

Identificador	[P21]
A) Dados da publicação	
Título:	Busca local eficiente para problemas de escalonamento da produção em máquinas paralelas com penalidades por antecipação e atraso.
Autores:	Arthur Krammer e Anand Subramanian.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2015
Resumo:	Este trabalho trata dos problemas de escalonamento da produção em máquinas paralelas com penalidades por antecipação e atraso. O método proposto foi incorporado em uma metaheurística baseada no *Iterated Local Search* e testado em instancias da literatura para diferentes problemas.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Metaheurística ILS.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	Este trabalho estende a metodologia proposta por Ergun e Orlin (2006) para tratar problemas que envolvam múltiplas máquinas paralelas não-relacionadas, com o objetivo de minimizar a soma ponderada dos atrasos e antecipações sem permitir a inserção de tempo ocioso entre as tarefas, denotado por $$R\|\|\sum e_j^1 E_j + w_j T_j$$
Instâncias de Teste ou *Benchmarks* utilizados:	40,50,100, 200
Ambiente de testes:	Intel Core i7 de 3.40 GHz, com 16 GB de memória RAM e com sistema operacional Linux Ubuntu 12.04
Principais resultados e contribuições apresentados:	- Os resultados obtidos pela metaheurística, em termos de tempo de execução, demostram que a generalização do método proposto por Ergun e Orlin (2006) foi realizada com sucesso. - Maior tempo médio de processamento gasto para uma instância foi de 1047 segundos, que no caso foi ocasionada pela instância 200. - Todas as instâncias propostas foram resolvidas.

Identificador	[P22]
A) Dados da publicação	
Título:	Sequenciamento de tarefas em máquinas paralelas considerando desgastes dependentes da sequência
Autores:	Vivían Santos e José Arroyo.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional

Ano da Publicação:	2015
Resumo:	Este trabalho estuda um problema de sequenciamento de tarefas em máquinas paralelas considerando que as tarefas causam certos desgastes das máquinas. Estes desgastes irão diminuir o desempenho das máquinas aumentando os tempos de processamento das tarefas ao longo do tempo. De acordo com o que se conhece sobre o estado da arte do problema, esta é a primeira aplicação da metaheurística ILS para o problema abordado.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Metaheurística ILS.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	De acordo com a classificação de Graham et al. (1979), este problema será denotado $$Rm\|S_{dd}\|C_{max}$$ em que *Rm* significa o ambiente das máquinas não-relacionadas, S_{dd} corresponde às restrições de desgaste dependentes da sequência e C_{max} é relativo ao critério de otimização.
Instâncias de Teste ou *Benchmarks* utilizados:	20,35,50
Ambiente de testes:	Intel Core i7, CPU (4GHz), com 32 GB de RAM, sistema operacional Ubuntu 14.04, 64 bits.
Principais resultados e contribuições apresentados:	− Os testes computacionais mostraram que o ILS apresentou desempenhos superiores em comparação da metaheurística *Simulated Annealing*. − Maior tempo médio de processamento gasto para uma instância foi de 12 segundos, que no caso foi ocasionada pela instância 50. − Todas as instâncias propostas foram resolvidas.

Identificador	[P23]
A) Dados da publicação	
Título:	Sequenciamento de tarefas em maquinas paralelas com tempos de preparação e precedência entre as tarefas: modelagem e heurísticas construtivas
Autores:	Felippe Faêda, José Arroyo e André Santos.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2015
Resumo:	Este trabalho aborda o problema de sequenciamento de tarefas em maquinas paralelas não-relacionadas considerando restrição de precedência entre as tarefas e tempos de preparação dependentes da sequência. Este problema tem como objetivo minimizar o tempo máximo de conclusão do sequenciamento, conhecido como *makespan*.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Heurística Construtiva.

