Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Matheurísticas para o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos

Lettiery D'Lamare Portela Procópio

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciências da Computação
Linha de Pesquisa: Computação Distribuída

Lucídio dos Anjos Formiga Cabral (Orientador)

João Pessoa, Paraíba, Brasil

P963m Procópio, Lettiery D'Lamare Portela.
Matheurísticas para o problema de custo de disponibilidade de recurso com múltiplos modos / Lettiery D'Lamare Portela Procópio.- João Pessoa, 2016.
56f. : il.
Orientador: Lucídio dos Anjos Formiga Cabral
Dissertação (Mestrado) - UFPB/CI
1. Informática. 2. Custo. 3. Disponibilização de recursos. 4. Escalonamento de atividades. 5. Algoritmo genético. 6. Matheurística.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **LETTIERY DLAMARE PORTELA PROCOPIO**, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 02 de fevereiro de 2016.

Ao segundo dia do mês de fevereiro do ano de dois mil e dezesseis, às quatorze horas, no Centro de Informática - Universidade Federal da Paraíba (unidade Mangabeira), reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final do **Sr. Lettiery Dlamare Portela Procopio** vinculado a esta Universidade sob a matrícula 2014108485, candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa *"Sistemas Distribuídos"*, do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: **Dr Lucidio dos Anjos Formiga Cabral (PPGI-UFPB),** Orientador e Presidente da Banca, **Dr Roberto Quirino do Nascimento (UFPB),** Examinador Externo ao Programa e **Dr Marco Cesar Goldbarg (UFRN)**, Examinador Externo à Instituição. Dando início aos trabalhos, o professor Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado *"Matheurísticas para o Problema de Custo de Disponibilidade de Recursos com Multiplos Modos"*. Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "*aprovado*". Assim sendo, eu, Clauirton de Albuquerque Siebra, Coordenador do Programa de Pós-Graduação em Informática - PPGI, lavrei a presente ata que vai assinada por mim e pelos membros da Banca Examinadora.

João Pessoa, 02 de fevereiro de 2016.

Clauirton de Albuquerque Siebra

Prof Dr Lucidio dos Anjos Formiga Cabral
Orientador (PPGI-UFPB)

Prof Dr Roberto Quirino do Nascimento
Examinador Externo ao Programa (UFPB)

Prof Dr Marco Cesar Goldbarg
Examinador Externo à Instituição (UFRN)

Resumo

Este trabalho descreve a construção de duas Matheurística baseadas em Algoritmos Genéticos e na Otimização por Enxame de Partícula, afim de solucionar o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos. Inspirado na necessidade de balancear a utilização de recursos renováveis com o tempo total (*makespan*), através do escalonamento das atividades com seus diversos modos de execução presentes no projeto. Testes revelam a eficácia na utilização da programação matemática adaptada com o Algoritmo Genético.

Palavras-chave: Custo, Disponibilização de recursos, Escalonamento de atividades, Algoritmo Gen ético, Matheurística.

Abstract

This paper discribes the construction of two Matheuristics based on Genetic Algorithm and Particle Swarm Optimization in order to solve the Availability of Cost Problem Resources with Multi-Modes. Inspired by the need to balance the use of renawable resources with the total time (makespan), by scheduling the activities with its various implementations executions modes present in the project. Tests show the effectiveness in te use of mathematical programming adapted to the Genetic Algorithm.

Keywords: Cost, Resource Availability, Activity scheduling, Genetic Algorithm, Matheuristics.

"Não sabendo que era impossível, ele foi lá e fez"

Jean Cocteau

Agradecimentos

Dedico este Trabalho aos meus pais Jorge e Nilda, que me educaram para a vida. Aos meus amigos Bruno, Paulo e Luís Feliphe, minha eterna gratidão por compartilhar comigo seus conhecimentos e momentos de diversão. A minha companheira de vida Amanda Vilar, que de todas as formas me fortificou, encorajou e meu deu alegria nos momentos mais difíceis. Quero agradecer as meus professores, em especial ao Tio Gilberto que me mostrou a beleza da ciência e ao meu orientador Lucídio, que me guio pelos caminhos dos saberes. Por fim, porém o mais especial, agradeço a Deus por minha existência, sabedoria e fé.

Conteúdo

Lista de Símbolos

AG : *Algoritmo Genético*

PSO : *Particle Swarm Optimization*

AGM : *Algoritmo Genético - Matheurística*

PSOM : *Particle Swarm Optimization - Matheurística*

SS : *Scatter Search*

CPM : *Critical Path Method*

PERT : *Project Evaluation and Review Technique*

PCDRMM : *Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos*

PCDR : *Problema de Custo de Disponibilidade de Recursos*

PPPRL : *Problema de Programação de Projetos com Recursos Limitados*

PPMRL : *Programação de Projetos com Multi- Modos e Recursos Limitados*

PO : *Pesquisa Operacional*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

1.1 Apresentação

O processo de tomada de decisão tem sido a base da computação científica desde seu nascimento e já é pressuposta por alguns autores como uma ciência própria, a Ciência da Gestão (*Management Science*), que escolhe métodos científicos para a avaliação de alternativas das ações que devem ser realizadas [Nobrega 2004]. É comum atribuir a Frederic Taylor a obstinação pela eficiência, o combate ao desperdício e o melhor uso dos recursos produtivos – detalhado pelo seu livro Príncipe da Administração Científica publicado em 1911, que se tornou o marco inicial da Ciência da Gestão. Certamente, o ato de refletir e planejar os passos de um determinado projeto impacta fortemente na sua execução, destinando-o para o sucesso ou fracasso em casos de omissão desta etapa.

Assim, o planejamento de projetos necessita da coordenação de atividades relacionadas para atingir objetivos pré-definidos. Muitos desses objetivos são pensados cobiçando o tempo e o custo que irá demandar para a concretização total do projeto. Ações dessa classe, que tentam organizar as atividades de maneira a alcançar um objetivo, são mais conhecidas na literatura da computação como problemas de programação de projetos (*scheduling*), nos quais uma das principais metas é a minimização do tempo de término do projeto, também conhecido como *makespan*.

Ainda são encontradas diversas formas clássicas de *scheduling*, como o *JobShop*, que tem como objetivo principal a conclusão de *jobs* (conjuntos de atividades com uma dada ordem), em um número finito de máquinas. Cada finalização acrescenta unidades de tempo

pré-definidas ao projeto. A precedência entre as atividades, menciona a prioridade de termino das mesmas, e pode ser vista como a restrição que possibilita as inúmeras combinações de organização de um projeto, impactando diretamente no *makespan*.

Ao modificar algumas propriedades de execução, encontramos outras nomenclaturas para projetos de escalonamento, como o *OpenShop*, onde há diversas máquinas e atividades a serem processadas sem precedências. Já o *FlowShop* determina um fluxo de execução nas máquinas para cada *job*. Esses problemas compartilham o alto número de soluções factíveis com, no entanto, um conjunto finito de soluções viáveis, por isso, são classificados como problemas de otimização combinatória.

Também com um grande número de possíveis soluções, surgiram no fim da década de 50, problemas que necessitavam da alocação de recursos para a realização das atividades ao longo do projeto, suas primeiras resoluções foram através do clássico método de caminho crítico CPM (*Critical Path Method*) e PERT (*Project Evaluation and Review Technique*). Uma revisão interessante pode ser vista em [Moder, Phillips e Davis 1983]. Desde então, a programação de projetos tem sido alvo constante de pesquisas, influenciada também por sua grande aplicação em problemas reais, como a construção de pontes e grandes edificações, a produção nas indústrias, projetos de roteamento de cargas e recentemente na computação em nuvens, onde os computadores são como os recursos a serem alocados, o processamento são como as atividades a serem realizadas e o *makespan* como o tempo de processamento, que é um dos objetivos pré-definidos a serem alcançar.

Na busca da solução de um projeto que se aplica a este último exemplo de computação em nuvens, podemos separar duas necessidades que devem ser atendidas: a primeira, se refere a otimizar o *makespan* do projeto, ou seja, o projeto deve ser entregue o mais rápido possível; a segunda é a definição da menor quantidade de computadores (recursos) que utilizaremos para realizar o processamento, já que para cada recurso alocado teremos a adição de um custo ao projeto. Se esses dois objetivos forem avaliados em conjunto, podemos ver um conflito de ideias, uma vez que para diminuir o tempo de processamento total do projeto deveremos utilizar mais recursos, o que aumentaria o custo do projeto, e para diminuir esse custo, consequentemente, teremos um processamento mais lento.

Problemas como esse que envolvem custo de recursos e tempo do projeto são bastante pesquisados pela sua grande aplicabilidade em diversas áreas, esse é um dos argumentos que

direcionou a pesquisa pela busca de soluções para o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos (PCDRMM), o qual tenta otimizar o custo da alocação de recursos nas atividades, preocupando-se também com o *makespan* do projeto.

Esse problema se torna mais aplicável a situações reais pela característica de suas atividades que assumem diversos modos de execução, impactando na sua quantidade de recursos e tempo de processamento gastos, por isso sua descrição utiliza o termo "múltiplos modos". Com um exorbitante número de soluções factíveis, sua resolução não é imediata e utilizaremos vários trabalhos para debater a escolha de métodos científicos que solucionem em um tempo aceitável.

Alguns dos trabalhos encontrados utilizam métodos exatos para buscar soluções ótimas, no entanto, a literatura mostra que esses métodos são custosos em relação ao tempo de processamento e sua aplicação é possível apenas em instâncias pequenas do problema. Este argumento pode ser confirmado nos testes computacionais apresentado no Capítulo 6.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo principal deste trabalho é o desenvolvimento de Matheurísticas para a construção de soluções eficientes no problema de custo de disponibilidade de recursos com múltiplos modos.

1.2.2 Objetivos Específicos

- Revisar e discutir o estado da arte do problema de custo de disponibilidade de recursos com múltiplos modos para auxiliar na sua resolução.
- Verificar algoritmos para a construção de boas soluções em tempo computacional aceitável.
- Aplicar métodos heurísticos para o teste das novas ideias e conhecimentos adquiridos pelos estudos de trabalhos relacionados.

- Adaptar a formulação matemática do problema para auxiliar na construção das Matheurísticas para encontrar boas soluções em um tempo aceitável.

- Comparar resultados obtidos pelas novas construções com soluções obtidas pela implementação da modelagem do problema.

1.3 Relevância

A criação de métodos específicos para auxiliar no processo de tomada de decisão foi uma prática presente na Segunda Guerra Mundial, onde a necessidade de realizar grandes cálculos para estratégias bélicas deu início a Investigação Operacional ou Pesquisa Operacional, um ramo interdisciplinar da matemática aplicada que se apropria de modelos e heurísticas para avaliar linhas de ação ou organização dos indivíduos.

O término da guerra trouxe ainda a expansão capitalista e a concorrência entre as grandes empresas, que para inovar nas suas estratégias de mercado buscavam o auxílio de novos métodos tecnológicos para melhorar o processo de produção e organização das atividades, resultando na logística, uma área de planejamento responsável por organizar, armazenar e controlar a movimentação de produtos e serviços relacionados, desde a concepção da matéria prima até o consumo final do produto, como relatado por [NOVAES 2001].

A logística apresentada por [Ballou 1993] como uma ferramenta que proporciona o aprimoramento de serviços e produtos ao seu cliente, se transforma em mais um artefato estratégico que as empresas utilizam para organizar suas atividades. Já [NETO 2001] mostra que a logística quando é verdadeiramente assimilada possibilita a redução de custos e aperfeiçoamento dos produtos oferecidos. Esses autores mostram que a logística aumenta o potencial competitivo dessas empresas no mercado. Dessa forma, a redução de custos é um dos principais objetivos a serem alcançados, além disso, problemas como o PCDRMM, adicionam valores concretos à empresas que pesquisam como solucioná-los.

Soluções para problemas como estes são medidas por duas grandezas distintas: a quantidade de recursos que será utilizada e o tempo final do projeto. Métodos exatos se apresentam como a melhor escolha para solucionar problemas dessa classe, onde soluções ótimas seriam o cenário desejável para a indústria, no entanto, o grande número de variáveis e cenários possíveis para um projeto nos remete a métodos heurísticos que tragam valores esperados e

ocasionem ótimas soluções em projetos menores.

Assim, este trabalho é motivado pela busca de soluções de boa qualidade para o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos através de exemplos na literatura que obtiveram sucesso com heurísticas para resolver problemas onde a utilização de recursos é cara para o projeto como um todo ou que ainda existe uma escassez dos recursos utilizados.

O PCDRMM pode ser aplicado a vários problemas reais, como na construção civil, onde máquinas de construção, matéria prima e equipes de trabalhadores são geralmente vistos como recursos escassos e onde também existe a necessidade de minimizar o tempo do projeto como um todo. Junto com a computação nas nuvens, projetos de produção industrial e até projetos de desenvolvimento de softwares podem ser mapeados para o PCDRMM, sua solução em forma de um algoritmo que possibilita a otimização desses projetos demonstra ainda mais a capacidade da computação em facilitar tarefas diárias.

1.4 Estrutura da Dissertação

O Capítulo 1 apresenta o contexto em que está inserido este trabalho em conjunto com os principais objetivos e as motivações que fundamentaram esta pesquisa.

O Capítulo 2 destrincha as variáveis, restrições e objetivos do PCDRMM e sua semelhança com problemas desta classe.

O Capítulo 3 traz uma revisão dos trabalhos relacionados presentes na literatura e voltados para problemas semelhantes, detalhando heurísticas utilizadas por esse grupo de pesquisa.

O Capítulo 4 mostra os algoritmos propostos e as principais heurísticas que foram utilizadas para a construção deste projeto.

O Capítulo 5 apresenta os resultados dos testes computacionais e uma comparação dos métodos propostos.

Por fim, o Capítulo 6 realiza as considerações finais e expectativas para trabalhos futuros.

Capítulo 2

Formulação e Classificação do Problema

Este capítulo apresenta restrições, classificações e objetivos do Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos e outros problemas com objetivos semelhantes.

