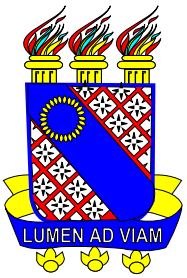


UNIVERSIDADE ESTADUAL DO CEARÁ

ÍTALO MENDONÇA ROCHA

UMA ABORDAGEM OTIMIZADA PARA O PROBLEMA DE ALOCAÇÃO DE EQUIPES E ESCALONAMENTO DE TAREFAS PARA A OBTENÇÃO DE CRONOGRAMAS EFICIENTES

FORTALEZA - CEARÁ

2011

ÍTALO MENDONÇA ROCHA

UMA ABORDAGEM OTIMIZADA PARA O PROBLEMA DE ALOCAÇÃO DE EQUIPES E ESCALONAMENTO DE TAREFAS PARA A OBTENÇÃO DE CRONOGRAMAS EFICIENTES

Dissertação apresentada no Curso de Mestrado em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadores: Gerardo Valdisio Rodrigues Viana e Jerffeson Teixeira de Souza

FORTALEZA - CEARÁ

2011

R672a Rocha, Ítalo Mendonça Rocha.

Uma Abordagem Otimizada para o Problema de Alocação de Equipes e Escalonamento de Tarefas para a Obtenção de Cronogramas Eficientes / Ítalo Mendonça Rocha. – Fortaleza, 2011.

120p.;il.

Orientadores: Prof. Dr. Gerardo Valdisio Rodrigues Viana e Prof. Dr. Jerffeson Teixeira de Souza.

Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

1. Otimização em Engenharia de Software 2. Alocação de Equipes 3. Escalonamento de Tarefas 4. Metaheurística 5.
. I. Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

CDD:001.6

ÍTALO MENDONÇA ROCHA

UMA ABORDAGEM OTIMIZADA PARA O PROBLEMA DE ALOCAÇÃO DE EQUIPES E ESCALONAMENTO DE TAREFAS PARA A OBTENÇÃO DE CRONOGRAMAS EFICIENTES

Dissertação apresentada no Curso de Mestrado em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre.

Aprovada em: 05/08/2011

BANCA EXAMINADORA

Prof. Dr. Gerardo Valdisio Rodrigues Viana
(Orientador)
Universidade Estadual do Ceará - UECE

Prof. Dr. Jerffeson Teixeira de Souza
(Orientador)
Universidade Estadual do Ceará - UECE

Prof. Dra. Mariela Inés Cortés
Universidade Estadual do Ceará - UECE

Prof. Dr. Carlos Eduardo Ferreira
Universidade de São Paulo - USP

AGRADECIMENTOS

Primeiramente gostaria de agradecer ao meu orientador, Prof. Dr. Valdisio, pela oportunidade de mais uma vez ser orientado por tamanho exemplo de professor. Agradeço a confiança depositada em mim e por toda sua sabedoria e atenção em conduzir essa orientação.

Ao meu orientador, Prof. Dr. Jerffeson, por toda orientação e excelência de conhecimento em SBSE a mim transmitidos.

A todos os professores do curso de Mestrado Acadêmico em Ciência da Computação da UECE.

Aos meus colegas, alunos do curso, pelos diversos conhecimentos compartilhados. Em especial ao meu eterno amigo Tales, que me incentivou e me apoiou em todos os momentos desse curso.

Ao SERPRO, pelo incentivo à pós-graduação. Agradeço pelas horas cedidas e flexibilizadas e pelos dados disponibilizados que me serviram de insumo para essa pesquisa.

Aos meus colegas do SERPRO Lacerda e Maiza, que me ajudaram em questões de matemática e gerência de projeto, respectivamente.

A minha mãe, Terezinha, e ao meu Pai, Gilbert (*in memoriam*), responsáveis pela minha verdadeira formação. Agradeço pelo enorme amor e carinho de sempre. Pai, essa dissertação é para você.

A minha esposa, Sayonara, por sua compreensão nas diversas vezes que estive ausente para realização deste mestrado. Por todo apoio que tenho recebido em todas minhas atividades acadêmicas. Pelo seu amor, fonte de energia para meus estudos.

Ao meu filho, Vitor, que me incentivou ainda mais a concluir esse trabalho antes de sua chegada.

“Embora ninguém possa voltar atrás e fazer um novo começo, qualquer um pode começar agora e fazer um novo fim.”
Chico Xavier.

RESUMO

Em desenvolvimento de software, assim como qualquer outra atividade de larga escala da engenharia, um efetivo planejamento de projeto é essencial. Falhas no planejamento ou planejamento precário podem causar atrasos e custos que, dadas restrições de tempo e orçamento, são frequentemente inaceitáveis, conduzindo a falhas críticas de negócio.

Uma das principais atividades do planejamento de projetos é o planejamento de cronograma, que envolve alocar equipes e escalonar tarefas. Alocar uma equipe consiste em atribuir as responsabilidades de uma tarefa a um conjunto de recursos humanos, enquanto o escalonamento de tarefa determina a data de início de execução de cada tarefa.