Variação do(s) Problema(s) a ser(em) resolvidos(s):	O problema abordado tem como objetivo encontrar o sequenciamento das n tarefas nas m máquinas a fim de minimizar o tempo máximo de conclusão das tarefas, chamado de *makespan* C_{max}
Instâncias de Teste ou *Benchmarks* utilizados:	6,8,10,12, 15,20,25,30
Ambiente de testes:	Intel Core i7 com 4.00 GHz de *clock*, 32 GB de memória e sistema operacional Windows 8 64 bits.
Principais resultados e contribuições apresentados:	− Foi observado que a heurística obteve resultados significativamente melhores que soluções geradas aleatoriamente e possui um tempo computacional próximo de zero. − Maior tempo médio de processamento gasto para uma instância foi de 961 segundos, que no caso foi ocasionada pela instância 30. − Todas as instâncias propostas foram resolvidas.

Identificador	[P27]
A) Dados da publicação	
Título:	Um algoritmo genético com inserção gulosa para o problema de escalonamento de tarefas *job shop* sem espera
Autores:	Frederico Paes e Artur Pessoa.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2016
Resumo:	Este artigo apresenta um algoritmo genético (GA) para resolver o problema *job shop* sem espera (PJSSE), cujo objetivo é minimizar o *makespan* entre todas as tarefas. O PJSSE é uma extensão do problema *job shop* clássico, onde uma restrição adicional garante que duas operações consecutivas de uma mesma tarefa sejam processadas sem qualquer interrupção.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Algoritmo Genético.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	O PJSSE é um problema de escalonamento de n tarefas, onde cada tarefa *i* é composta de *ni* operações, denotadas por *Oiu* (*u = 1, · · ·, ni*), que devem ser processadas em *m* máquinas. Deve-se notar que a máquina onde a operação *Oiu* será executada é denotada por *Mil.* Um pequeno exemplo com três máquinas (*m* = 3) e quatro tarefas (*n* = 4) tem uma solução viável com makespan *Cmax* = 19.
Instâncias de Teste ou *Benchmarks* utilizados:	15,20,30,50
Ambiente de testes:	Intel Core i7 − 3770, com 3, 4 GHz de CPU e 11, 7 GB de memória RAM em um sistema operacional Linux.
Principais resultados e contribuições apresentados:	− Como resultado, observou-se um desempenho superior da abordagem proposta em instâncias de

	grande porte, no que se refere à qualidade das soluções, se comparado com as demais abordagens. – Todas as instâncias propostas foram resolvidas.

Identificador	[P28]
A) Dados da publicação	
Título:	Uma abordagem ACO com reconexão por caminhos para o problema de escalonamento de tarefas em máquinas paralelas não relacionadas com penalização por adiantamento e atraso.
Autores:	Celio Larcher, Stênio Soares e Luciana Gonçalves.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2016
Resumo:	Neste trabalho e proposta uma heurística híbrida para tratar o Problema de Sequenciamento de Tarefas em Maquinas Paralelas não Relacionadas, onde são considerados os tempos de preparação dependentes da sequência de tarefas, bem como a possibilidade de inclusão de tempo de ociosidade nas máquinas. O objetivo do problema é a minimização das penalizações por antecipação e atraso.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Algoritmo de Otimização por Colônia de Formigas.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	A formulação matemática que segue foi apresentada no trabalho de [Nogueira et al., 2014]. A variável binária x_{ijk} assume valor 1 se a tarefa *j* e processada após a tarefa *i* na máquina *k*,0 caso contrário. A variável C_{ik} armazena o instante de conclusão da tarefa *i* na máquina *k*. Para registrar o adiantamento e atraso das tarefas são utilizadas as variáveis *Ei* e *Ti* , respectivamente, onde $Ei = max(di - Ci, 0)$ e $Ti = max(Ci - di, 0)$
Instâncias de Teste ou *Benchmarks* utilizados:	20,30,50, 60,70,80, 90,100
Ambiente de testes:	Intel Core i5 4210U 1.7 GHz, com 6 GB de memória RAM e sistema *OpenSuse* Linux 13.2.
Principais resultados e contribuições apresentados:	– Os resultados indicam que a abordagem e competitiva quando comparados aos obtidos por um trabalho recente da literatura, que propôs um algoritmo baseado em GRASP e PR. – Maior tempo médio de processamento gasto para uma instância foi de 4,27 segundos, que no caso foi ocasionada pela instância 60. – Todas as instâncias propostas foram resolvidas.