2.1 Problemas Semelhantes

Problemas de programação de projetos, em geral, consistem em determinar um tempo de início para todas as atividades a fim de minimizar o *makespan*, porém quando os recursos são escassos, sua prioridade é alterada a minimização do custo atribuido a estes recursos. Assim é descrito o Problema de Programação de Projetos com Recursos Limitados (PPPRL) por [Blazewicz, Lenstra e Kan 1983], dessa forma, este trabalho provou que o PPPRL é da classe *NP-Hard*. Outras propostas podem ser vistas em [Herroelen, B. e Demeulemeester 1998], [Brucker et al. 1999], [Kolisch e Hartmann 1998] e [Hartmann e Kolisch 2000].

O Problema de Custo de Disponibilidade de Recursos (PCDR), também conhecido como *Resource Availability Cost Problem* (RACP), é distinto do PPPRL pelo tempo escasso do projeto e recursos ilimitados, além de apresentar um único modo de execução das atividades, onde a sua formulação original tenta alocá-las da melhor maneira para que o custo do projeto final seja reduzido (custo esse medido pela utilização dos recursos). Tudo isso com data de entrega determinada e quantidade enumerada de disponibilidade dos recursos.

O PCDR foi proposto por [Möhring 1984], encorajado por projetos de construções de pontes de pequeno porte. Tem sido resolvido com um método exato baseado em modelos gráficos. Möhring ainda mostrou por experimentos computacionais que o problema também

é da classe *NP-Hard*.

A Programação de Projetos com Multi-Modos e Recursos Limitados (PPMRL), também conhecida por *Multi-Mode Resource-Constrained Project Scheduling Problem* (MRCPSP), pode ser definido como outro problema semelhante que traz a ideia das várias maneiras de execução das atividades "Multi-Modos", porém com recursos limitados a um número constante pré-definido, sem contabilizar custos para o projeto final. É focada apenas na otimização do *makespan*, proporcionando melhor escolha na ordenação de precedências das atividades. Para esse novo problema, muito trabalho tem sido desenvolvido, incluindo métodos heurísticos e modelos exatos.

O Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos (PCDRMM) também conhecido como *Multi-Mode Resource Availability Cost Problem* (MRACP), se apresentou na literatura como um novo problema, entretanto, através de características de problemas anteriormente citados foi possível chegar a sua formulação completa. Um *mix* de algumas restrições e objetivos deu origem ao PCDRMM, cujo custo de disponibilização de recursos pode ser visto no PCDR. Já a possibilidade de uma atividade ser executada de várias formas distintas foi espelhada do PPMRL. E do PPPRL (Problema de Programação de Projetos com Recursos Limitados) foi herdada a tentativa de minimização do *makespan*.

Uma adaptação do PCDRMM (MRACP) introduzida por [Nadjafi 2014] foi denominada como *Multi-Mode Resource Availability Cost Problem with Recruitment and Release dates for resources* (MRACP - RR), que segue as mesmas regras do MRACP, porém, com uma alteração na forma de contabilizar o custo da disponibilização dos recursos, traz a mesma ideia de custo por unidade de tempo, onde se um recurso permanecer disponível, mesmo sem utilização por apenas 5 unidades de tempo do projeto, o custo do recurso será contabilizado por essas 5 unidades, ou seja, seu custo será calculado somente durante o tempo que permanecerá no projeto.

Uma comparação das características dos problemas acima citados, pode ser encontrada na Tabela 2.1.

Problemas/Siglas	Recursos limitados	Múltiplos modos	Custo por disponibilização	Data de entrega	Custo por tempo
PPPRL	X				
PCDR, RACP			X	X	
PPMRL, MRCPSP	X	X		X	
PCDRMM, MRACP		X	X	X	
PCDRMM - RL, MRACP - RR		X	X	X	X

Tabela 2.1: Comparação de problemas conhecidos

Ao confrontar todos os problemas podemos observar que o Problema de Programação de Projetos com Recursos Limitados (PPPRL) é o único que não possui uma data de entrega do projeto, diferente do Problema de Programação de Projetos com Múltiplos Modos e Recursos Limitados (PPMRL), no qual também existe a limitação dos recursos a serem utilizados, mas com data de entrega para o projeto.

O Problema de Custo de Disponibilização de Recursos (PCDR) não apresenta a característica de vários modos de execução das atividades, contudo, assim como o Problema de Custo de Disponibilização de Recursos com Múltiplos-Modos (PCDRMM), o PCDR também contabiliza o custo dos recursos por todo o horizonte do projeto, mesmo que não esteja utilizado-os. Diferente da sua adaptação o Problema de Custo de Disponibilização de Recursos com Múltiplos-Modos com datas de Recrutamento e Liberação dos recursos (PCDRMM-RL) contabiliza o custo apenas nas unidades que o recurso estiver disponível.

No nosso trabalho, elegemos o PCDRMM como problema a ser solucionado pela característica da maioria das atividades reais de poder ser executada de várias maneiras distintas, e por tratar um recurso como um objeto que pode ser alugado ou que são comprados por todo o projeto. Projetos de construção civil, computação nas nuvens e a produção industrial podem ser mapeados para o PCDRMM como uma forma mais completa e dinâmica do que o PCDR.

2.2 Descrição do Problema

Podemos definir o PCDRMM como um problema que balanceia a minimização da utilização dos recursos renováveis e o tempo de duração do projeto, contudo, suas atividades podem assumir diversos modos de execução que variam o tempo e a quantidade de recursos utilizados para concretiza-las. Elas ainda possuem uma relação de precedência temporal que

determina o tempo mínimo de início das mesmas.

O custo de utilização dos recursos é contabilizado por todo o horizonte do projeto, que podem ser reutilizados por outras atividades sem adicionar quaisquer gastos à função objetivo. Este problema equilibra a utilização dos recursos de forma a não penalizar o tempo final do projeto, por isso, sua aplicabilidade a projetos reais se apresenta de forma imediata ou de fácil adaptação.

2.3 Classificação

2.3.1 Classificação segundo a utilização dos recursos

Projetos que envolvem alocação de recursos tendem a duas situações distintas: na primeira, a oferta é maior do que a demanda, assim não há preocupação com a sua utilização; na segunda os recursos são escassos e a utilização impactará diretamente no desenvolvimento do projeto, nesse caso, há ainda três possíveis eventos: (1) quando existe uma necessidade de redução da variação dos perfis da demanda; (2) se faz necessário reduzir a duração do projeto e adicionar recursos com os menores custos; (3) deve-se alocar as atividades de maneira que as quantidades disponíveis de recursos não sejam ultrapassadas em nenhum tempo.

Segundo [Daveis e Patterson 1975], a classificação dos problemas de alocação de recursos se dá na seguinte ordem de nomenclatura: (1) Problemas de Nivelamento de Recursos, (2) Problema de Compressão de Projetos e (3) Problema de Alocação de Recursos Limitados. O PCDRMM se encaixa na terceira definição, devido à sua necessidade de determinar uma data de início para as atividades a fim de minimizar o custo da disponibilização dos recursos.

Os recursos podem ser classificados por suas características de utilização, como Renováveis, Não-renováveis e Duplamente Restritos:

- Recursos Renováveis (*Renewable*) são limitados em quantidade, mas reutilizáveis de período a período sem adição de custo ao projeto, como a mão de obra, máquinas, ferramentas ou espaços reservados para as atividades.

- Recursos Não-Renováveis (*Non-Renewable*) são recursos que, ao término do período de utilização da atividade, é contabilizado o seu custo para o projeto. Assim, cada

vez que qualquer atividade fizer uso do recurso, será acrescentado seu custo ao projeto como valor monetário, energia, serviços ou matéria prima.

- Recursos Duplamente Restrito (*Doubly Constrained*) quando os recursos são limitados na sua soma e por período, como dinheiro, máquinas, ferramentas ou matéria prima.

2.3.2 Classificação segundo a complexidade

Como o PCDRMM é gerado da forma estática do *JobShop*, devido às suas características de minimização do *makespan*, alocação de atividades com seus recursos e possibilidade das atividades serem executadas de várias formas diferentes, torna-se ainda mais complexo o problema que, além disso, é uma variante dos problemas PPMRL, PCDR e PPPRL.

Esses problemas são de Otimização Combinatória pertencentes à classe NP-Hard em [Möhring 1984; Blazewicz, Lenstra e Kan 1983; Parker e Rardin 1981; Parker e Rardin 1982]. Assim, podemos classificar o PCDRMM como também um problema NP-Hard.

2.4 Características

2.4.1 Característica das Atividades

Vale ressaltar que o principal objetivo do PCDRMM é organizar as atividades a fim de minimizar o tempo total do projeto (*makespan*) e o custo atribuído à disponibilização dos recursos para as atividades. Assim, uma das primeiras características que torna o problema ainda mais aplicável a projetos reais, é a possibilidade de uma atividade ser realizada de várias maneiras distintas.

Um exemplo dessa característica pode ser dado pelo projeto de construção de uma casa, na qual, se contratamos um grupo de 5 trabalhadores, pode ser finalizado em um prazo de 10 semanas, já se esse número de contratados fosse dobrar para 10, a construção seriá concluída em apenas 6 semanas. Ainda cabe um terceiro caso, onde teremos nossos 5 trabalhadores contratados e 2 máquinas especiais para auxiliar na construção do alicerce da casa, dessa forma, nosso projeto iria ser finalizado em um cruto prazo de 5 semanas.

Assim, as atividades do PCDRMM podem ser descritas como ações a serem realizadas que necessitam de uma quantidade variante de tempo, de acordo com uma quantidade (tam-

bém variante) de recursos utilizados. Essas atividades influenciam em duas grandezas do projeto: o tempo de execução e a quantidade de recursos utilizados.

Como exemplo, a Figura 2.1 abaixo mostra uma atividade "A"que pode ser executada de 3 modos distintos: no primeiro modo a atividade realiza suas ações em 5 unidades de tempo e gasta 3 recursos do tipo vermelho e 1 do tipo azul; no segundo modo ela gasta 8 unidades de tempo e apenas 2 recursos do tipo vermelho; e no terceiro modo ela utiliza 3 unidades de tempo com 2 recursos do tipo vermelho e 2 do tipo azul.

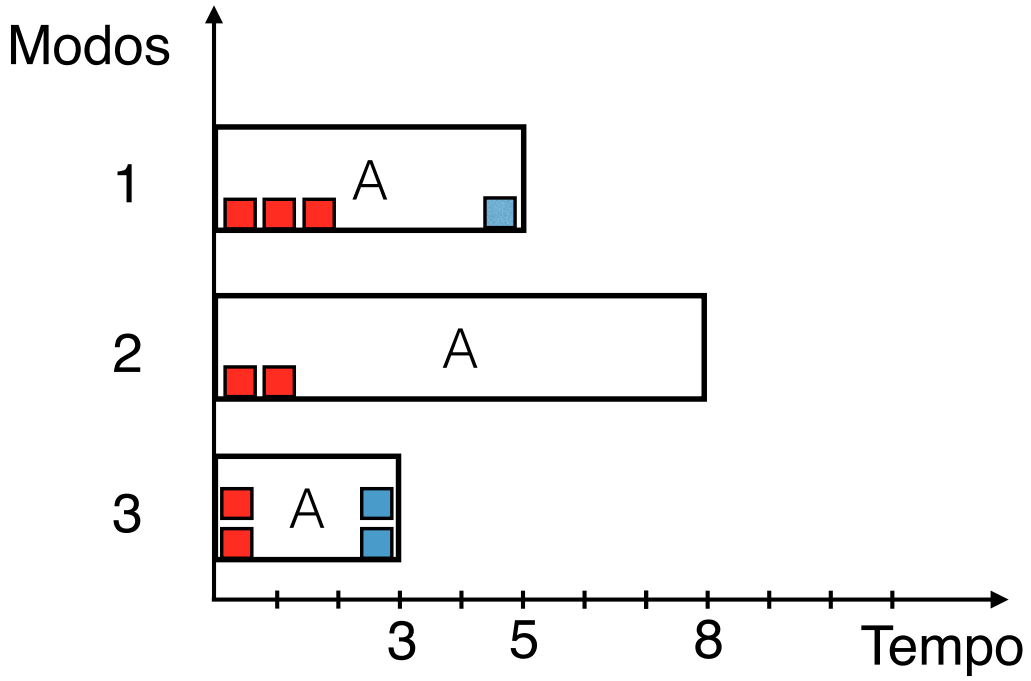

Figura 2.1: Exemplos de modos de atividades

A escolha desses modos pode variar de acordo com a disponibilidade dos recursos no projeto e o custo a eles atribuídos. No exemplo acima, pensando apenas no menor tempo gasto, a eleição do terceiro modo seria a escolha ótima, e se desconsiderarmos a variável temporária das atividades, a utilização do segundo modo seria, então a melhor opção.

A precedência das atividades pode ser um fator que influência na escolha de seus modos, dada por um relacionamento lógico onde uma atividade sucessora (B) só poderá iniciar ao término de uma atividade predecessora (A), o tempo entre o término de A e o início de B não necessariamente será 0, a natureza desse relacionamento é vista claramente entre as ati-

vidades envolvidas, como em um projeto de construção civil onde a atividade que construirá um telhado de uma casa não pode ser realizada antes ou durante a atividade de construção da sua base.

Para marcar o início e o fim do projeto, foram adicionadas duas atividades virtuais (s, t), onde o custo e o tempo são zero, assim, toda atividade sucessora da atividade que marca o início do projeto s, não necessitará do término de qualquer outra atividade para ser processada, e a atividade que é predecessora de t, que define o tempo final do projeto, pode ser compreendida como a última atividade e um dos objetivos de minimização (o *makespan*).