Essas são atividades complexas. Há uma grande quantidade de alocações possíveis, isto é, o gerente pode ter que formar equipes a partir de um grande conjunto de pessoas disponíveis e alocá-las em muitas tarefas distintas. De forma análoga, existem várias combinações possíveis de escalonar as tarefas. Além disso, existem várias alternativas a ponderar e várias restrições a respeitar, tais como prazos, orçamentos, limitação de recursos humanos, carga horária máxima e dependências entre tarefas. Os desenvolvedores podem ter capacidades técnicas diferentes e as tarefas podem exigir necessidades distintas.

Dessa forma, o gerente de projeto que deseja elaborar o cronograma com base apenas em suas experiências pode não alcançar bons resultados, tais como a redução do tempo e do custo do projeto. Além do mais, apesar de técnicas tradicionais, tais como *Project Evaluation and Review Technique (PERT)*, *Critical Path Method (CPM)*, diagramas *Gantt* e *Earned Value Analysis*, ajudarem a planejar e traçar marcos, elas normalmente não se propõem em desenvolver um cronograma ótimo na presença de alocação de recursos humanos e escalonamento de tarefas.

Um cronograma bem elaborado implica em benefícios ao projeto, que podem ser a diminuição da duração ou do custo do projeto e alocar equipes mais qualificadas em cada atividade do projeto.

Esta pesquisa propõe uma modelagem que utiliza técnicas de otimização para o Problema de Planejamento de Cronograma a fim de encontrar boas soluções num tempo computacional aceitável.

Palavras-Chave: Otimização em Engenharia de Software. Alocação de Equipes. Escalonamento de Tarefas. Metaheurística.

.

ABSTRACT

In software development, as well as any other activity of a wide range of engineering, an effective project planning is essential. Failures in planning or poor planning can cause delays and costs that, given time and budget constraints, are often unacceptable, leading to critical failures in business.

One of the main activities of project planning is the planning schedule, which involves allocating teams and scheduling tasks. Allocate a team is to assign the responsibilities of a task to a set of human resources while scheduling a task determines the date of execution of each task.

These are complex activities. There are a large number of possible allocations, ie, the manager may have to form teams from a large pool of people available and allocate them in many different tasks. Similarly, there are several possible combinations of scheduling tasks. In addition, there are several alternatives to consider and respect the various constraints such as deadlines, budgets, limited human resources, maximum workload and dependencies between tasks. The developers may have different technical capabilities and tasks may require different needs.

Thus, the project manager who wishes to develop the schedule based on his experiences just can not achieve good results, such as reducing the time and cost of the project. Moreover, although traditional techniques, such as *Project Evaluation and Review Technique (PERT)*, *Critical Path Method (CPM)*, diagrams *Gantt* and *Earned Value Analysis*, help plan and map landmarks, they usually do not propose to develop an optimal scheduling in the presence of human resource allocation and task scheduling.

A well-designed schedule implies benefits to the project, which may be the reduction of the project duration or a lower cost or allocate more qualified teams for each project activity.

We propose a model that uses optimization techniques to the problem of planning schedule in order to find good solutions in acceptable computational time.

Keywords: Search-Based Software Engineering. Allocation of Teams. Scheduling Tasks. traffic flows. Metaheuristic.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Para construir softwares de qualidade, as organizações geralmente seguem um processo de dividir o esforço do desenvolvimento em várias atividades ou pacotes de trabalho. Cada uma dessas atividades possuem características específicas, tais como habilidades, capacidades, e experiências. A maioria dessas características é encontrada em recursos humanos atribuídos para completar as atividades do projeto. Além da força de trabalho humano ser aportada como a responsável direto da duração e qualidade do projeto, recursos humanos geralmente representam a conta de maior custo em um projeto de software. Isto leva a uma questão fundamental: dado um grupo de desenvolvedores disponíveis e um conjunto de atividades, como alocar os recursos e como sequenciar as atividades de forma a trazer maiores benefícios à organização?

(TCHAO, 2007) diz, de acordo com o *PMBOK® Guide* (PMI, 2008), guia publicado pelo Project Management Institute, um projeto pode ser definido como uma sequência de atividades (ou eventos) com início e fim definidos, executadas segundo uma ordem previamente determinada, objetivando alcançar uma meta temporal pré-estabelecida. Estas atividades possuem duração e precedência pré-determinadas. Além disso, lançam mão de recursos - escassos e limitados - que realizam as atividades.

O **Gerenciamento do Prazo** do Projeto é um dos processos na realização de um projeto, e é constituído por um conjunto de subprocessos que interagem com outras etapas do projeto: **Definição das Atividades**, **Sequenciamento das Atividades**, **Estimativa da Duração das Atividades**, **Desenvolvimento do Cronograma**, e **Controle do Cronograma**. Ainda conforme o PMBOK® Guide, o principal produto gerado ao final do Sequenciamento das Atividades é o **Diagrama de Rede do Projeto** (Figura 6).