Identificador	[P29]
A) Dados da publicação	
Título:	Sequenciamento de Tarefas em Máquinas Paralelas de Processamento em Lotes com Entrega.
Autores:	Gilson Faria, José Arroyo, André Santos, Thiago Nogueira e Jonatas Chagas.

Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2017
Resumo:	Neste artigo trata-se o problema de sequenciamento de tarefas com tempo de processamento distintos em máquinas paralelas idênticas que processam mais de uma tarefa simultaneamente. As tarefas deverão ser sequenciadas nos veículos capacitados que estão disponíveis para realizar as entregas em momentos específicos.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Heurística Construtiva.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	A formulação de programação linear inteira mista (PLIM) proposta neste trabalho para o problema teve como base a formulação proposta por Cheng et al. [2015] para o problema de sequenciamento em uma única máquina de processamento em lotes com entrega. Variáveis de decisão: • x_{yk}^{l}: variável binária que assume 1 se a tarefa *i* pertence ao lote *k* na máquina l, 0 caso contrário. • y_{it}^{v}: variável binária que assume 1 se a tarefa i é associada ao veículo v pertencente ao intervalo t, 0 caso contrário. • w_{k}^{l}: variável contínua que indica o tempo de processamento do lote k na máquina l. • o_{k}^{l}: variável contínua que indica o momento em que o lote k termina de ser processado na máquina l. • u_{i}: variável contínua que indica o momento em que a tarefa i termina de ser processada. Utilizando essas variáveis é possível descrever a seguinte formulação PLIM: $$\max f = \sum_{i=1}^{n}\sum_{t=1}^{z}\sum_{v=1}^{Vt} R_i \, y_{it}^{v}$$
Instâncias de Teste ou *Benchmarks* utilizados:	20,30,40, 60,80,100
Ambiente de testes:	Opteron (TM) 6272 CPU @ 1400MHz, com 96GB de memória RAM e sistema operacional Ubuntu 14.04.5 LTS, 64 bits.
Principais resultados e contribuições apresentados:	− Os resultados alcançados comprovaram a eficiência dos métodos heurísticos quando comparados ao limite superior. − Todas as instâncias propostas foram resolvidas.

Identificador	[P31]
A) Dados da publicação	
Título:	Formulação matemática e algoritmo híbrido para o problema de escalonamento em máquinas paralelas idênticas com servidor único de *setup*.
Autores:	João Silva, Everton Teixeira e Anand Subramanian.
Fonte de Publicação:	Simpósio Brasileiro de Pesquisa Operacional
Ano da Publicação:	2018
Resumo:	Este artigo aborda o problema de escalonamento em m máquinas paralelas idênticas com servidor único e tempos de setup dependentes da sequência. Nessa variante, cujo objetivo considerado é o de minimizar o tempo máximo de término (*makespan*), tem-se disponível apenas uma máquina, indivíduo ou equipe responsável pelas operações de *setup*.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Metaheurística ILS.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	O modelo matemático proposto, denominado ATIF (*Arc-Time Indexed Formulation*), é baseado na formulação de Silva et al. [2018] originalmente desenvolvida para uma variante do problema $$1\|S_{ij}\|\sum w_j T_j$$
Instâncias de Teste ou *Benchmarks* utilizados:	6,8,9,10,12,14,15,20,21,25,28,30,35,40,49,50,70,100
Ambiente de testes:	Intel Core i7, com 3,4 GHz, 16 GB de memória RAM e sistema operacional Linux Ubuntu 16.04.
Principais resultados e contribuições apresentados:	− Os resultados obtidos sugerem que os métodos exato e heurístico desenvolvidos foram capazes de encontrar soluções competitivas quando comparadas com as aquelas determinadas por outras abordagens propostas na literatura. − Maior tempo médio de processamento gasto para uma instância foi de 15,15 segundos, que no caso foi ocasionada pela instância 30. − Todas as instâncias propostas foram resolvidas.