A Figura 2.2 mostra um exemplo que envolve 6 atividades. As duas atividades virtuais s e t representam respectivamente o fim e o início do projeto, o arco do grafo ilustra a precedência entre a atividade 1 e a atividade 3, esta última ainda tem como predecessora a atividade 2, e a atividade 4 não precede nem sucede nenhuma atividade real. A utilização de s permite a visualização de 3 atividades que podem iniciar o projeto (1, 2, 4), e a adoção de t facilita ver as duas possíveis atividades que marcam o final do projeto (3, 4).

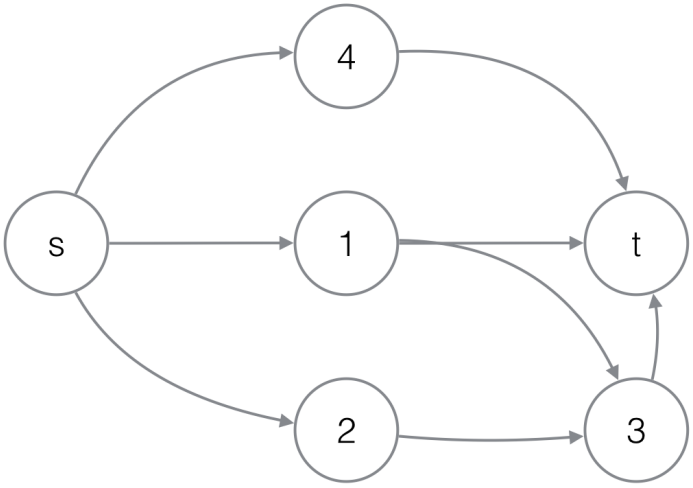

Figura 2.2: Exemplo de grafo de precedência entre as atividades

2.4.2 Característica do Custo

O PCDRMM calcula o custo de disponibilidade e não de utilização do recurso, que para esse problema se torna recurso renovável, uma vez que ao ser utilizado em uma tarefa a reutilização não será mais contabilizada por outras atividades, ou seja, o pagamento por ele é feito por todo o projeto, logo, se durante nosso projeto forem realizadas várias reutilizações

do mesmo recurso, não será cobrado um custo adicional. Como descrito por [Nudtasomboon e Randhawa 1997], recursos renováveis são aqueles que ao término do seu uso voltam a estar disponíveis para sua realocação em outra atividade. Mão de obra, aquisição de equipamentos e máquinas podem servir como exemplos de recursos renováveis.

O projeto pode utilizar m tipos de recursos, somando diferentes custos de utilização para cada um deles. Esse custo é dado por uma função linear não decrescente $C_k(a_k)$ que representa a quantidade a_k do recurso do tipo k, ou seja, se forem utilizados 4 recursos do tipo 3 no projeto, isso contabilizará o valor retornado da função $C_3(4_3)$. A soma de todas as quantidades e de todos os tipos de recursos, representa o custo total do projeto.

Outra maneira de contabilizar o custo também podem ser encontrada na literatura, a figura 2.3 é um exemplo de como contabilizar o PCDRMM-RR.

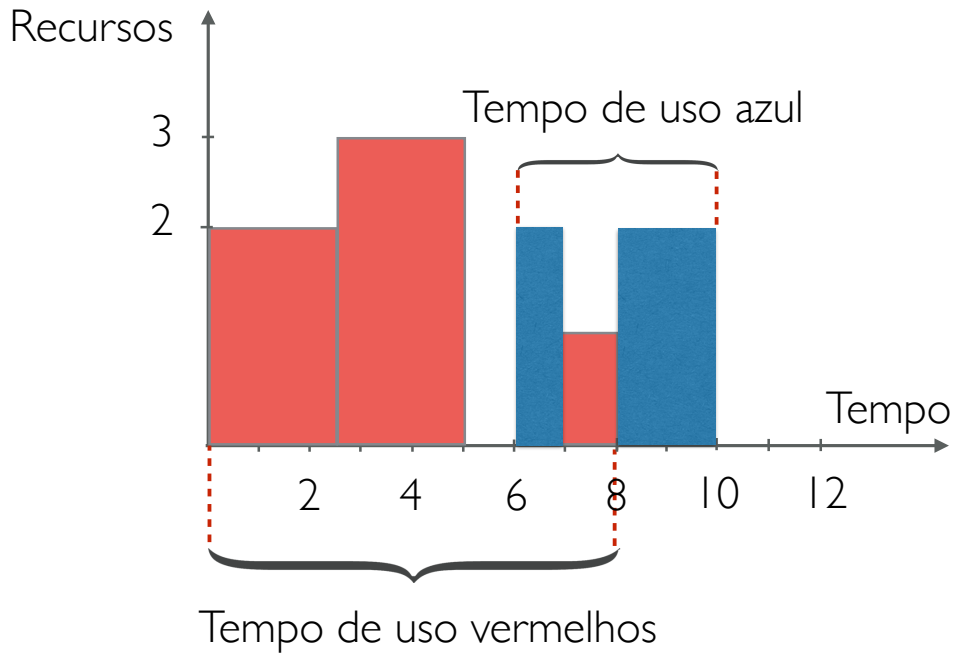

Figura 2.3: Exemplo de custo do PCDRMM-RR

Nessa variante, que se diferencia do PCDRMM justamente por contabilizar apenas o período de disponibilização do recurso, pode ser observado que o recurso do tipo vermelho foi disponibilizado no início até o tempo 8 e, mesmo não sendo utilizado do tempo 5 ao 6, será somado o valor das 8 unidades de tempo ao projeto. A função objetivo da solução exemplificada na Figura 2.3, toma como valor por unidade de tempo do recurso azul Vu_{azul} = 4 reais

e valor unitário do recurso vermelho $Vu_{vermelho}$ = 2 reais, para o recurso do tipo vermelho foram atribuídas 8 unidades de tempo e disponibilizados 3 recursos com uma parcial: 3 x 8 x 2 = 48 reais do recurso vermelho. Para o azul temos 4 unidades de tempo, 2 recursos: 2 x 4 x 4 = 32 reais, totalizando 80 reais do projeto final.

Já no PCDRMM não existe o valor por unidade de tempo, e sim por todo o projeto, com isso podemos adotar que para um projeto com prazo de término 10 o aluguel do recurso vermelho é $a_{vermelho}$ = 20 reais, já para o azul temos a_{azul}= 40 reais, então o exemplo da função objetivo para a solução da Figura 2.3 adotando o PCDRMM teremos: $C_{vermelho}(3) =$ 3 x 20 = 60 reais + $C_{azul}(2) =$ 2 x 40 = 80 reais, totalizando 140 reais como custo final do projeto.

A diferença entre os dois tipos de função objetivo pode ser observada em projetos onde o uso do recurso não se destina apenas à utilização da mão de obra do projeto, mas também para manutenção e investimento do recurso. Podemos ver casos onde recursos são máquinas ou equipes de funcionários, que ao fim de suas atividades podem participar de capacitações, efetuar outras atividades secundárias. Além disso, contratações onde existe um abatimento no valor completo do projeto e até em casos isolados que o tempo ocioso também estará previsto no contrato pode resultar em menor custo para a empresa.

2.5 Formulação Matemática

A formulação matemática descrita nesse trabalho foi extraída de [Yamashita e Morabito 2012], onde o PCDRMM se apresenta como um projeto com n atividades, das quais a primeira e a última atividade (1 e n) são equivalentes às atividades virtuais$s$ e t , representando o início e o fim do projeto. O par ordenado (h, j), quando existe no conjunto H, representa uma relação de precedência da atividade h com j, ou seja, se $(h, j) \in H$, então h precede j. M_j é a quantidade de modos de execução da atividade j. Quando j é executada no modo i requer d_{ji} unidades de tempo para finalizar e r_{jik} quantidades do recurso do tipo $k(k = 1, \ldots, m)$. O custo da disponibilidade da quantidade de recurso a_k do tipo k é dado por uma função linear não decrescente $C_k(a_k)$, é adotado que a variável a_k assume valor constante em relação ao tempo. A data final de entrega do projeto é dado por D. Yamashita, baseada em [Talbot 1982], apresenta um modelo com as seguintes variáveis de decisão:

- $x_{jit} = \begin{cases} 1, & \text{se a atividade } j = 1, ..., n \text{ é executada no modo } i = 1, ..., M_j \text{ e termina} \\ & \text{no instante } t = 1, ..., D. \\ 0, & \text{caso contrário.} \end{cases}$
- a_k = quantidade de recursos do tipo k disponível ao longo do projeto, $k = 1, \ldots, m$.

Os dados de entrada do problema são:

- $C_k(a_k)$ = função de custo não decrescente associada à disponibilidade a_k do recurso do tipo k.
- D = data de entrega do projeto.
- H = conjunto de relações de precedência.
- d_{ji} = duração da atividade j quando executada no modo i.
- r_{jik} = quantidade de recursos do tipo k que a atividade j utiliza quando executada no modo i.
- LF_j = instante de término mais tarde que a atividade j pode ser finalizada deve ser menor do que D, considera ainda as relações de precedência entre as atividades, porém, descarta as restrições de recursos. O LF_j é obtido pelo alocação das atividades de forma regressiva, começando da atividade virtual t alocada no instante D. As demais atividades são alocadas iterativamente, sempre que todos os seus sucessores já tenham sido alocados.
- EF_j = instante de término mais cedo que a atividade j pode ser completada, também considera as relações de precedência entre as atividades e descarta as restrições de recursos. O EF_j é obtido pela alocação das atividades de forma progressiva, iniciando da atividade virtual s no tempo inicial e alocando todas suas atividades predecessoras.

objetivo:

$$Minimizar \sum_{k=1}^{m} C_k(a_k) \tag{2.1}$$

sujeito a:

$$\sum_{i=1}^{M_j} \sum_{t=EF_j}^{LF_j} x_{jit} = 1,\ j = 1, ..., n \tag{2.2}$$

$$\sum_{i=1}^{M_h} \sum_{t=EF_h}^{LF_h} t \cdot x_{hit} \leq \sum_{i=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{ji}) \cdot x_{jit}\ j = 1, ..., n,\ \forall h \mid (h, j) \in H \tag{2.3}$$

$$\sum_{j=1}^{n} \sum_{i=1}^{M_j} \sum_{b=1}^{t+d_{ij}-1} r_{jik} x_{jib} \leq a_k\ k = 1, ..., m\ t = 0, \ldots, D \tag{2.4}$$

$$x_{jit} \in \Big\{0, 1\Big\}\ j = 1, ..., n\ t = 1, ..., D\ i = 1, \ldots, M_j \tag{2.5}$$

$$a_k \geq 0,\ a_k \in Z\ k = 1, ..., m \tag{2.6}$$

Na formulação acima, Yamashita descreve o problema com a função objetivo (2.1) que minimiza o custo pela disponibilização dos recursos através da função $C_k(a_k)$ que quantifica o custo de utilização da quantidade a_k do recurso do tipo k. A restrição (2.2) garante que toda atividade só será executada exclusivamente uma vez em um único modo dentro da sua janela de tempo (LF_j, EF_j) permitida. A precedência das atividades é respeitada pela restrição (2.3), a qual garante que toda atividade $j \in (h, j) \in H$, com seu tempo de início $(t - dji)$ deve ser maior ou igual ao término da atividade predecessora h. (2.4) resguarda que durante o tempo de execução das atividades, a quantidade a_k do recurso do tipo k não será excedida e garante que os recursos são do tipo renováveis. As restrições (2.5) e (2.6) definem a variável de decisão binária x_{jit} e que a quantidade a_k seja inteira e positiva.

2.6 Exemplo de dados

Tomando como base o exemplo apresentado na Tabela 2.2, o conjunto H de precedência das atividades seria $H = \{(s, 1), (s, 2), (s, 4), (1, 3), (2, 3), (3, t), (4, t)\}$. A existência do par

ordenado (1, 3) no conjunto H implica que a atividade 1 precede a atividade 3. Vale ressaltar que as atividades s e t são virtuais, que marcam o início e o fim do projeto.

As variáveis d_{ji} que demarcam a duração da atividade j no modo i, e r_{jik} que quantifica a necessidade da atividade j no modo i pelo recurso do tipo k, são demonstradas na Tabela 2.2.

Atividade j		**Modo de execução** i	**Duração** d_{ji}	**Recurso 1** r_{ji1}	**Recurso 2** r_{ji2}	**Recurso 3** r_{ji3}
s	M_s **= 1**	1	0	0	0	0
1	M_1 **= 1**	1	3	2	2	1
2	M_2 **= 1**	1	4	1	2	1
3	M_3 **= 2**	1	2	2	2	3
		2	5	1	0	1
4	M_4 **= 1**	1	2	0	2	0
t	M_t **= 1**	1	0	0	0	0

Tabela 2.2: Exemplo de dados de entrada

A Tabela 2.2 exemplifica os dados de entrada referentes às atividades j que executam nos possíveis modos i, e duram d_{ji} unidades de tempo, também requerem $r_{ji1...3}$ quantidades de recursos. A propriedade de Múltiplos Modos torna-se visível na atividade 3 (j = 3) que pode executar em dois modos diferentes. O primeiro modo (i = 1), gastará 2 unidades de tempo, 2 recursos do tipo 1 e 2, e 3 recursos do tipo 3, no seu segundo modo (i = 2) necessitará de 5 unidades temporárias, 1 recurso do tipo 1 e 3, e não necessitará de recursos do tipo 2. A Tabela 2.2 também demonstra a não influência das atividades virtuais s e t para o projeto, pois, não necessitam de unidade de tempo nem quantidade de recurso.

O custo dessa instância do problema pode ser dado pela função $C_k(a_k) = c_1a_1 + c_2a_2 + c_3a_3$, onde c_1, c_2 e c_3 é o custo de utilização de uma unidade do recurso do tipo c_k, que neste exemplo assumem os valores de 2, 1 e 3 respectivamente.