A Figura 1 descreve a troca de artefatos entre processos segundo a visão do *Project Management Institute - PMI*. Conforme ilustra a figura, são entradas básicas para o desenvolvimento de cronograma:

- **Atividades**: uma lista de atividades, incluindo suas especificações, suas interdependên-

cias e o esforço estimado para sua realização;

- **Recursos**: recursos disponíveis para o projeto e seus atributos; e

- **Informações Gerenciais**: informações tais como escopo do projeto, planejamento de risco etc.

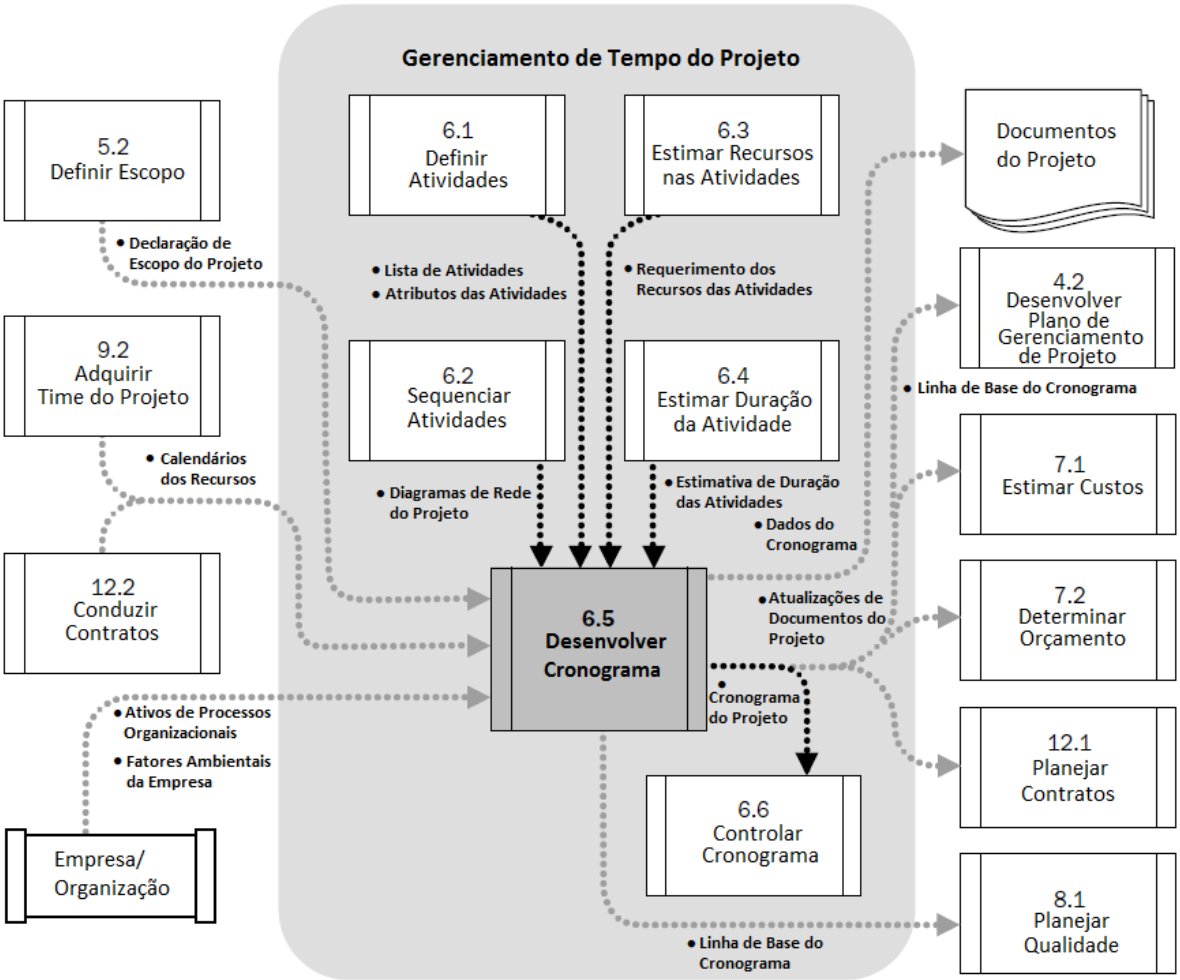


Figura 1: Fluxo de Dados entre Processos (PMI, 2008)

O Problema de Desenvolvimento do Cronograma é um importante problema que não pode ser negligenciado. É uma prática fundamental da Engenharia de Software que define as responsabilidades dos recursos humanos e os instantes de início de cada atividade em um projeto de software com restrições específicas, a fim de alcançar objetivos diversos e muitas vezes conflitantes, tais como reduzir a duração e o custo do projeto.

O planejamento de cronograma considerado nessa pesquisa compreende a junção de dois conheci- dos problemas: a Alocações de Equipes e o Escalonamento de Tarefas. Alocar uma equipe consiste em atribuir as responsabilidades de uma tarefa a um conjunto de recursos humanos. Já o Escalonamento de Tarefas determina a ordem de execução de cada tarefa. Ambos os problemas procuram buscar cronogramas ótimos ou próximos do ótimo. A partir do momento em que sabemos as atribuições de cada recurso e a ordem de execução das tarefas, podemos determinar o instante de início de execução de cada tarefa e assim teremos o cronograma estabelecido por completo.