Identificador	[P32]
A) Dados da publicação	
Título:	Resource-constrained project scheduling: Notation, classi®cation, models, and methods
Autores:	Perter Brucker, Andreas Drexl, Rolf Mohring, Klaus Neumann e Ervin Pesch.
Fonte de Publicação:	ScienceDirect European Journal of Operational Research
Ano da Publicação:	1999
Resumo:	A programação do projeto se preocupa com a produção de item único ou em pequenos lotes, onde recursos escassos devem ser alocados para atividades dependentes ao longo do tempo. Até o momento, não existe nenhum esquema de classificação compatível com o que é comumente aceito na

	programação da máquina. Um dos objetivos de nosso artigo é preencher essa lacuna. O segundo objetivo deste artigo é revisar alguns dos desenvolvimentos recentes.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Como essa literatura clássica foi um comparativo de vários autores sobre o problema, os tipos de algoritmos se divergem entre os mais diversos autores.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	Este modelo constitui o problema central entre a classe de recursos com problemas tensos de agendamento de projetos. Basicamente, enquanto minimizamos o *makespan* do projeto, temos que observar a precedência e as restrições de recursos. $$PS\|prec\|C_{max}$$
Instâncias de Teste ou *Benchmarks* utilizados:	Como essa literatura clássica foi um comparativo de vários autores sobre o problema, as instâncias utilizadas se divergem entre os diversos autores.
Ambiente de testes:	Como essa literatura clássica foi um comparativo de vários autores sobre o problema, os ambientes de testes utilizados se divergem entre os diversos autores.
Principais resultados e contribuições apresentados:	Como essa literatura clássica foi um comparativo de vários autores sobre o problema, os principais resultados se divergem entre os mais diversos autores.

Identificador	[P36]
A) Dados da publicação	
Título:	Multi objective Task Scheduling in Cloud Environment Using Nested PSO Framework.
Autores:	R K Jena
Fonte de Publicação:	ScienceDirect Procedia Computer Science
Ano da Publicação:	2015
Resumo:	O agendamento de tarefas é uma etapa importante para melhorar o desempenho geral da computação em nuvem. O agendamento de tarefas também é essencial para reduzir o consumo de energia e melhorar o lucro dos provedores de serviços, reduzindo o tempo de processamento. Este artigo enfoca o agendamento de tarefas usando uma TSPSO (*Particle Swarm Optimization*) aninhada com vários objetivos para otimizar a energia e o tempo de processamento.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Otimização por Enxame de Partículas.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	No modelo proposto, um aplicativo em nuvem é considerado como uma coleção de tarefas do usuário que realizam um complexo tarefa de computação usando recursos de nuvem.
Instâncias de Teste ou *Benchmarks* utilizados:	180-360
Ambiente de testes:	-

Principais resultados e contribuições apresentados:	− Os resultados experimentais ilustraram que os métodos propostos superaram os métodos comparados existentes na literatura. − Todas as instâncias propostas foram resolvidas.

Identificador	[P38]
A) Dados da publicação	
Título:	An approach using SAT solvers for the RCPSP with logical constraints.
Autores:	Mario Vanhoucke e José Coelho.
Fonte de Publicação:	ScienceDirect European Journal of Operational Research
Ano da Publicação:	2016
Resumo:	Este artigo apresenta uma nova abordagem de solução para resolver o problema de escalonamento de projetos com restrição de recursos na presença de três tipos de restrições lógicas. Um solucionador de satisfações (SAT) é usado para garantir a lógica de precedência original e está embutido em uma busca metaheurística para esquemas viáveis de recursos que respeitem tanto a disponibilidade limitada de recursos renováveis quanto a lógica de precedência.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Otimização por Colônia de Formigas
Variação do(s) Problema(s) a ser(em) resolvidos(s):	O RCPSP pode ser definido como segue. Um conjunto de atividades *N*, numerado de um nó inicial fictício 0 a um nó final fictício $n + 1$, deve ser agendado sem preempção em um conjunto *R* de recursos renováveis.
Instâncias de Teste ou *Benchmarks* utilizados:	
Ambiente de testes:	-
Principais resultados e contribuições apresentados:	− Um primeiro experimento de teste em instâncias MRCPSP sem restrições lógicas mostra que o procedimento é capaz de competir com os melhores procedimentos da literatura. − Um segundo teste em instâncias monomodo com restrições lógicas mostra resultados promissores, pois o tamanho das redes transformadas não aumenta muito, e resultados promissores que podem ser usados para fins de comparação de pesquisas futuras foram gerados em um tempo razoável. − O terceiro experimento usou as instâncias MRCPSP com restrições lógicas e uma tabela semelhante para fins de comparação futura foi exibida. Esses resultados mostram que o tamanho e a complexidade das redes aumentam rapidamente, e, portanto, nem sempre foi possível encontrar soluções aprimoradas (em comparação com as soluções dessas instâncias quando as restrições lógicas são ignoradas).