Ao utilizar a atividade 1 (j = 1), o custo acrescentado ao projeto será calculado como $C_k(a_k) = 2 * 2 + 1 * 2 + 3 * 1$, ou seja, $C_k(a_k) = 9$. Esse custo só será acrescentado totalmente ao projeto se ao executar a atividade 1 ainda não tiver sido disponibilizado nenhum recurso para o projeto, ou se os recursos estiverem sendo utilizados por outra atividade, caso

contrário, eles serão reutilizados pela atividade 1 e serão acrescentadas somente as novas disponibilizações dos novos recursos alocados.

2.6.1 Solução

Uma solução viável para esse problema é a definição do tempo de início de todas as atividades. Porém, uma boa solução almeja a minimização tanto do *makespan*, quanto da quantidade de recursos utilizados no projeto. É possível ver na Figura 2.4 uma solução de boa qualidade para os dados citados acima.

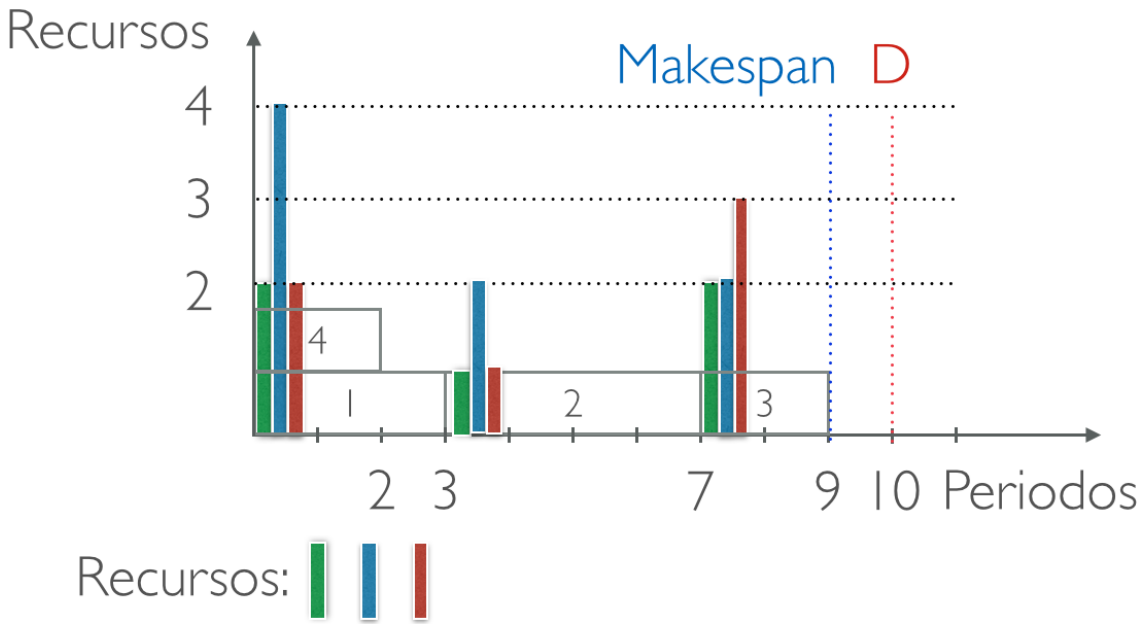

Figura 2.4: Exemplos de solução

Na Figura 2.4 a disponibilidade dos recursos é demarcada pelas barras verdes, azuis e vermelhas que diferem respectivamente os tipos 1, 2 e 3. As quatro atividades reais são mostradas com retângulos enumerados segundo a Tabela 2.2, assim, podemos ver no eixo horizontal do gráfico a marcação temporária do projeto e no eixo vertical são enumerados as disponibilidades dos recursos. A linha pontilhada vermelha demarca a data $D = 10$ de entrega do projeto e a linha pontilhada azul refere-se ao *makespan* $= 9$ da solução.

O custo final de utilização dos recursos são demarcados pelos maiores valores que os 3 tipos de recursos assumiram, que foram na ordem 2, 4 e 3, onde os recursos verde e azul (tipo 1 e tipo 2) assumiram os seus maiores valores no primeiro período com a solicitação

das atividades 1 e 4, já o recurso 3 assume seu maior valor para atender a atividade 3 que executa no seu modo 1 (i = 1). Para contabilizar o custo total do projeto utilizamos a função $C_k(a_k) = 2a_1 + 1a_2 + 3a_3$, onde os valores que multiplicam a_k são os pesos dos respectivos tipos de recursos, $C_k(a_k) = 2 * 2 + 1 * 4 + 3 * 3 = 17$.

Se a formulação de [Yamashita e Morabito 2012] encontrar o valor ótimo para o custo do projeto e não ultrapassar o valor D, será considerada uma solução ótima para o problema. Porém, em projetos reais o tempo influência bastante na escolha do processo, assim, quaisquer alterações que reduzam o tempo, sem elevar o custo, é considerada como a criação de uma nova solução dominante, isso pode ser visto na Figura 2.5, a qual a solução da Figura 2.4 sofreu um deslocamento no instante de início das atividades 4 e 2, reduzindo o *makespan* para 6 (marcado pela linha roxa), uma redução de 3 períodos com permanência do valor do custo de disponibilidade dos recursos.

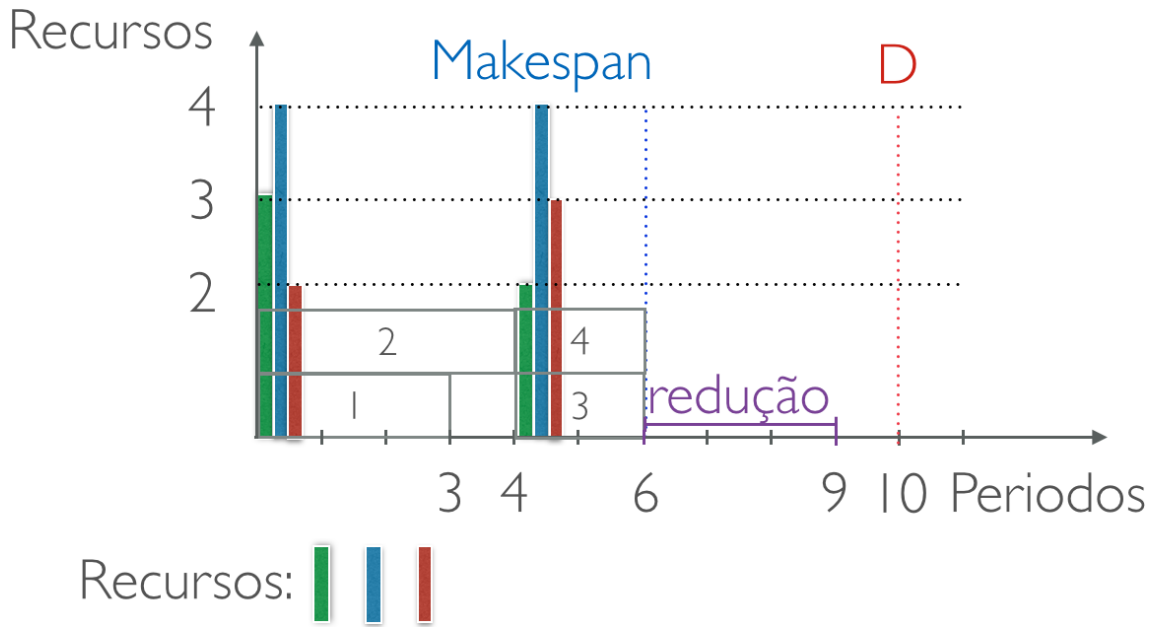

Figura 2.5: Exemplo de solução com redução de *makespan*

Capítulo 3

Trabalhos Relacionados

Este capítulo comenta os principais trabalhos publicados sobre o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos e problemas semelhantes.

3.1 Revisão de Trabalhos com Problemas Semelhantes

Recém formulado na literatura da Pesquisa Operacional (PO), o PCDRMM tem seus primeiros trabalhos datados de 2005, mas como é derivado de outros problemas, é importante entender quais métodos foram aplicados para auxiliar na escolha de uma heurística que construa boas soluções.

Vários trabalhos foram encontrados na literatura propondo resoluções para problemas que tentam minimizar o tempo e o custo dos recursos, a seleção deles foi baseada na semelhança do problema resolvido com o PCDRMM, detalhada a seguir.

Um problema baseado em otimização de projetos com múltiplos modos e objetivos e várias classificações de recursos (renováveis, não renováveis e dupla restrição) foi solucionado por [Slowinki 1981], que apresentou duas abordagens baseadas em programação linear, considerando ainda a interrupção das atividades.

Já [Talbot 1982], mostrou métodos para formular problemas de programação de projetos envolvendo recursos escassos. Para solucioná-los foi utilizado um método com duas etapas, a primeira aplicava heurística e a outra um algoritmo de enumeração para minimizar a duração do projeto. Além disso, o algoritmo também é testado para solucionar um problema com recursos limitados.

Em [Boctor 1990], foi mostrado uma solução para problemas de tamanhos consideráveis, onde a minimização do tempo do projeto depende da alocação dos recursos. Esse trabalho realiza um estudo sobre o desempenho de 21 regras heurísticas e propõe um modelo que aplica regras de prioridade para selecionar os modos das atividades.

Para o problema de recursos limitados com múltiplos modos, [Drexl e Gruenewald 1993] apresentaram um método estocástico com um modelo que generaliza as regras de programação determinística. Além disso, eles também adicionaram uma regra de seleção ponderada das atividades e de seus modos.

Em outro trabalho, [Boctor 1996] descreve um algoritmo que aloca as atividades em cada instante de tempo com base em blocos ou conjuntos de atividades, sua heurística elege o bloco que seja mais favorável para a solução até que todas as atividades tenham um instante de início determinado.

Resoluções com algoritmos genéticos para a Programação de Projetos com Multi-Modos e Recursos Limitados (PPMRL) podem ser vistas em [Mori e Tseng 1997], [Hartmann 2001], [Lova et al. 2009] e [Mendes, Gonçalves e Resende 2009], porém, todos esses trabalhos minimizam apenas o *makespan* e não contabilizam o custo de disponibilidade de recursos como o PCDRMM, pois existe um valor limitante pré-definido da disponibilidade dos recursos, não gerando quaisquer custos para o projeto, independente da adição de qualquer quantidade. Essa é uma das principais características que diferenciam o PCDRMM do PPMRL.

Como já foi descrito no capítulo anterior, o Problema de Custo de Disponibilidade de Recursos (PCDR) tem como principal objetivo minimizar o custo de disponibilidade dos recursos de forma que uma data predeterminada de entrega do projeto seja respeitada. Esse problema foi formulado por [Möhring 1984] para auxiliar na construção de pontes de pequeno porte, ele percebeu que a disputa entre o custo e o tempo está presente nos principais problemas reais de gerenciamento de projetos. [Drexl e Kimms 2001] mostra que uma solução com diversas datas de entrega para o PCDR é uma ótima informação quando é desejável a negociação do prazo e do custo do projeto.

O PCDR teve um algoritmo exato proposto por [Demeulemeester 1995]. Experimentos computacionais com instâncias de projetos de construção de pontes mostraram que esse algoritmo é mais eficiente computacionalmente do que o descrito por Möhring 1984.

Utilizando o método branch-and-bound, [Rangaswamy 1998] conseguiu melhorar a qua-

lidade das soluções encontradas nos outros dois trabalhos sobre o PCDR, entretanto, como os testes foram realizados em plataformas distintas não podemos chegar a uma conclusão sobre a análise do tempo computacional.

O trabalho de [Drexl e Kimms 2001] conseguiu apresentar limites inferiores e superiores através da relaxação lagragiana e um método de geração de colunas. Já [Yamashita, Armentano e Laguna 2006] apresentou uma heurística *multi-start* baseada no método *Scatter Search* (SS), que se mostrou eficiente para instâncias de grande porte. Uma versão utilizando Algoritmos Genéticos foi apresentada por [Shadrokh e F. 2007] com a penalidade por atraso do projeto.

3.2 Estado da Arte do PCDRMM

Ao fim da revisão de alguns trabalhos semelhantes que envolvem custo de disponibilidade e tempo, vamos agora analisar o que já foi proposto para o nosso problema através do que foi encontrado na literatura, que trazem como principais características:

1. A necessidade das atividades por recursos renováveis;
2. A atribuição do instante de início de um conjunto de atividades com suas precedências lógicas;
3. O custo por disponibilizar uma nova quantidade de recursos;
4. A tentativa de reduzir o tempo da última atividade (redução do *makespan*);
5. A capacidade das atividades de serem executadas com distintos modos, os quais são diferenciados pela quantidade de recursos e tempo;

Nos trabalhos encontrados, a característica (4) é substituída por uma data final de entrega do projeto, pois é atribuída a um decisor (gestor do projeto) a responsabilidade da escolha entre as soluções geradas pelo algoritmo que respeitam a data final de entrega do projeto. Uma avaliação entre o custo e o tempo é feita para escolher entre perder em tempo e ganhar nos custos do projeto, ou adiantar o prazo de entrega e acrescentar um custo.

Essa relação de "perde-ganha" é denominada *trade-off* e está muito presente na logística e na economia. Alguns dos trabalhos abaixo utilizam desse dilema como ponto para solucionar o PCDRMM, e nós também utilizaremos *trade-off* como solução para a escolha das melhores respostas do nosso algoritmo.

O primeiro trabalho encontrado na literatura [Hsu e Kim 2005], que une as 5 principais características citadas acima, trata-se de uma adaptação do PCDR acrescentando a característica (5), na qual as atividades podem assumir vários modos de execução. Essa adaptação é nomeada de *Multi-Mode Resource Investment Problem (MMRIP)*.

Além disso, é proposta uma regra de decisão que considera restrições de data de vencimento (*due date*) simultaneamente. O uso de recursos foi proposto para selecionar e agendar tarefas. Nesse trabalho a característica (4) tenta ser almejada, mas apenas com uma data de entrega do projeto já pré definida sem a utilização do *trade-off*, ou seja, duas soluções com mesmo custo e diferentes datas, podem ser consideradas ótimas.