Apesar destes dois problemas estarem intrinsecamente relacionados, geralmente eles são trabalhados individualmente. Pesquisas que abordam apenas a alocação de equipes consideram previamente estabelecida a ordem de execução das tarefas. Portanto procura-se apenas definir a melhor forma de alocar os recursos naquela ordem. De forma análoga, problemas de escalonamento de tarefas geralmente consideram as alocações pré-estabelecidas e destinam-se apenas em definir a melhor ordem de execução das tarefas. Este último problema é conhecido como **Problema de Escalonamento de Tarefas de Projetos** (*PSP - Project Scheduling Problem*) (ICHIHARA, 2002).

De acordo com o *PMBOK® Guide*, existem técnicas para o desenvolvimento do cronograma baseadas em análises matemáticas (*CPM - Método do Caminho Crítico*, *PERT - Program Evaluation and Review Technique*), em compressão da duração (*Crashing, Fast Tracking*), em simulações (*Análise Monte Carlo*), e em heurísticas (*Critical Chain*, *SGS*, metaheurísticas). Adicionalmente, existem softwares que auxiliam os Gerentes de Projetos nesta tarefa, como o *MSProject*, que utiliza a heurística *single pass methods - SGS* -, o *Primavera*, o *CS Project*, o *Fast Track Schedule*, entre outros (TCHAO, 2007). No entanto, elas normalmente não se propõem em desenvolver um cronograma ótimo na presença de alocação de recursos humanos e escalonamento de tarefas.

O Problema de Alocação de Equipes e de Escalonamento de Tarefas são difíceis de serem resolvidos em sua otimalidade. A junção dos dois problemas no intuito de desenvolver o cronograma é ainda um problema mais complexo e, portanto, exigiria um tempo muitas vezes

inaceitável para o planejamento de cronogramas em projetos de grande porte (grande quantidade de atividades e recursos humanos). Isso justifica o fato desse trabalho adotar heurísticas para o desenvolvimento do cronograma ótimos ou próximos do ótimo.

Portanto, essa pesquisa tem como objetivo propor uma modelagem para o Problema de Elaboração de Cronograma, compreendendo tanto as alocações de equipes nas atividades do projeto quanto em definir a melhor ordem de execução dessas atividades, a fim de gerar bons cronogramas, minimizando a duração e o custo do projeto e alocando equipes mais qualificadas em cada atividade do projeto.

1.1 Objetivos

Essa pesquisa tem como principal objetivo propor uma abordagem otimizada híbrida, que envolve o Problema de Alocação de Equipes e o Problema de Escalonamento de Tarefas com Restrição de Recursos, a fim de elaborar bons cronogramas que minimize o tempo, o custo e maximize a qualidade na formação das equipes nas atividades do projeto.

Discutir as principais características relacionadas ao Problema de Alocação de Equipes e ao Problema de Escalonamento de Tarefas com Restrição de Recursos.

Incluir na modelagem as características mais condizentes com situações reais, de forma que permita a modelagem ser aplicado na prática.

Implementar um ou mais algoritmos baseados na modelagem e desenvolver um aplicativo (ferramenta) com interface gráfica que auxilie no cadastro das habilidades, recursos e tarefas, e que permita ajustar os parâmetros de entrada dos algoritmos. Além disso, disponibilizar a visualização gráfica e bem detalhada dos cronogramas gerados pelos algoritmos.

Efetuar testes em um projeto real e comparar os resultados dos algoritmos com o cronograma elaborado pelo gerente de projeto. Além disso, disponibilizar na ferramenta a opção de gerar projetos sintéticos para que testes possam ser realizados também em dados artificiais.

Conduzir experimentos e investigações da modelagem proposta através de simulações auxiliadas pelo aplicativo implementado. Isso permite o gerente de projeto tomar decisões mais

precisas baseadas em um modelo.

1.2 Estrutura do Documento

Os capítulos seguintes estão dispostos da seguinte maneira: o Capítulo 2 contextualiza o problema foco desse trabalho, o Problema de Elaboração de Cronograma. Nesse mesmo capítulo, o Problema de Alocação de Equipes e o Problema de Escalonamento de Tarefas são tratados separadamente.

Noções de Otimização e Engenharia de Software são exibidas no Capítulo 3. Nesse capítulo, apresentamos uma visão geral de SBSE, área emergente da Ciência da Computação que utiliza técnicas de otimização na resolução de problemas de engenharia de software. Também apresentamos as metaheurísticas GA e NSGA-II, algoritmos base de implementação da abordagem proposta.

No Capítulo 4, apresentamos a modelagem proposta, reunindo todos os conceitos aprendidos nos capítulos anteriores.

Testes computacionais são realizados no Capítulo 5. Além de projetos fictícios, é realizado um estudo empírico em um projeto real de uma empresa de grande porte e uma análise de sensibilidade do modelo proposto.

Finalmente o Capítulo 6 expõe as conclusões obtidas e discute possíveis trabalhos futuros.