Identificador	[P40]
A) Dados da publicação	
Título:	A tool to test and validate algorithms for the resource-constrained project scheduling problem.
Autores:	Mario Vanhoucke e José Coelho.
Fonte de Publicação:	ScienceDirect European Journal of Operational Research
Ano da Publicação:	2018
Resumo:	Em um artigo escrito por Vanhoucke et al. (2016), uma visão geral dos artificiais bancos de dados de projetos empíricos e ciais foram fornecidos para gerenciamento e controle integrados de projetos. Esses bancos de dados são coleções das instâncias de dados mais conhecidas e difundidas disponíveis na literatura para a construção de um cronograma de linha de base, a análise de risco de cronograma ou o uso para controle de projeto. O presente artigo serve como um estudo de acompanhamento para uma maior elaboração sobre o uso dessas instâncias de dados e para dar aos pesquisadores um incentivo para usar esses conjuntos de dados para suas pesquisas sobre o desenvolvimento e validação de novos algoritmos para o planejamento de projetos.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Otimização por Colônia de Formigas
Variação do(s) Problema(s) a ser(em) resolvidos(s):	Este artigo se restringe a uma discussão dos dados do projeto que podem ser usados para resolver os conhecidos problemas de agendamento de projetos com restrição de recursos, nos quais a duração total do projeto, frequentemente referida como o *makespan* do projeto, é minimizada
Instâncias de Teste ou *Benchmarks* utilizados:	
Ambiente de testes:	-
Principais resultados e contribuições apresentados:	− Novo conjunto de dados: Um novo conjunto de dados com 4 subconjuntos de instâncias de recursos gerados sob um design controlado, foi gerado e disponibilizado publicamente. − Soluções disponíveis: Uma simples, mas eficiente abordagem de upload para soluções recém-obtidas foi proposta tanto para o RCPSP quanto para o MMRCPSP. − Nova ferramenta: É proposta uma ferramenta fácil para fazer upload de novos conjuntos de dados e novas soluções ou para fazer download das soluções mais conhecidas existentes, que estimulará os pesquisadores a se concentrarem nas instâncias de dados do projeto que podem ser melhoradas.

Identificador	[P42]

A) Dados da publicação	
Título:	Hybrid Differential Evolution and Greedy Algorithm (DEGR) for Solving Multi–Skill Resource–Constrained Project Scheduling Problem.
Autores:	Pawel Myszkowski, Lukasz Laszczyk e Marek Skowronski.
Fonte de Publicação:	ScienceDirect Applied Soft Computing
Ano da Publicação:	2018
Resumo:	O artigo apresenta um híbrido de Evolução Diferencial (DE) e Algoritmo Ganancioso (DEGR) aplicado para resolver o Problema de Programação de Projetos com Restrições de Múltiplas Habilidades. A representação indireta especializada e a transformação do espaço da solução de discreto (típico para esse problema) para contínuo (típico para abordagens de DE) são propostas e examinadas.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Evolução Diferencial e Algoritmo Guloso.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	MS – RCPSP (Problema de Programação de Projetos com Restrições de Múltiplas Habilidades) como uma extensão prática do RCPSP adiciona o domínio de habilidades. Cada tarefa no RCPSP é não-preemptiva e tem tempo de duração (como dados de entrada). Além disso, na programação (como solução) cada tarefa tem hora de início e de término.
Instâncias de Teste ou *Benchmarks* utilizados:	100,200
Ambiente de testes:	-
Principais resultados e contribuições apresentados:	− Os resultados mostraram o domínio do método proposto, onde o método ganhou o menor cronograma mais conhecido para instâncias. − Todas as instâncias propostas foram resolvidas.