[Chen, Li e Cai 2012] demostrou que o trabalho de [Hsu e Kim 2005] foi o primeiro da literatura a tratar o PCDR com atividades que possuem múltiplos modos, sendo o primeiro a idealizar o PCDRMM. Chen ainda propôs um método heurístico de enumeração baseada em árvore de precedência que seleciona a ordem em que as atividades serão executadas, logo em seguida são realizados três passos para alcançar os objetivos: a seleção das atividades, atribuição de seus modos e a decisão de seus tempos de início.

Testes computacionais foram realizados utilizando 469 instâncias do PPMRL com 10 e 20 atividades, a motivação apresentada é o aluguel de recursos computacionais na nuvem para processamento, uma vez que essa alocação é paga apenas uma vez, caracterizando os recursos renováveis.

O terceiro trabalho [Yamashita e Morabito 2012], traz uma formulação mais concreta, já não trata o problema com uma adaptação do PCDR, mas como uma nova modelagem, por isso, podemos citar Yamashita e Morabito 2012 como o trabalho que modelou o PCDRMM de maneira íntegra. Neste trabalho também foram criados novos problemas testes pelo programa Progen, bastante conhecido na literatura.

[Yamashita e Morabito 2012] foi o primeiro trabalho da literatura a nomear o *Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos*, e a propor um algoritmo exato que utiliza *Brach-and-Bound* para solucionar e comparar seus resultados com a for-

mulação matemática apresentada na seção 2.5, implementada pelo aplicativo CPLEX. Sua comparação mostrou que o algoritmo proposto supera pelo tempo computacional a implementação do modelo. Além disso, ainda é demonstrada a prova de correção do algoritmo, comprovando que as soluções da sua heurística são ótimas.

Percebe-se que os trabalhos de [Yamashita e Morabito 2012] e [Chen, Li e Cai 2012] foram publicados no mesmo período, isso impossibilita uma melhor comparação entre eles, pois, além de não compartilhar das mesmas plataformas computacionais, não utilizam as mesmas instâncias. Mas, Yamashita publicou uma nota em [Yamashita e Morabito 2009], onde aborda apenas suas ideias iniciais do algoritmo que foi apresentado mais tarde.

Ainda é possível encontrar na literatura uma adaptação do PCDR com a particularidade (5), que atribui às atividades diferentes modos de execução. Com isso, o PCDR torna-se ainda mais complexo e idêntico ao problema abordado por este trabalho também chamado de *Multi-Mode Resource Avaliability Cost Problem* (MRACP).

Na busca pelo MRACP foi possível encontrar dois trabalhos na literatura atual:

O primeiro, [Qi et al. 2015], utiliza uma heurística de Otimização por Enxame de Partículas (*Particle Swarm Optimization*), que foi introduzida por [Kennedy e Eberhart 1995] para resolver problemas no domínio contínuo. Essa heurística foi inspirada por algoritmos que modelam o comportamento social observado em muitas espécies de pássaros, cardumes de peixes e também na sociedade humana.

Por fim, uma adaptação do PCDRMM é encontrada em [Nadjafi 2014] com o nome de *Multi-mode Resource Availability Cost Problem with Recruitment and Release dates for resources* (MRACP-RR), onde o custo de disponibilização dos recursos é contabilizado apenas pelo período entre a primeira utilização do recurso e a última, tornando o problema aplicável a problemas reais, nos quais o aluguel de produtos é cobrado apenas pelo tempo em que está disponível para utilização (tempo de contratação) e não por toda a duração do projeto.

Esse problema se aplica bastante em projetos de construção civil, onde os aluguéis de equipes de funcionários especializadas ou grandes máquinas (recursos), são contabilizados apenas pelo período que o recurso estiver disponível na obra.

Nadjafi utiliza a heurística *Simulated Annealing* (SA) para construir soluções do MRACP-RR de boa qualidade. Como não existe registro antes desse trabalho do problema adaptado, Nadjafi realiza testes computacionais com o modelo matemático implementado na

plataforma Lingo, e seu algoritmo se mostra eficaz para instâncias com muitas atividades.

O MRACP-RR foi inspirado no MRACP, portanto, herdou a característica de apenas uma data para entrega final do projeto. Nadjafi utiliza datas espaçosas para suas instâncias e calcula o prazo de entrega do projeto com a melhor solução possível para os custos. Esse valor é obtido pela alocação de todas as atividades no modo que utiliza menos recursos, desconsiderando a minimização do *makespan*.

A tabela abaixo, mostra um comparativo das características principais dos trabalhos que abordam o PCDRMM e variantes:

Trabalhos	**Heurística Utilizada**	**Ótimas**	**Custo por Recurso**	**Intâncias de Teste**
Hsu e Kim 2005	Heurística de Enumeração	Não	Por todo o projeto	Não informado
Chen, Li e Cai 2012	Regras de Prioridade	Não	Por todo o projeto	469 instâncias de 10 e 20 atividades
Yamashita e Morabito 2012	*Brach-and-Bound*	Sim	Por todo o projeto	12 instâncias de 10 atividades
Nadjafi 2014	*Simulated Annealing*	Não	Por unidade de tempo	300 instâncias de 20 a 90 atividades
Qi et al. 2015	*Particle Swarm Optimization*	Não	Por todo o projeto	180 instâncias de 10 a 30 atividades.

Tabela 3.1: Comparação dos trabalhos encontrados

Nesta tabela o trabalho de [Yamashita e Morabito 2012] apresenta-se como o único a propor uma heurística que garante soluções ótimas. Já [Chen, Li e Cai 2012], manifesta sua quantidade de instâncias distintas, porém com um conjunto limite de até 20 atividades, enquanto [Nadjafi 2014] mostrou ser eficaz em instância de até 90 atividades, além de a sua adaptação do MRACP ser o único trabalho que contabiliza o custo apenas nas unidades de tempo que o recurso estará disponível para o projeto. É destacável também a sua preocupação em liberar o prazo de entrega do projeto para uma decisão com mais abrangências nos valores de tempo e custo (decisão de *trade-off*).

Capítulo 4

Métodos Propostos

Neste capítulo é proposto uma nova construção para o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos baseada em Matheurísticas criadas pela junção de Meta-heurísticas hibridizadas com uma etapa de Programação Matemática. A metodologia para alcançar os objetivos deste trabalho se divide em três partes: pesquisa de trabalhos relacionados; análises e seleção das heurísticas para a solução; codificação do algoritmo e testes computacionais.

A pesquisa de trabalhos sobre problemas semelhantes se mostrou satisfatória ao mapear as principais metodologias utilizadas na literatura como visto no Capitulo 3. Vários dos trabalhos encontrados, solucionam o problema com apenas um método heurístico ou exato, alguns também trazem uma etapa de refinamento de sua melhores soluções, assim, a junção dessas abordagens se apresenta como uma boa proposta para solucionar problemas de grande porte que possuam a complexidade assinalada.

4.1 Etapas do Algoritmo Genético

A etapa de codificação do algoritmo teve como ponto de partida a criação do Algoritmo Genético (AG), que se destacou ao apresentar uma construção favorável em [Mendes, Gonçalves e Resende 2009] para a Programação de Projetos com Multi-Modos e Recursos Limitados (PPMRL). Ele inicia a criação das soluções iniciais com a escolha do modo M_j para toda atividade j como um cromossomo, mutações e cruzamentos são efetuados entre os indivíduos da população de soluções iniciais, até alcançar uma solução aceitável em seus

parâmetros de avaliação. Posteriormente, as soluções são refinadas utilizando uma heurística de busca local descrita por [Klein 2000] para melhorar seus resultados obtidos.

Nosso algoritmo seguirá a sequencia de atividades do Algoritmo Genético, porém, são necessárias algumas alterações nas suas etapas para a adaptação e construção das soluções do PCDRMM, como pode ser visto na Figura 4.1 abaixo:.

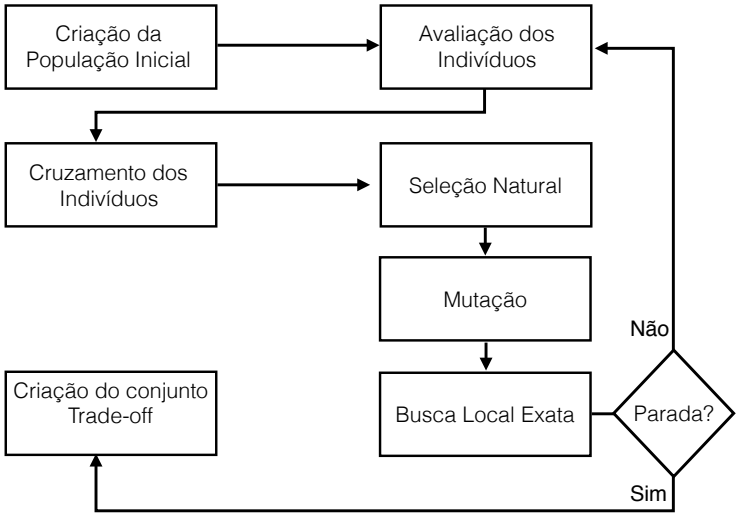

Figura 4.1: Fluxograma da Construção

4.1.1 População Inicial

A construção da nossa população inicial, tem como objetivo a diversificação de indivíduos, para possibilitar o busca por pontos distintos no espaço de soluções factíveis. Espera-se que ocorra uma melhora na qualidade da solução final da mesma forma que foi demonstrado por [Lima, Costa e Ochi 2003], onde algoritmos que partem de soluções distintas podem convergir para bons pontos no espaço de soluções.

Com essa realidade três heurísticas de construções iniciais foram propostas e detalhadas a seguir:

1. Melhor *makespan*: são selecionados os modos das atividades com a menor duração do tempo de execução, desconsiderando a utilização dos recursos. A escolha do tempo de início das atividades é exatamente o instante de término da sua última predecessora $EF_j - min(d_{ji})$. EF_j é o instante de término mais cedo que a atividade j pode ser

completada, d_{ji} duração da atividade j no modo i. Dessa forma, garantimos que a solução terá o melhor *makespan*, porém uma péssima utilização dos recursos. Compreenderemos um pouco mais com o pseudocódigo abaixo:

Algoritmo 1: Criação da solução com melhor *makespan*

Entrada: Conjunto de atividades: *J*, Conjunto de modos de j : M_j
Saída: Solução: s
Dados: Tempo de início de j: t_j, Modo da atividade j: $modo_j$
início
  para todo $j \in J$ **faça**
    $ti_j = EF_j$
    $modo_j = minTempo(M_j)$
    $s \leftarrow [j, ti_j, modo_j]$
  fim
  retorna s
fim

2. Modos aleatório: os modos das atividades são selecionados de maneira aleatória e são alocadas de acordo com sua ordem de prioridade (precedência), ou seja, são alocadas as atividades sucessoras de s (atividade fictícia que demarca o início do projeto), depois serão alocadas suas sucessoras até que a solução esteja completa. O tempo de início é atribuído da mesma maneira que a construção (1) $EF_j - min(d_{ji})$ como pode ser visto abaixo:.

Algoritmo 2: Criação da solução com modos aleatórios

Entrada: Atividades: *J*, Modos de j : M_j, Sucessores de j: H_j
Saída: Solução: s
Dados: Atividades já alocadas: atAloc
início

$atAloc \leftarrow [1]$

Aloca a atividade 1 no tempo 0 com o modo 0

$s \leftarrow [1, 0, 0]$

para todo $j \in J$ **faça**

para todo $h \in H_j$ **faça**

se $h \notin atAloc$ **então**

$ti_h = EF_h$

$modo_h = minTempo(M_h)$

$s \leftarrow [h, ti_h, modo_h]$

$atAloc \leftarrow [h]$

fim

fim

fim

retorna s

fim

3. Melhor modo por custo: a escolha do modo m_{ij} da atividade j é realizada verificando o custo de utilização dos recursos do modo i, isso possibilita uma tentativa de redução do custo do projeto, no entanto, ao desconsiderar os tempos das atividades, criaremos uma solução com *makespan* elevado. Essa construção é semelhante ao Algoritmo 2 modificando apenas a função $modo_h = minCusto(M_h)$, onde ela calcula o custo de cada modo individualmente e escolhe o menor, de acordo com o custo a ser processado.

A construção (1) pode ser considerada o limite inferior para as soluções em relação à minimização do *makespan*. Uma vez que o instante de término mais cedo de uma atividade (respeitando suas precedências) é dado por EF_j, e com a verificação do modo m_{ij} da atividade j que tem a menor duração $min(d_{ji})$, podemos afirmar que o tempo de início mais cedo de uma atividade é dada por $EF_j - min(d_{ji})$. Portanto, a alocação de todas as atividades

no menor tempo de início e o modo com a menor duração garante uma solução factível com melhor *makespan* possível.

Partindo dessas construções, teremos nossa população inicial P_1 com duas soluções criadas pelas construções (Melhor *makespan* 1) e (Melhor modo por custo 3), e todos os outros indivíduos criados pela construção (Modos aleatório 2) para uma boa diversificação dos cromossomos, tanto pela otimização do tempo quando pela minimização dos custos, isso proporcionou uma redução no número de gerações necessárias para o critério de parada e um redução na quantidade das soluções necessárias que compõem a população inicial P_1.

4.1.2 Avaliação

O cromossomo ou individuo do Algoritmo Genético, é um escalonamento que respeita todas as restrições mostradas na Seção 2.5 Formulação Matemática, o qual pode ser avaliado tanto pelo seu tempo *makespan*, quanto pelo seu custo de utilização dos recursos $C_k(a_k)$. A minimização desses valores são as metas a serem alcançadas, porém, a função objetivo da modelagem matemática do problema, tenta minimizar apenas o custo, deixando o tempo como uma restrição complementar do problema.

Essa falha é corrigida na etapa de avaliação dos indivíduos pela função *fitness*, que quantifica uma solução s como $f(s) = p_t * t + p_c * c$, onde p_t é a proporção do tempo final da solução t, e p_c é a proporção do custo por disponibilidade c. A utilização de $p_t + p_c = 1$ balanceia a necessidade do custo e o tempo da solução, serve também de parâmetros de ajuste do algoritmo de acordo com a necessidade do decisor.