2 O PROBLEMA DE ELABORAÇÃO DE CRONOGRAMA

O planejamento de cronograma considerado nessa pesquisa compreende a junção de dois conhecidos problemas: o Problema de Alocação de Equipes (PAE) e o Problema de Escalonamento de Tarefas com Restrição de Recursos (*Resource-Constrained Project Scheduling Problem - RCPSP*). Consultar (COLARES, 2010) e (ICHIHARA, 2002) para obter maiores detalhes do PAE e RCPSP, respectivamente. Alocar uma equipe consiste em atribuir as responsabilidades de uma tarefa a um conjunto de recursos humanos. O Escalonamento de Tarefas determina a ordem de execução de cada tarefa. As Seções 2.2 e 2.3 têm como meta contextualizar os dois problemas isoladamente.

A Figura 2 ilustra uma visão geral sobre o problema de alocação e de escalonamento e o problema proposto de Elaboração de Cronograma. Nesse gráfico, percebe-se a existência de duas principais entidades: Os Recursos Humanos e as Tarefas. Os recursos humanos possuem habilidades, carga horárias e salários individuais e seus períodos de disponibilidades ao longo do projeto. As tarefas possuem interdependências, esforço estimado em homens-hora e habilidades requeridas, representadas pelo Diagrama de Rede do Projeto (consultar Seção 2.3 e Figura 6). Essas entidades representam os dados de entrada comum aos problemas. Além dos dados em comum, o Problema de Escalonamento recebe as alocações dos recursos humanos. Logo, esse problema resume-se apenas em definir qual a melhor ordem de execução das atividades. De forma análoga, o Problema de Alocação de Equipes, além dos dados em comum, conhece previamente a ordem de execução das atividades, cabendo a ele definir qual a melhor forma de alocar os recursos humanos. O problema proposto de Elaboração de Cronograma recebe como entrada apenas os dados em comum. Portanto seus objetivos consistem em determinar tanto a melhor forma de alocação dos recursos quanto a melhor ordem de execução das atividades. Todos os três problemas procuram buscar bons cronogramas que, por exemplo, diminuam a duração e/ou o custo do projeto.

A partir do momento em que sabemos as atribuições de cada recurso e a ordem de execução das tarefas, podemos determinar o instante de início de execução de cada tarefa e assim