Identificador	[P43]
A) Dados da publicação	
Título:	Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures.
Autores:	Carlos Guerreo, Isaac Lera e Carlos Juiz.
Fonte de Publicação:	ScienceDirect Future Generation Computer Systems
Ano da Publicação:	2019
Resumo:	Este estudo compara três algoritmos evolutivos para o problema da colocação de serviços de neblina: algoritmo genético de soma ponderada (WSGA), algoritmo genético de classificação não dominado II (NSGA-II) e algoritmo evolutivo multiobjetivo baseado em decomposição (MOEA / D). Os algoritmos são avaliados com uma topologia de rede Barabasi – Albert aleatória com 100 dispositivos e com dois tamanhos de experimentos de 100 e 200 serviços de aplicativos.
B) Dados derivados do objetivo	

Tipos de Algoritmos Utilizados:	Multitarefa Evolutiva e Algoritmo Genético.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	A computação em nevoeiro é uma arquitetura que distribui as funções de computação e armazenamento de aplicativos tradicionais baseados em nuvem para dispositivos mais próximos dos usuários, ao longo de um continuum de nuvem para coisa. Esses dispositivos com recursos computacionais e de armazenamento, comumente chamados de dispositivos de neblina, são distribuídos pelas camadas da topologia de rede. São necessárias políticas de gerenciamento de dados e serviços para decidir quando e onde colocar os serviços e os dados.
Instâncias de Teste ou *Benchmarks* utilizados:	100,200
Ambiente de testes:	-
Principais resultados e contribuições apresentados:	– Os resultados mostraram que o método obteve as mais altas otimizações dos objetivos e a maior diversidade do espaço da solução. – Todas as instâncias propostas foram resolvidas.

Identificador	[P46]
A) Dados da publicação	
Título:	Task optimization and scheduling of distributed cyber–physical system based on improved ant colony algorithm.
Autores:	Na Yi, Jianjun Xun, Limei Yan e Lin Huang.
Fonte de Publicação:	ScienceDirect Future Generation Computer Systems
Ano da Publicação:	2020
Resumo:	O sistema ciber-físico (CPF) é o produto do desenvolvimento tecnológico até um certo estágio e também as tendências futuras da tecnologia da informação. A capacidade computacional de alto desempenho é a garantia dos aplicativos de precisão e em tempo real do CPF, e o surgimento da tecnologia distribuída oferece a possibilidade de implementação do CPS de alto desempenho. O agendamento de tarefas é um problema típico de otimização de combinação e o problema de alocação de tarefas em sistemas distribuídos com vários processadores refere-se a como usar os recursos do sistema com mais eficiência em um ambiente de computação distribuído para concluir um conjunto limitado de tarefas.
B) Dados derivados do objetivo	
Tipos de Algoritmos Utilizados:	Algoritmo de Otimização por Colônia de Formigas.
Variação do(s) Problema(s) a ser(em) resolvidos(s):	A essência do agendamento de tarefas é alocar n tarefas independentes para m recursos disponíveis heterogêneos, para que o tempo total de conclusão da tarefa seja minimizado e os recursos totalmente utilizados. Possui uma plataforma heterogênea, em larga escala, descentralizada e recursos como escalabilidade. Cada agendador de tarefas deve obedecer às restrições de recursos deste processador quando ele é executado. O problema de agendamento de

	tarefas no sistema refere-se à atribuição de cada tarefa a um determinado processador para execução, sob a condição de que certas restrições sejam atendidas e a função objetivo correspondente seja minimizada.
Instâncias de Teste ou *Benchmarks* utilizados:	200-4000
Ambiente de testes:	-
Principais resultados e contribuições apresentados:	– Os resultados da simulação mostram que o modelo de algoritmo proposto aprimora a capacidade de busca local e melhora a qualidade do problema de agendamento de tarefas, além de ter boa eficácia, estabilidade e adaptabilidade. – Todas as instâncias propostas foram resolvidas.