Para construir uma solução totalmente equilibrada entre o tempo e o custo dos recursos, adotaremos que os dois valores devem ser $0,5$, ou seja, uma solução com a mesma prioridade para o tempo e o custo. Casos em que o tempo seja uma necessidade, o valor de p_t deve ser maior do que o custo. Para melhorar a avaliação dos indivíduos os valores de t e c são normalizados, uma vez que são grandezas distintas.

A função de avaliação ainda atribui uma penalidade à solução que não cumprir o requisito da data final do projeto D, porém pode gerar filhos que respeitem essa data ao realizar o cruzamento com outro indivíduo. Essa função objetivo guiará o Algoritmo Genético para uma solução de boa qualidade e eliminará os indivíduos com os menores valores da população.

4.1.3 Cruzamento

O cruzamento das soluções assume um papel fundamental no Algoritmo Genético, pois é nessa etapa que acontece a evolução da população para uma nova geração. O algoritmo seleciona duas soluções distintas e mapeia cada par ordenamo (atividade, modo de execução) como um genes a ser doado, logo em seguida o cruzamento aleatório desses genes é aplicado. O Algoritmo 4.1.3 mostra a execução do cruzamento de duas soluções.

Algoritmo 3: Cruzamento de duas soluções

Entrada: Solução s1, Solução s2
Saída: Solução: *s'*
Dados: Nova Solução *s'*, Tempo de início de j: ti_j
início
  Aloca a atividade 1 no tempo 0 com o modo 0
  $s' \leftarrow [1, 0, 0]$
  para todo $j \in J$ **faça**
    Solução $s_{eleita} = EscolhaAleatoriamenteUmaSolucao(s1, s2)$
    $modo_j = Modo(j, s_{eleita})$
    se $ti_j \in s_{eleita} > (EF_j - d_{modo_j}) \in s'$ **então**
      $ti_j = ti_j \in s_{eleita}$
    fim
    senão
      $ti_j = (EF_j - d_{modo_j}) \in s'$
    fim
    $s' \leftarrow [j, ti_j, modo_j]$
  fim
  retorna s'
fim

A função $EscolhaAleatoriamenteUmaSolucao(s1, s2)$ elege uma entre as duas soluções $s1$ e $s2$, com a mesma probabilidade. Essa etapa define quem doará o genes para a atividade j. O Algoritmo 4.1.3 ainda tenta atribuir o tempo de início da atividade ti_j encontrada da solução eleita s_{eleita}, isso só acontecerá se o menor tempo de alocação possível ($EF_j - d_{modo_j} \in s'$) for maior do que o tempo de início de j na solução eleita ($ti_j \in s_{eleita}$),

caso contrário a atividade é alocada em $EF_j - d_{modo_j} \in s'$, sendo s' o escalonamento atual da nova solução.

Esse processo é repetido em toda geração com 50% dos indivíduos da população atual, e os filhos gerados por esse processo são adicionados ao conjunto de soluções do algoritmo. Se escalonamentos idênticos forem criados, não serão adicionados a população. Uma prioridade conhecida como elitismo é dada aos melhores indivíduos da nova população para disseminarem e proporcionar melhorias em outras soluções cruzadas.

4.1.4 Seleção

O processo de seleção natural dos indivíduos criados em cada geração, é apenas uma exclusão das piores soluções da população conhecida. Essa eliminação ocorre para que o conjunto das soluções factíveis continue com um número constante $\textit{população}_{size}$ ao final de cada geração, vários testes foram aplicados para fixar um número ideal para essa população, os quais demonstraram que quanto maior essa constante, mais tempo o algoritmo necessitará para a conclusão das etapas em cada geração.

Ao fim desse processo, todas as soluções geradas que não conseguiram se posicionar entre as melhores da população, serão eliminadas e não participaram de cruzamentos e avaliações futuras, para que o tamanho da população seja sempre igual a $\textit{população}_{size}$ no final desta etapa. Isso permite que a população P_g de indivíduos sempre evolua de uma geração a outra ($P_{g+1} \geq P_g$) em seus parâmeros de avaliação (custo e tempo).

4.1.5 Mutação

A etapa de mutação do Algoritmo Genético é inspirada nas alterações sofridas pelos indivíduos ao longo do processo evolucionário, sendo elas benéficas ou maléficas. No nosso algoritmo são selecionados a cada geração 20% dos indivíduos de forma aleatória para sofrerem alterações nos seus genes (atividade, modo de execução).

O gene que será alterado na solução é selecionado de forma aleatória entre as atividades que não estejam alocadas em seu tempo máximo de termino $t_j < EF_j$, e a modificação é feita alterando o modo de execução daquela atividade, mas para isso ser possível, o novo tempo de término da atividade com o novo modo alterado pela mutação não pode extrapolar

os limites de suas sucessoras $t'_j \leq EF_j$. Se essa modificação não for possível, o processo de mutação não é aplicado a essa solução.

4.1.6 Busca Local Exata com a Matheurística

Nessa etapa o algoritmo seleciona 20% das soluções da população para serem alteradas, sendo que 10% são constituídos pelos melhores indivíduos conhecidos, e os outros 10% são eleitos de forma aleatória, esse processo apenas modifica a solução se for possível a minimização nos valores de tempo ou custo.

Essa busca local é feita utilizando a modelagem matemático descrita na Seção 2.5 e implementação pelo *framework CPLEX* da *IBM* que utiliza o método *Simplex* para auxiliar na programação matemática. São aplicados algumas alterações no modelo original para possibilitar a otimização de apenas parte da solução em um tempo computacional aceitável.

A adaptação feita nessa etapa adiciona ao modelo original do PCDRMM o sub-conjunto de atividades fixas F que não serão alteradas pelo modelo, o qual é formado pelas atividades de um escalonamento viável $F = \{\forall j \in s' \mid t'_j - d'_{ji} \notin a_{kt}\}$, onde t'_j é o tempo em que a atividade j foi escalonada, d'_{ji} é a duração da atividade j no modo i, e a_{kt} é o conjunto de intervalos de tempo em que a demanda a_k do recurso do tipo k foi alcançada.

O conjunto F é formado pelas atividades j que não alocaram novos recursos para sua conclusão, assim as atividades que serão alteradas pelo novo modelo, são consideradas críticas para a solução em relação ao custo. O conjunto F deve conter mais de 60% no número total de atividade, caso isso não ocorra, novas atividades são escolhidas de forma aleatória para que a solução criada s'' não seja descaracterizada da solução original s'. Todas as outras variáveis descritas na Seção 2.5 Formulação Matemática, também são utilizadas nessa adaptação e os novos conjuntos criados são:

- s' = escalonamento viável com todas as atividades j alocadas no tempo t no modo i.
- a_{kt} = conjunto de intervalos de tempo t que foram utilizados toda a demanda do recurso do tipo k.
- F = conjunto de atividades de uma solução s' que satisfazem a condição $t'_j - d'_{ji} \notin a_{kt}$.

$$Minimizar \sum_{k=1}^{m} C_k(a_k) \tag{4.1}$$

sujeito a:

$$\sum_{i=1}^{M_j} \sum_{t=EF_j}^{LF_j} x_{jit} = 1,\ j = 1, ..., n,\ \forall j \notin F \tag{4.2}$$

$$x_{jit} = 1,\ \forall j \in F,\ \forall j, i, t \in s' \tag{4.3}$$

$$\sum_{i=1}^{M_h} \sum_{t=EF_h}^{LF_h} t \cdot x_{hit} \leq \sum_{i=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{ji}) \cdot x_{jit}\ j = 1, ..., n, ,\ \forall h \mid (h, j) \in H \tag{4.4}$$

$$\sum_{j=1}^{n} \sum_{i=1}^{M_j} \sum_{b=1}^{t+d_{ij}-1} r_{jik} x_{jib} \leq a_k\ k = 1, ..., m\ t = 0, \ldots, D \tag{4.5}$$

$$x_{jit} \in \Big\{0, 1\Big\}\ j = 1, ..., n\ t = 1, ..., D\ i = 1, \ldots, M_j \tag{4.6}$$

$$a_k \geq 0,\ a_k \in Z\ k = 1, ..., m \tag{4.7}$$

A descrição acima, modifica a restrição (2.2) do modelo original removendo as atividades pertencente ao conjunto F, e adiciona a restrição (4.4) que fixa o tempo t o modo i da atividade $j \in F$ na solução s'. Todas as outras restrições permaneceram idênticas ao modelo original.

4.1.7 Criação do *Trade-off*

Ao final da busca local exata, as soluções escolhidas para serem modificadas pelo novo modelo, são reavaliadas e inseridas na população para poderem participar das gerações seguintes, logo apos essa inserção, uma verificação de quantas gerações já ocorreram é efetivada, e caso o número de gerações não tenha sido o suficiente, o algoritmo retorna ao processo de avaliação e ordenação dos indivíduos, caso contrário o algoritmo entra na faze final para a criação da curva de *trade-off*, baseada na heurística da Fronteira de Pareto, onde uma solução s^1 é dita dominante da solução s^2, se ambas as condições a seguir forem satisfeita:

- A solução s^1 não é pior do que a solução s^2 em nenhuma das grandeza (custo e tempo), ou seja, $custo_{s^1} \leq custo_{s^2}$ e $tempo_{s^1} \leq tempo_{s^2}$.

- A solução s^1 é melhor do que a solução s^2 em no mínimo uma grandeza, ou seja, $custo_{s^1} < custo_{s^2}$ ou $tempo_{s^1} < tempo_{s^2}$.

A população final serão indivíduos as quais suas dominantes não foram conhecidas pelo algoritmo. Esse processo entrega o conjunto de soluções que formarão o *trade-off* para o decisor como mostrada na Figura 4.2.

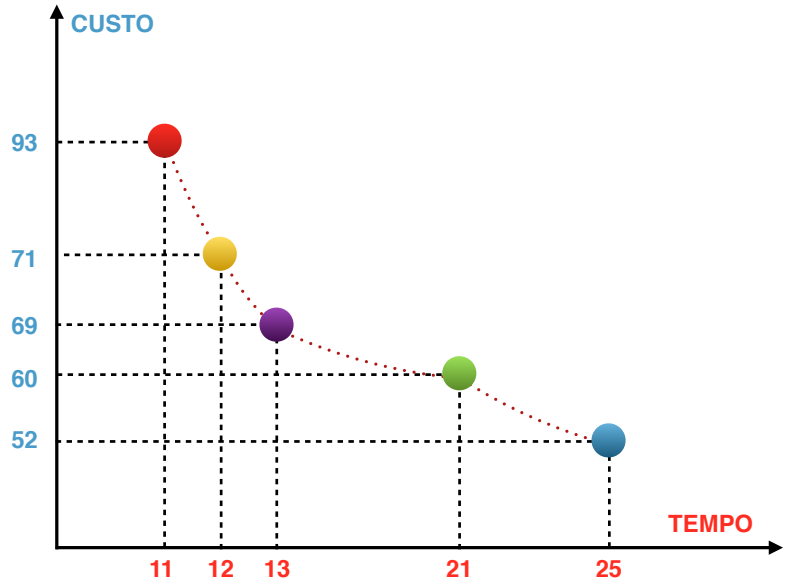

Figura 4.2: Exemplo de *trade-off* criado pelo Algoritmo

4.2 Otimização por Enxame de Partícula

Outra Matheurística desenvolvida neste trabalho, foi a Otimização por Enxame de Partícula (PSO) motivada por [Qi et al. 2015], que também construiu uma solução para o PCDRMM utilizando o PSO e consegui bons resultados em relação a implementação do modelo.

A Otimização por Enxame de Partícula foi apresentada por [Kennedy e Eberhart 1995] com base no comportamento de pássaros e outros animais que andam em grupo, onde foi observado que existe um padrão para a movimentação dos animais para obter sua nova direção.

No PSO cada indivíduo (representado por uma solução) de uma população (enxame), quantificam suas experiências locais (vizinhos locais) e ideais do enxame (melhores globais), assim, dá-se o aprendizado individual de cada partícula do enxame. Um pseudocódigo do PSO é apresentado de forma resumida abaixo:

Algoritmo 4: Exemplo de PSO

Dados: Conjunto de partículas: A, Melhores locais: $pbest$, Melhor global: $gbest$, Épocas: E

início

$A \leftarrow IniciaEnxameDeParticulas()$

para todo $e \leftarrow 1$ *até* F **faça**

$Atualizar(pbest, gbest, A)$

para todo $a \in A$ **faça**

$AvaliarParticula(a)$

$A \leftarrow MoverParticula(a, e)$

fim

fim

fim

As posições das partículas e velocidades são conduzidas por sua própria experiência e pela observação da sua vizinhança. Elas se movem no espaço de busca de acordo com as equações a seguir:

$$v_{i(e+1)} = w.v_{ie} + c_1.r_1(pbest_{ie} - x_{ie}) + c_2.r_2(gbest_e - x_{ie}) \tag{4.8}$$

$$x_{i(e+1)} = x_{ie} + v_{i(e+1)} \tag{4.9}$$

$$e = 1, ..., E \tag{4.10}$$

$$i = 1, ..., m \tag{4.11}$$

Onde v_{ie} é a velocidade da partícula i na época e, w representa a inércia da direção que estava sendo seguida, r_1 e $r_2 \in [0, 1]$ são constantes aleatórias, c_1 e $c2$ são os coeficientes

que representam a aceleração cognitiva que levam os indivíduos em direção ao $pbest$ e ao $gbest$ respectivamente, $pbest$ é um vetor com as coordenadas das melhores partículas do conjunto de vizinhos locais, e $gbest$ representa a melhor coordenada global entre todas já visitadas pelo enxame. Já x_{ie} representa a posição do indivíduo i na época e, m é o número de partículas do enxame A.