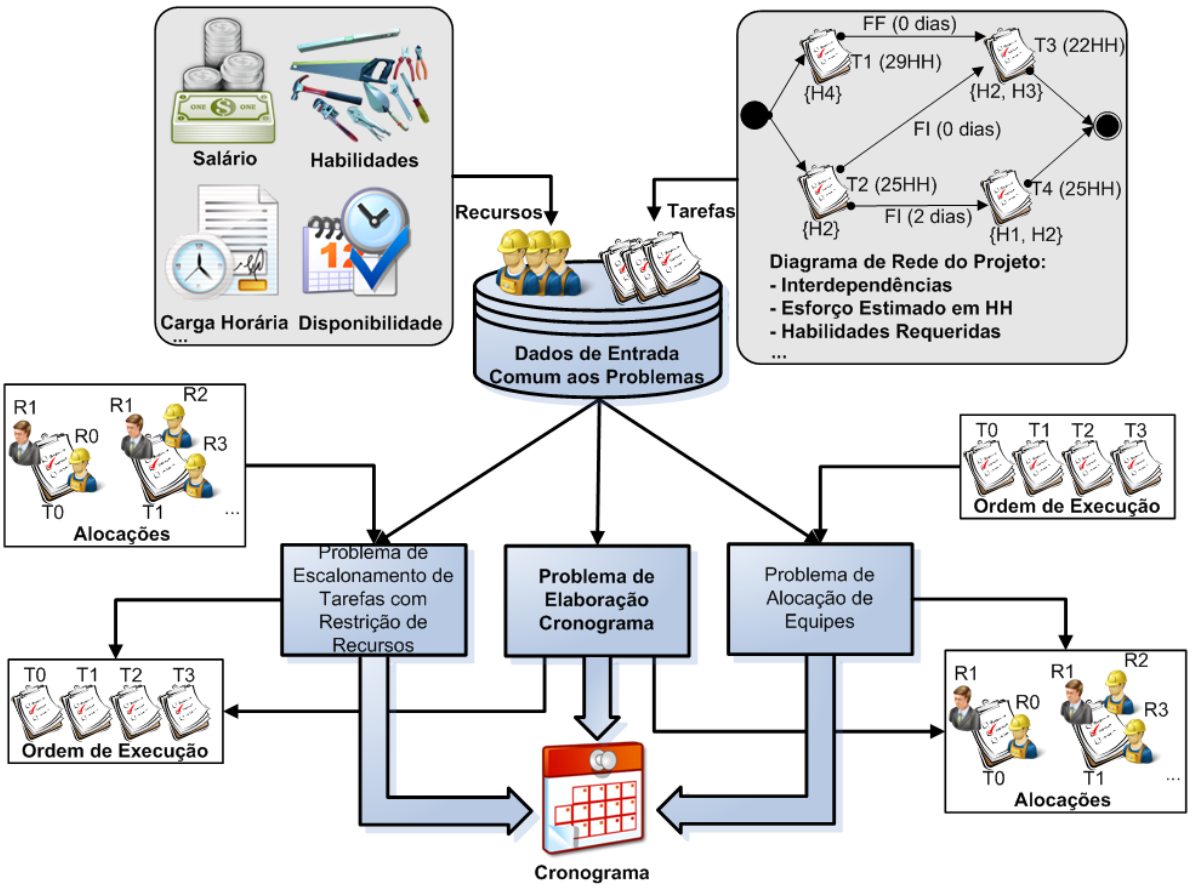


Figura 2: Visão Geral do Problema de Elaboração de Cronograma

teremos o cronograma estabelecido por completo. Um exemplo de um cronograma adequado à modelagem proposta pode ser visto na Figura 3. Nessa figura, podemos perceber a existência de vários elementos que serão detalhados ao longo desse trabalho. O eixo da abscissa representa a duração do projeto em **dias** e o eixo da ordenada as horas de dedicação dos recursos humanos $R1$ e $R2$ na execução das tarefas. Cada tarefa é representada por uma cor distinta: $\{T_{verde}$, $T_{vermelho}$, T_{azul}, $T_{amarelo}$, ...$\}$. Logo, as alocações são representadas por blocos preenchidos na cor da tarefa em que o recurso está sendo alocado. A linha grossa ao longo do cronograma significa a disponibilidade total (horas normais + extra) do recurso durante o projeto. A carga horária diária do recurso é informada pelo círculo em volta da hora no eixo da ordenada. Para maiores informações consultar a Seção 2.4.

Uma observação importante é que a unidade *hora* é trabalhada nessa pesquisa não é necessariamente um valor inteiro discreto. Podemos então ter uma alocação de 5,3 horas ou

uma tarefa com esforço estimado de 32,6 horas. Na realidade, trabalhamos essa unidade em nível de minutos, permitindo a construção de cronogramas mais flexíveis. Logo, daqui para frente, toda referência à unidade hora será tratada como um valor numérico real ou em função de minutos.

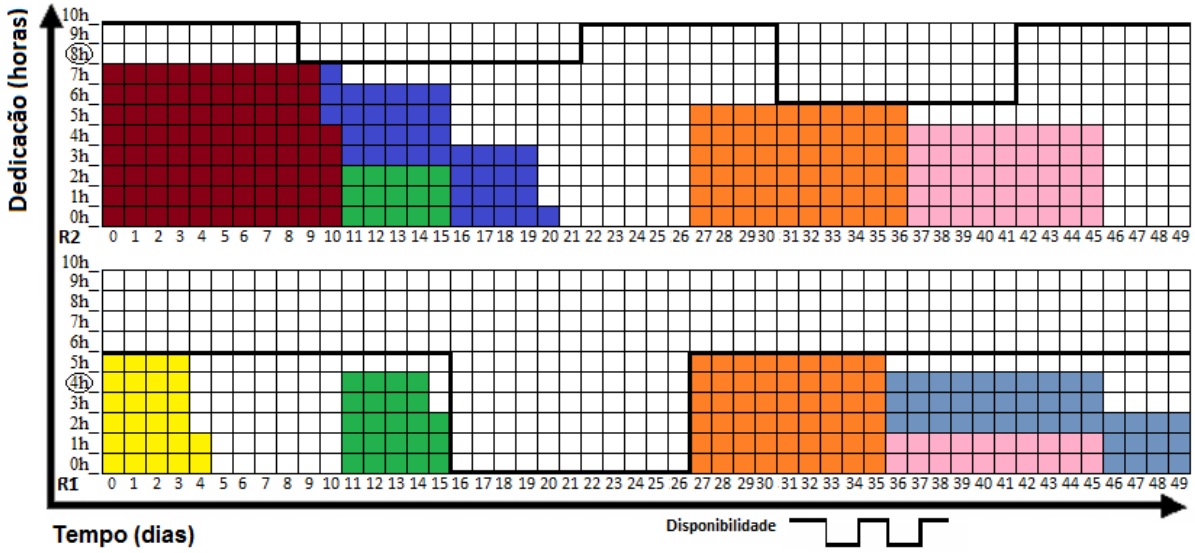


Figura 3: Exemplo de um Cronograma

2.1 Trabalhos Relacionados

Foi mostrado em (COFFMAN JR.; GAREY; JOHNSON, 1997) que o problema de formação de equipes pode ser pensado como um problema *Bin Packing*, conhecido problema pertencente à classe de problemas NP-Difíceis. Este fato demonstra a complexidade do problema e, consequentemente, torna-o ideal à aplicação de técnicas de otimização.

(BURDETT; LI, 1995) foram os primeiros a especificar como quantificar as capacidades técnicas dos recursos humanos, associá-los em atividades e combinar em conjunto para produzir equipes eficazes em um problema de alocação de recursos. Até então a alocação de recursos humanos em atividades de projeto era feito manualmente, através de experiências do gerente e geralmente utilizando técnicas como PERT e CMP.