APÊNDICE C – RESULTADOS DO MAKESPAN OBTIDOS E TEMPO DE CPU COM AS INSTÂNCIAS DA PSPLIB

Instâncias	*Makespan*	*Tempo de CPU*
J301	28	0.22s
J302	35	0.24s
J303	37	0.26s
J304	39	0.27s
J305	53	0.36s
J306	39	0.31s
J307	39	0.33s
J308	33	0.34s
J309	62	0.37s
J3010	32	0.34s
J3011	35	0.33s
J3012	39	0.34s
J3013	42	0.35s
J3014	45	0.31s
J3015	36	0.32s
J3016	38	0.36s
J3017	34	0.32s
J3018	30	0.33s
J3019	31	0.35s
J3020	32	0.32s
J3021	34	0.36s
J3022	39	0.33s
J3023	37	0.36s
J3024	41	0.34s
J3025	85	0.34s
J3026	35	0.33s
J3027	33	0.33s
J3028	41	0.35s
J3029	33	0.31s
J3030	42	0.35s
J3031	33	0.32s
J3032	31	0.33s
J3033	37	0.35s
J3034	34	0.32s
J3035	38	0.31s
J3036	32	0.34s
J3037	36	0.31s
J3038	36	0.33s
J3039	37	0.34s
J3040	32	0.33s
J3041	41	0.33s
J3042	39	0.33s
J3043	38	0.33s
J3044	33	0.35s

J3045	86	0.47s
J3046	39	0.33s
J3047	40	0.34s
J3048	32	0.29s
J6001	59	0.69s
J6002	50	0.68s
J6003	44	0.69s
J6004	49	0.71s
J6005	64	0.77s
J6006	50	0.74s
J6007	52	0.78s
J6008	55	0.63s
J6009	52	0.77s
J6010	49	0.79s
J6011	63	0.79s
J6012	35	0.71s
J6013	81	0.85s
J6014	39	0.74s
J6015	35	0.69s
J6016	44	0.74s
J6017	44	0.76s
J6018	48	0.74s
J6019	52	0.71s
J6020	39	0.72s
J6021	48	0.71s
J6022	54	0.71s
J6023	44	0.74s
J6024	53	0.71s
J6025	84	0.79s
J6026	51	0.71s
J6027	49	0.76s
J6028	36	0.69s
J6029	68	0.66s
J6030	56	0.77s
J6031	52	0.72s
J6032	56	0.76s
J6033	49	0.79s
J6034	51	0.74s
J6035	33	0.66s
J6036	74	0.72s
J6037	51	0.71s
J6038	52	0.74s
J6039	55	0.71s
J6040	74	0.72s
J6041	40	0.66s
J6042	61	0.72s
J6043	50	0.79s
J6044	42	0.71s
J6045	85	0.76s
J6046	46	0.76s

J6047	43	0.73s
J6048	46	0.76s
X1	93	2.44s
X2	69	2.41s
X3	71	2.42s
X4	63	2.51s
X5	59	2.31s
X6	57	2.25s
X7	62	2.36s
X8	58	2.39s
X9	59	2.39s
X10	51	2.25s
X11	55	2.42s
X12	58	2.42s
X13	61	2.34s
X14	65	2.36s
X15	62	2.36s
X16	72	2.53s
X17	55	2.42s
X18	55	2.31s
X19	60	2.11s
X20	59	2.39s
X21	54	2.25s
X22	67	2.34s
X23	61	2.23s
X24	58	2.33s
X25	59	2.25s
X26	86	2.34s
X27	57	2.42s
X28	82	2.25s
X29	68	2.53s
X30	58	2.25s
X31	64	2.53s
X32	96	2.11s
X33	67	2.33s
X34	73	2.33s
X35	52	2.33s
X36	89	2.34s
X37	75	2.38s
X38	80	2.39s
X39	71	2.53s
X40	59	2.29s
X41	56	2.53s
X42	54	2.25s
X43	65	2.33s
X44	55	2.53s
X45	66	2.30s
X46	53	2.33s
X47	112	2.34s
X48	85	2.26s

X49	60	2.19s
X50	79	2.29s
X51	50	2.11s
X52	78	2.42s
X53	97	2.53s
X54	70	2.34s
X55	66	2.19s
X56	62	2.34s
X57	60	2.19s
X58	90	2.33s
X59	78	2.30s
X60	61	2.11s