A representação da solução do PCDRMM é dada por dois vetores: $t_j = (t_1, t_2, ..., t_n)$, que demarca o tempo de inicio t_j de cada atividade j e $m_j = (m_1, m_2, ..., m_n)$, que enumera o modo m_j de todas as n atividades. Porém, para mapear a solução como uma partícula do PSO, o vetor x é concatenado para $x = t_j \cup m_j$, que terá como elementos $x_i = (q_1, q_2, ..., q_n, q_{n+1}, q_{n+2}, ..., q_{2n})$, onde os primeiros n números $(q_1, q_2, ..., q_n)$ representam os tempos de início das atividades e, logo em seguida $(q_{n+1}, q_{n+2}, ..., q_{2n})$ os modos são apresentados: $x_i = (t_1, t_2, ..., t_n, m_1, m_2, ..., m_n)$.

Como uma solução do PCDRMM não pode ser representada no domínio contínuo, e o PSO gera soluções com essas características, adotamos o maior número inteiro $\lceil x \rceil$ para as variáveis contínuas.

Utilizando essa ideia, Qi et al. 2015 adaptou instâncias de outro problema para comparar sua heurística com um Algoritmo Genético aplicado ao RACP, os resultados são melhorados quando adiciona a busca local *Scatter Search* ao PSO.

O PSO se mostrou no trabalho de [Qi et al. 2015] como uma boa heurística para solucionar o PCDRMM, onde consegue boas soluções em um tempo computacional aceitável para as instâncias. Qi demostra que a sua heurística construiu boas soluções para o problema, porém, a adição da *Scatter Search* retarda o tempo computacional.

Na implementação do PSO feita para este trabalho a etapa de Busca Local Exata será idêntica ao apresentado na Seção 4.1.6, isso também servirá para uma comparação mais justa das duas Matheurísticas.

Capítulo 5

Testes Computacionais

Neste capítulo apresentaremos os resultados encontrados para os algoritmos implementados para o Problema de Custo de Disponibilidade de Recursos com Mútilos Modos.

5.1 Características das instâncias

Algumas das instâncias testes foram disponibilizadas por Yamashita e Morabito 2012, nas quais todas as atividades possuem 3 modos de execução distintos, 4 tipos de recursos disponíveis e 12 atividades, sendo a primeira e a última atividades virtuais, com tempo e quantidade de recursos nulas.

Uma adaptação foi necessário para utilizar os dados de entrada do problema *Multi-Mode Resource Constrained Project Scheduling Problem (MRCPSP)* disponível na biblioteca PSPLIB. Neste problema existe a inclusão dos recursos não renováveis, porém para o PCDRMM foi necessário considerar todos os recursos como renováveis. Esses dados, são diferenciados pela quantidade de atividades com 10, 20 e 30 identificadas como (J10), (J20) e (J30) respectivamente. Todas as atividades possuem 3 modos de execução distintos e 4 tipos de recursos disponíveis.

As instâncias do PSPLIB para o MRCPSP, não atribuem custos aos diferentes tipos de recursos, por isso surgiu a necessário da criação de uma função que quantifique a utilização dos recursos k, ao analisar sua utilização nos modos de todas as atividades, tratando um recurso que seja bastante requerido pelas atividades com um valor menor, e um tipo de recurso que é mais escarço na instância como um custo maior de utilização. O código fonte

desta função pode ser visto o Apêndice B.

Inspirado por Nadjafi 2014, o calculo da data limite é baseado na heurística de construção de soluções (3), a qual é obtida pela alocação sistemática dos melhores modos ao analisar a quantidade de utilização de recursos, essa decisão é justificada pela melhor variedade de soluções criadas para que o decisor tenha mais opções no *trade-off*.

5.2 Resultados

O modelo matemático e sua adaptação descritos nas seções 2.5 e 4.1.6, foram implementados no software *ILOG CPLEX* disponibilizado pela *IBM* através da Iniciativa Acadêmica. Todo código deste trabalho foi desenvolvido na linguagem de programação C++, e os testes executados em um computador com processador Intel Core i5 2,3 GHz, com 4 GB de memória Ram.

Nas tabelas abaixo, o custo das soluções são apresentados pelas colunas com o "(C)", e o tempo de execução que é dado em segundos pode ser encontrado nas colunas demarcadas com "(T)", as Matheurísticas são identificadas pela adição da letra "M"(AGM e PSOM). O Algoritmo Genético (AG) necessitou de 22 gerações para alcançar seu objetivo assim como o PSO utilizou 50 épocas, e cada experimento realizou uma bateria de testes com 50 execuções de cada algoritmo para chegar nos resultados abaixo. Todos os parâmetros de configuração do Algoritmo Genético (tamanho da população, número de gerações, porcentagem de mutação e cruzamento) foram obtidos pela plataforma de configuração iRace, com 50 interações em 50 instâncias de treinamento excluídas dos testes finais.

A Tabela 5.1 demonstra os melhores resultados encontrados sem a etapa da Mathéuristica, e foram obtidos por uma amostra de 20 instâncias da PSPLIB com 10 atividades (J10).

Tabela 5.1: Comparação entre AG e o PSO, sem a Matheurística (J10)

Instâncias	**AG (C)**	**PSO (C)**	**AG (T)**	**PSO(T)**
1	65,42	85,86	0,91	1,1
2	79,71	102,44	0,6	0,98
3	85,79	103,33	0,63	0,94
4	69,94	77,53	0,8	1,3
5	78,88	95,31	0,7	1,01
6	95,12	116,31	0,77	1,24
7	117,28	138,02	0,62	0,89
8	62,75	68,88	0,49	0,84
9	82,17	100,38	0,52	0,84
10	68,89	86,46	0,43	0,64
11	72,53	90,46	0,67	1,06
12	80,31	95,55	0,85	1,28
13	95,24	105,92	0,57	0,85
14	111,07	132,46	0,59	0,75
15	62,38	81,96	0,69	0,96
16	90,82	100,36	0,53	0,85
17	91,64	112,83	0,57	0,88
18	75,41	80,46	0,67	0,96
19	88,94	115,68	0,42	0,73
20	74,88	93,4	0,73	1,12
Total	*1583,75*	1897,74	*11,85*	18,12

Na comparação do AG com o PSO, pode-se observar que uma redução de 16% no custo (C), e 34% no tempo de execução (T), comprova que sem a utilização da Matheurística as melhores soluções em menor tempo são alcançadas pelo AG.

Para alcançar os resultados da Tabela 5.2, a etapa da Matheurística foi inserida nos algoritmos que proporcionou uma melhora de 18,12% para o Algoritmo Genético AGM e 19,34% no PSOM dos teste da Tabela 5.1.

Tabela 5.2: Comparação entre AGM e o PSOM

Instâncias	**AGM (C)**	**PSOM (C)**	**AGM (T)**	**PSOM (T)**
1	55,98	58,73	1,37	2,13
2	66,03	77,23	1,6	4,18
3	77,51	103,6	1,71	2,67
4	63,17	63,17	1,91	3,13
5	61,95	66,74	1,3	2,45
6	75,35	84,89	1,96	2,34
7	83,29	91,87	1,32	2,02
8	51,20	71,8	1,31	2,23
9	68,54	71,43	1,47	2,91
10	56,45	91,34	1,15	1,87
11	63,19	74,12	1,41	3,56
12	60,30	92,34	2,07	2,98
13	73,21	91,4	1,44	3,15
14	85,61	94,1	1,04	4,03
15	51,39	76,51	1,92	2,14
16	73,96	81,34	1,2	1,94
17	82,49	86,54	1,25	2,65
18	54,75	62,45	1,52	2,9
19	79,44	82,45	0,95	3,12
20	57,02	68,1	1,95	2,97
Total	1340,83	1590,15	29,85	55,37

Estes testes provam que a adição da etapa da Matheurística traz um ganho significativo para o dois algoritmos em relação ao custo, porém mesmo com essa redução o Algoritmo Genético continua conseguindo melhores resultados para o problema.

A aleatoriedade está presente em algumas etapas do Algoritmo Genético proporcionando diferentes resultados ao final de testes distintos. A Tabela 5.3 faz um estudo sobre esses resultados descriminando alguns valores estatísticos como a média, a moda, o pior e o melhor caso de uma amostra de 50 execuções nas instâncias do PSPLIB com 10 atividades (J10).

Tabela 5.3: Aleatoriedade do AGM

Instâncias	Melhor (C)	Média (C)	Moda (C)	Pior (C)	Melhor (T)	Média (T)	Pior (T)
1	55,98	60,08	57,3316	71,05	1,37	3,08	6,46
2	66,03	74,04	72,1185	93,15	1,6	2,74	6,34
3	77,51	81,7	82,6954	85,25	1,88	5,02	15,63
4	63,17	65,64	63,178	71,77	1,91	3,55	9,55
5	61,95	67,74	61,9524	81,99	1,3	2,49	6,64
6	75,35	81,86	78,3621	90,3	1,96	3,31	11,04
7	83,29	91,91	83,2975	100,76	1,32	2,59	5,7
8	51,2	56,8	51,2089	66,14	1,31	2,49	6,44
9	68,54	74,14	70,6667	88,61	1,47	2,63	4,87
10	56,45	62,7	64,2284	76,61	1,15	1,86	3,98
11	63,19	70,59	71,6594	78,43	1,41	2,47	5,06
12	60,3	69,12	65,6078	93,07	2,07	3,6	11,28
13	73,21	84,28	75,9741	100,18	1,44	3,63	12,67
14	85,61	96,33	91,4601	103,36	1,04	3,16	10,68
15	51,39	54,84	53,4355	62,46	1,92	2,67	4,68
16	73,96	80,73	76,6105	110,14	1,2	1,76	3,62
17	82,49	87,51	82,4865	96,8	1,25	2,29	5,47
18	54,75	59,25	55,1108	75,37	1,52	3,57	6,25
19	79,44	85,06	81,5866	101,9	0,95	1,69	3,74
20	57,02	61,27	58,1714	71,35	1,95	3,95	10,52
Total	1340,83	1465,59	1397,1422	1718,69	30,02	58,55	150,62

Uma diferença de 28% separam um melhor resultado do pior encontrado pelo AGM, e esse valor cai para 4% quando comparado o melhor resultado com o a moda encontrada.

Uma implementação do modelo matemático utilizando o *CPLEX* servirá de parâmetro para quantificar a qualidade das soluções criadas pelo Algoritmo Genético AGM, esses testes utilizam as instâncias criadas para o PCDRMM disponibilizadas por Yamashita e Morabito 2012.

Tabela 5.4: Comparação entre AGM e o *CPLEX* com instâncias do PCDRMM

Instâncias	**CPLEX (C)**	**AGM (C)**	**CPLEX(T)**	**AGM (T)**
1	38,76	38,96	1,03	1,16
2	117,1	*117,1*	2,86	1,15
3	277,29	285,63	39,4424	1,23
4	376,1	378,31	55,28	3,1
5	93,49	98,1	4,49	1,38
6	121,52	132,36	1,57	1,54
7	105,52	108,96	7,08	1,65
8	148,75	156,67	79,42	2,82
9	21,31	*21,31*	1,73	1,12
10	172,07	187,47	4,64	2,33
11	104,31	*104,31*	90,19	2,1
12	159,84	165,91	102,67	1,87
Total	1736,06	1795,09	390,4024	21,45

Uma diferença de 3,27% separam os resultados obtidos do AGM das soluções ótimas, contudo, o AGM consegue alcançar em 3 das 12 instâncias o valor obtido pelo *CPLEX*, essa comparação se inverte quando observado o tempo de processamento computacional do *CPLEX* retardando 65,74% dos valores obtidos pelo AGM.

A Tabela 5.5 mostra os testes realizados com as 51 instâncias do PSPLIB de 10, 20 e 30 atividades demarcados por (J10), (J20) e (J30), porém apenas 2 instâncias de 20 atividades são contabilizadas, pois o *CPLEX* não consegue chegar a uma solução em um tempo limite de 5 horas de processamento, isso se agrava nas instâncias de 30 atividades, onde o *CPLEX* não apresenta nenhuma solução.

Tabela 5.5: Comparação entre AGM com o *CPLEX*

Instâncias	**CPLEX (C)**	**AGM (C)**	**Diferênça**	**CPLEX (T)**	**AGM (T)**	**Diferênça**
51 x J10	3547,22	3626,04	2,22%	5780,33	83,25	98,56%
2 x J20	108,54	111,88	3,08%	33502,41	4,22	99,99%
51 x J30	*	5339,86	*	*	222,93	*

É notável com o crescimento do número de atividades, que o tempo de processamento para a criação de uma solução do *CPLEX*, torna-se exponencial e inviabiliza o uso em aplicações reais, já o AGM se mantém com a mesma proporção de tempo de processamento por quantidade de atividades nas instância, assim, a adoção do AGM como um método auxiliar que construa soluções de boa qualidade é justificada pelos testes descritos nesse capítulo.

Capítulo 6

Considerações finais

Uma busca sistemática na literatura do problema mostrou que é desconhecido trabalhos que solucionem o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos com o auxilio de Matheurísticas, porém, são encontradas soluções criadas com heurísticas ou com métodos exatos.

Portanto, este trabalho alcançou seu objetivo de construiu Matheurísticas para a resolução do Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos baseadas em métodos evolutivos com Algoritmo Genético e a Otimização por Enxame de Partículas.

Também foi criada uma adaptação na modelagem original do problema, que integralizou os métodos com a programação matemática, e aplicou-a com a nova restrição de seleção das atividades críticas, que favoreceu uma melhora significativa na qualidade das xsoluções assim como uma redução considerável no tempo de processamento computacional.

A contribuição desta pesquisa traz ainda três novas heurísticas de criação de soluções para o problema, também servindo uma delas como um limitante inferior do *makespan*. Elas guiaram a diversificação dos pontos iniciais da busca no espaço de soluções, e reduziram as iterações dos algoritmos, consequentemente uma melhora no esforço computacional na convergência para bons resultados.