(QURESHI; HARMAN, 2006) apresentaram um trabalho que trata do problema de alocação de equipes em pacotes de projeto utilizando metaheurísticas e estudando o efeito da Lei de Brooks, a fim de entender o relacionamento entre recursos humanos e tempo na presença

de tarefas inter-relacionadas. Os membros de cada equipe já são pré-definidos, isto é, não existe formação dinâmica de equipes nos pacotes de projeto. O problema considera a redução do tempo do projeto e os testes são realizados em cima de uma base de dados artificiais.

(ALBA; CHICANO, 2007) apresentaram um estudo sobre o Problema de Escalonamento de Projeto (*Project Scheduling Problem - PSP*), que consiste em definir quem faz o que ao longo do projeto. Foi utilizado um algoritmo genético para minimizar dois objetivos: o tempo e o custo do projeto. A formulação proposta considera aspectos importantes, como habilidades individuais dos recursos, interdependência entre tarefas e percentual de dedicação dos recursos nas tarefas. A validação da abordagem foi realizada apenas utilizando dados artificiais obtidos a partir de um gerador automático de projetos, não sendo testada em projetos reais.

(PENTA et al., 2007) apresentam uma descrição formal dos modelos da Lei de Brooks e analisam o impacto desses modelos sobre a otimização dos tempos de conclusão em manutenção de projetos de software reais.

(BARRETO; BARROS; WERNER, 2008) formularam o problema como um problema de satisfação de restrição. Os autores consideram diversos possíveis objetivos na alocação, como equipes mais qualificadas, menos qualificadas, de menor custo ou mais velozes. O problema proposto é resolvido por um algoritmo exato denominado *backtracking*. Nesse trabalho, os autores consideram que o projeto é dividido em pequenas tarefas de curta duração e elas são executadas por uma única pessoa.

(WEN; LIN, 2008) propuseram uma nova codificação de melhoria do GA multi-objetivo (moGA) para resolver de forma multi-objetiva o problema de alocação de recursos humanos de múltiplos estágios (MHRAP).

Mais recentemente, (COLARES, 2010) trabalhou a alocação de equipes e desenvolvimento de cronogramas utilizando otimização multi-objetiva e abordando vários conceitos interessantes, tais como habilidades e experiências individuais, interdependência entre tarefas, estimação de esforço e produtividade das equipes, hora extra, *overhead* de comunicação, curvas de aprendizado e treinamentos. Entretanto, o trabalho considera a ordem de execução das atividades previamente estabelecida e, portanto, concentra esforços apenas na busca de aloca-

ções ótimas.

(ANTONIOL; PENTA; HARMAN, 2005) foram os primeiros a avaliar o problema de ordenação de pacotes de trabalho e formações de equipes em cada pacote em projetos de manutenção de software. Reportaram a evolução de três diferentes técnicas de otimização, combinados com teoria das filas em um projeto de software real. De acordo com o modelo definido pelos autores, a duração do projeto pode variar de acordo com o número de pessoas disponíveis e tamanhos das equipes. O artigo emprega representações interessantes referente à ordenação de pacotes de trabalho e alocação de equipes nesses pacotes, onde a ordem da execução dos pacotes influencia na qualidade da solução. Porém essa ordem de execução dos pacotes não chega a ser um Esquema de Geração de Escalonamento (SGS) geralmente utilizado em problemas RCPSP.

De forma análoga ao trabalho anterior, (ANTONIOL; PENTA; HARMAN, 2004) aplicaram um algoritmo genético que, no primeiro estágio, estabelece a ordenação dos pacotes de trabalhos e, em segundo estágio, designa as equipes nos pacotes. Foi aplicado um GA combinado com teoria das filas ao problema de alocação de equipes em projeto real de manutenção de software. Foram considerados fatores interessantes como erros de estimativa, abandono e retrabalho. A partir de estudos empíricos, os autores concluíram que a utilização de Otimização em Engenharia de Software (*Search Based Software Engineering - SBSE*) (consultar Capítulo 3) pode reduzir a duração do projeto em até 50%.

(CHANG et al., 2008) apresentaram ideias e uma modelagem interessante utilizando o conceito de linha de tempo para o problema de alocação de Recursos Humanos e Escalonamento de Tarefas. Eles modelaram o problema em uma matriz tridimensional, onde a primeira dimensão é a unidade do tempo escolhida, a segunda é o empregado, a terceira é a tarefa e a célula é a dedicação do empregado na tarefa. A abordagem utilizada evitou o uso de técnicas de geração de escalonamento, que definem os instantes de início e fim de cada atividade, geralmente utilizadas para resolver o RCPSP. Além disso, os recursos podem ser alocados ou desalocados em tarefas em andamento e as tarefas podem ser pausadas e reiniciadas (preempção).

Trabalhos em Escalonamento de Tarefas que permitem uma tarefa ser interrompida e

depois reiniciada (preempção) podem ser encontrados em (SCHRAGE, 1972), (WEGLARZ et al., 1977), (SLOWINSKI, 1980) e (SLOWINSKI, 1981).