Os testes computacionais demonstraram que o Algoritmo Genético se sobrepôs ao PSO em todas as execuções das instâncias, e que a adição da programação matemática como mecanismo de busca local resultou em uma melhoria de 18% na qualidade das soluções obtidas.

A Matheurística baseada no Algoritmo Genético, apresentou-se como a melhor escolha

para solucionar o Problema de Custo de Disponibilidade de Recursos com Múltiplos Modos, uma vez que cria soluções com diferença de apenas 2% dos valores ótimos em um tempo 98% vezes menor do que os obtidos pela implementação matemática do problema. Ela provou ser eficaz e uma nova base para trabalhos futuros que necessitem da otimização de problemas dessa classe.

Acredita-se que um melhoria para o problema, seria a modelagem dos movimentos dos recursos entre as atividades e a criação de alguma heurística baseada em grafo de dependência descrito em [Procópio et al. 2013]. Ou ainda uma adaptação de Meta-heurísticas como o GRASP ou a Busca Tabu com as técnicas proposta neste trabalho.

Bibliografia

[Ballou 1993]BALLOU, R. H. *Logística empresarial: transportes, administração de materiais e distribuição física*. São Paulo: Atlas, 1993.

[Blazewicz, Lenstra e Kan 1983]BLAZEWICZ, J.; LENSTRA, J.; KAN, A. R. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, v. 5, p. 11–24, 1983.

[Boctor 1990]BOCTOR, F. F. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, v. 49, 1990.

[Boctor 1996]BOCTOR, F. F. A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research*, v. 90, 1996.

[Brucker et al. 1999]BRUCKER, P. et al. Resource-constrained project scheduling, notation, classification, models and methods. *European Journal of Operational Research*, v. 112, n. 1, p. 3–41, 1999.

[Chen, Li e Cai 2012]CHEN, L.; LI, X.; CAI, Z. Heuristic methods for minimizing resource availability costs in multi-mode project scheduling. *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, 2012.

[Daveis e Patterson 1975]DAVEIS, W. E.; PATTERSON, J. H. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science.*, v. 21, n. 8, p. 944–955, 1975.

[Demeulemeester 1995]DEMEULEMEESTER, E. Minimizing resource availability costs in time- limited project networks. *Management Science*, v. 41, 1995.

[Drexl e Gruenewald 1993]DREXL, A.; GRUENEWALD, J. Nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transaction*, v. 25, n. 5, p. 74–81, 1993.

[Drexl e Kimms 2001]DREXL, A.; KIMMS, A. Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*, v. 52, 2001.

[Hartmann 2001]HARTMANN, S. Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, v. 102, 2001.

[Hartmann e Kolisch 2000]HARTMANN, S.; KOLISCH, R. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, n. 2, p. 394–407., 2000.

[Herroelen, B. e Demeulemeester 1998]HERROELEN, W.; B., D. R.; DEMEULEMEESTER, E. Resource-constrained project scheduling: a survey of recent developments. *Computers Operations Research*, v. 25, n. 4, 1998.

[Hsu e Kim 2005]HSU, C. C.; KIM, D. S. A new heuristic for the multi-mode resource investment problem. *Journal of the Operational Research Society*, 2005.

[Kennedy e Eberhart 1995]KENNEDY, J.; EBERHART, R. Particle swarm optimization. *International Conference on Neural Networks*, 1995.

[Klein 2000]KLEIN, R. Bidirectional planning: improving priority rule-based heuristics for scheduling resource constrained projects. *European Journal of Operational Research*, v. 127, 2000.

[Kolisch e Hartmann 1998]KOLISCH, R.; HARTMANN, S. *Heuristic algorithms for the resource-constrained project scheduling problem, Classification and computational analysis.* [S.l.]: Springer, 1998. (International Series, Operations Research & Management Science, v. 14).

[Lima, Costa e Ochi 2003]LIMA, B. B.; COSTA, F. L. P.; OCHI, L. S. Melhorando o desempleno de metaheuristicas graso e algorimos evolutivos. *Simpósio Brasileiro de Pesquisa Operacional*, 2003.

[Lova et al. 2009]LOVA, A. et al. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 2009.

[Mendes, Gonçalves e Resende 2009]MENDES, J. J. M.; GONÇALVES, J. F.; RESENDE, M. G. C. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, v. 36, 2009.

[Moder, Phillips e Davis 1983]MODER, J.; PHILLIPS, C.; DAVIS, E. *Project management with CPM, PERT, and precedence diagramming*. Van Nostrand Reinhold, New York.: Trird Edition, 1983.

[Möhring 1984]MÖHRING, R. Minimizing costs of resource requirements, project networks subject to a fixed completion time. *Operations Research*, v. 32, p. 89–120, 1984.

[Mori e Tseng 1997]MORI, M.; TSENG, C. A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, v. 100, 1997.

[Nadjafi 2014]NADJAFI, B. A. Multi-mode resource availability cost problem with recruitment and release dates for resources. *Applied Mathematical Modelling*, v. 38, Nov 2014.

[NETO 2001]NETO, F. F. A logística como estratégia para a obtenção de vantagem competitiva. *Revista FAE Business*, n. 1, p. 1 – 4, 2001.

[Nobrega 2004]NOBREGA, C. *Ciência da Gestão: Marketing, Inovação e Estratégia*. Rio de Janeiro: Editora Senac Rio, 2004.

[NOVAES 2001]NOVAES, A. C. *Logística e gerenciamento da cadeia de distribuição: estratégia, operação e avaliação*. Rio de Janeiro: Editora Câmpus, 2001.

[Nudtasomboon e Randhawa 1997]NUDTASOMBOON, N.; RANDHAWA, S. U. Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs. *Computers and Industrial Engineering*, v. 32, n. 1, p. 227–242, Jan 1997.

[Parker e Rardin 1981]PARKER, R. G.; RARDIN, R. L. An overview of complexity theory in discrete optimization: Part i. concepts. *IIE Transactions*, v. 14, n. 1, 1981.

[Parker e Rardin 1982]PARKER, R. G.; RARDIN, R. L. An overview of complexity theory in discrete optimization: Part ii. results and implications. *IIE Transactions*, v. 13, n. 1, 1982.

[Procópio et al. 2013]PROCÓPIO, L. D. P. et al. Uma abordagem híbrida aplicada ao problema da alocação dinâmica de espaços. *11th Brazilian Congress on Computational Intelligence (BRICSCCI and CBIC)*, 2013.

[Qi et al. 2015]QI, J.-J. et al. Schedule generation scheme for solving multi-mode resource availability cost problem by modified particle swarm optimization. *Journal of Scheduling*, v. 18, p. 285–298, June 2015.

[Rangaswamy 1998]RANGASWAMY, B. *Multiple Resource Planning and Allocation in Resource- Constrained Project Networks*. Tese (Doutorado) — Graduate School of Business, University of Colorado, 1998.

[Shadrokh e F. 2007]SHADROKH, S.; F., K. A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. *European Journal of Operational Research.*, 2007.

[Slowinki 1981]SLOWINKI, R. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, v. 7, 1981.

[Talbot 1982]TALBOT, F. B. Resource-constrained project scheduling problem with time-resource tradeoffs: The nonpreemptive case. *Management Science*, v. 28, n. 10, p. 1197 – 1210, 1982.

[Yamashita, Armentano e Laguna 2006]YAMASHITA, D.; ARMENTANO, V. A.; LAGUNA, M. Scatter search for project scheduling with resource availability cost, special issue on scatter search. *European Journal of Operational Research*, v. 169, 2006.

[Yamashita e Morabito 2009]YAMASHITA, D.; MORABITO, R. A note on time/cost trade-off curve generation for project scheduling with multi-mode resource availability costs. *Journal of Operational Research*, v. 5, n. 4, 2009.

[Yamashita e Morabito 2012]YAMASHITA, D.; MORABITO, R. Um algoritmo branch-and-bound para o porblema de programação de projetos com custo de disponibilidade de recursos e múltiplos modos. *Gestão e Produção*, 2012.

Apêndice A

Apêndice A

Código Fonte A.1: Exemplo de dados das instâncias

```
1  ************************************************************************
2  file with basedata            : a101_.bas
3  initial value random generator: 1987
4  ************************************************************************
5  projects                      :  1
6  jobs (incl. supersource/sink ):  12
7  horizon bound                 :  80
8  RESOURCES
9    - renewable                 :  4   R
10   - nonrenewable              :  0   N
11   - doubly constrained        :  0   D
12   - partially renewable       :  0   P
13 ************************************************************************
14 PROJECT INFORMATION:
15 prono.  #jobs rel.date duedate tardcost  CPM-Time
16     1     10      0       11        3       11
17 ************************************************************************
18 PRECEDENCE RELATIONS:
19 jobno.    #modes  #successors   successors
20    1        1          3           2   3   4
21    2        3          2           5   6
22    3        3          2           7   9
23    4        3          1           8
24    5        3          2           9  10
25    6        3          2           9  10
26    7        3          1          11
27    8        3          2          10  11
28    9        3          1          12
29   10        3          1          12
30   11        3          1          12
```

```
  12      1       0
************************************************************************
REQUESTS/DURATIONS:
jobno. mode duration  R 1  R 2  R 3  R 4
  1      1      0       0    0    0    0
  2      1      1       3    0    0    0
         2      3       0    0    5    0
         3      8       0    0    0    7
  3      1      2       0   10    0    0
         2      3       0    0    4    0
         3      7       0    0    0    3
  4      1      3       0    0    7    0
         2      6       0    0    0    8
         3      9       0    0    5    0
  5      1      3       0    0    5    0
         2      6       9    0    0    0
         3      7       0    5    0    0
  6      1      1       0    0    0    7
         2      4       5    0    0    0
         3      8       0    0    7    0
  7      1      3       0    0    0    8
         2      6       4    0    0    0
         3      9       0    0    0    7
  8      1      4       0    0    0   10
         2      5       0    6    0    0
         3      6       0    0    0    5
  9      1      1       0    0    8    0
         2      2       0    7    0    0
         3     10       0    0    4    0
 10      1      4       0    7    0    0
         2      9       0    0    0    8
         3     10       0    0    6    0
 11      1      1       0    6    0    0
         2      5      10    0    0    0
         3      6       5    0    0    0
 12      1      0       0    0    0    0
************************************************************************
RESOURCE AVAILABILITIES:
  R 1  R 2  R 3  R 4
    4    4    3    5
************************************************************************
```

Apêndice B

Apêndice B

Código Fonte B.1: Método de calculo de curso por recurso em C++

```
1  void criarCustos() {
2    vector<int> uso_tipos(tipos);
3
4    for (int j = 0; j < this->j; j++) {
5      for (int i = 0; i < this->M[j]; i++) {
6        for (int k = 0; k < tipos; k++) {
7          uso_tipos[k] += (r[j][i][k] * d[j][i]);
8        }
9      }
10   }
11
12   float maior = 0;
13   float total = 0;
14   for (int k = 0; k < tipos; k++) {
15     if (maior < uso_tipos[k]) {
16       maior = uso_tipos[k];
17     }
18     total+=uso_tipos[k];
19   }
20
21   for (int k = 0; k < tipos; k++) {
22     custo_recurso[k] = ( (total - uso_tipos[k])  / maior );
23   }
24 }
```

O vetor criado na linha 2 é responsável por armazenar o calculo de utilização dos tipos de recursos recursos, já a linha 7 é responsável por somar toda a utilização do k tipos de recursos, em todos os i modos das j atividade. Nas linhas 16 e 18, são obtidos os valores da

maior utilização e da soma total respectivamente, e por fim, na linha 22 é atribuído o custo ao tipo de recurso de acordo com a sua utilização proporcional ao outros recursos.

Código Fonte B.2: Função $C_k(a_k)$

```
1  float calcular_custo() {
2          custo = 0;
3          for (int k = 0; k < tipos; ++k) {
4                  custo += custo_recurso[k] * demanda[k];
5          }
6          return custo;
7  }
```

Método responsável por quantificar uma solução. Na linha 4 a variável custo soma de toda a demanda de um recurso *demanda* multiplicado pelo custo atribuído ao recurso *custo_recurso*.

Código Fonte B.3: Função de calculo do *makespan*

```
1  int calcular_tempo() {
2          tempo = 0;
3          for (int j = 0; j < d->j - 1; j++) {
4                  if ((Ti[j] + D[j]) > tempo) {
5                          tempo = (D[j] + Ti[j]);
6                  }
7          }
8          return tempo;
9  }
```

Ao final de um escalonamento viável, é necessário utilizar o método acima para encontrar qual atividade servirá de *makespan* e será responsável pelo tempo da solução. A linha 4 é responsável por verificado qual a atividade com o maior tempo de termino dado por $Ti[j] + D[j]$.

Código Fonte B.4: Implementação das Heurísticas de criação das soluções

```
1  void iniciarSolucaoComMelhorMakespan() {
2          for (int j = 0; j < d->j; j++) {
3                  int modo = verificarMelhorModoPeloTempo(j);
4                  int tini = verificarTempoInicioCedo(j);
5                  alocarAtividade(j, tini, modo);
6          }
7          calcular_tempo();
8          calcular_custo();
9  }
10
```

```
11 void iniciarSolucaoComMelhorCusto() {
12         for (int j = 0; j < d->j; j++) {
13                 int modo = verificarMelhorModoPelaUtilizacao(j);
14                 int tini = verificarTempoInicioCedo(j);
15 
16                 alocarAtividade(j, tini, modo);
17         }
18         calcular_tempo();
19         calcular_custo();
20 }
21 
22 void iniciarSolucaoComMenorUtilizacao() {
23         for (int j = 0; j < d->j; j++) {
24                 int modo = verificarMelhorModoPelaMenorQuantidadeUtilizada(j);
25                 int tini = verificarTempoInicioCedo(j);
26 
27                 alocarAtividade(j, tini, modo);
28         }
29         calcular_tempo();
30         calcular_custo();
31 }
```

O código fonte B.4 exibe as implementações das três construções Heurísticas descritas na seção 4.1.1.