Os procedimentos utilizados no escalonamentos de tarefas propostos nesse trabalho geram apenas uma único cronograma a partir de uma matriz de alocação e de uma lista de atividades. Pesquisas que aplicam o método X-Pass Multi-Pass no procedimento de escalonamento para a geração de múltiplos cronogramas podem ser obtidos em (BOCTOR, 1990), (BOCTOR, 1996a), (DREXL; GRUENEWALD, 1993) e (VACA, 1995).

(ICHIHARA, 2001) apresenta uma síntese das heurísticas baseadas em Regras de Prioridade como solução ao RCPSP. Esses métodos também têm sido usados para encontrar soluções iniciais para metaheurísticas destinadas a resolver esse problema. No ano seguinte, o mesmo autor em (ICHIHARA, 2002) resume em seu artigo as principais características do RCPSP.

Exemplos de trabalho que utilizam Regras de Prioridade como forma de representar as soluções ao RCPSP são encontrados em (DAVIS; PATTERSON, 1975), (COOPER, 1976), (COOPER, 1977), (LAWRENCE, 1985), (ALVAREZ-VALDES; TAMARIT, 1989), (BOCTOR, 1990), (KOLISCH, 1996) e (KOLISCH; HARTMANN, 1999).

Geralmente o que se deseja obter no RCPSP é o *makespan*, isto é, o menor tempo de execução de todas as tarefas. No entanto, é crescente o número de pesquisas que utilizam objetivos múltiplos. Podemos mencionar (DAVIS; PATTERSON, 1975), (LI, 1996), (EPSTEIN; WILAMOWSKY; DICKMAN, 1992) e (KURTULUS; DAVIS, 1982).

Métodos exatos baseados em enumeração implícita com "Branch-and-Bound" têm sido aplicados com sucesso no RCPSP. Dentre estes se destacam: (DEMEULEMEESTER; HERROELEN, 1992), (SIMPSON; PATTERSON, 1996), (BRUCKER et al., 1998), (MINGOZZI et al., 1998) e (SPRECHER, 2000). Devido à complexidade do problema, estes métodos são aplicados a problemas testes envolvendo, no máximo, 60 atividades. Outro método exato usado para resolver esse problema é a Programação Dinâmica. Os primeiros trabalhos foram propostos por (CARRUTHERS; BATTERSBY, 1966) 1966.

Metaheurísticas têm sido aplicados para resolver o RCPSP. (BOULEIMEN; LECOCQ, 2003), (LEE; KIM, 1996), (BOCTOR, 1996b), (CHO; KIM, 1997) desenvolveram algoritmos com Têmpera Simulada. Algoritmos Genéticos foram propostos por (LEE; KIM, 1996), (ÖZDAMAR, 1999), (HARTMANN, 1997), (HARTMANN, 2002) e (ALCARAZ; MAROTO, 2001). Algoritmos de Busca Tabu foram desenvolvidos por (SIMõES, 2004), (TCHAO, 2007), (LEE; KIM, 1996), (THOMAS; SALHI, 1998) etc. (VALLS; QUINTANILLA; BALLESTIN, 2003) desenvolveram um novo algoritmo que usa busca local com intensificação e diversificação. Também identificamos os trabalhos de (MERKLE; MIDDENDORF; SCHMECK, 2000) usando Colônia de Formigas.

A Tabela 1 compara algumas características dos trabalhos relacionados que mais influenciaram esta pesquisa e sua legenda é composta pela Tabela 2. Essas características serão explicadas ao longo desse capítulo. O resumo delas pode ser visto na Tabela 3.

2.2 O Problema de Alocação de Equipes

Sabemos que o esforço de desenvolvimento do projeto é dividido em tarefas. Uma vez que as estimativas de esforço dessas tarefas são traçadas e suas ordens de execução definidas, uma das mais importantes atividades do gerente de projeto é alocar recursos nessas tarefas. A forma como esses recursos são alocados pode influenciar na duração e/ou no custo do projeto e/ou na qualidade do produto.

Entretanto, essa não é uma atividade simples, há muitas alternativas a ponderar e várias combinações a avaliar, e o gerente pode ter que escolher um time dentre um vasto conjunto de recursos disponíveis. Devido a sua complexidade e ao fato de que normalmente as técnicas tradicionais de apoio a gestão não se propõem a desenvolver uma alocação ótima, esta atividade fica a cargo apenas da experiência e sentimentos do gerente, uma vez que seria considerado infinito o tempo dele listar todas as combinações possíveis de alocações de recursos humanos.

Foi mostrado em (COFFMAN JR.; GAREY; JOHNSON, 1997) que o problema de alocação de recursos pode ser pensado como o **Problema do Empacotamento** (*Bin Packing*), pertencente a uma classe mais geral denominada de **Problemas da Mochila** (*Knapsack Pro-*

blems) e pertencente à classe de problemas NP-Difícil (FALKENAUER; DELCHAMBRE; DELCHAMBRE, 1992). No Problema do Empacotamento, objetos de diferentes volumes devem ser colocados dentro de um número finito de caixas a fim de minimizar o número de caixas usadas, respeitando o limite de capacidade das caixas.

O problema de alocação de recursos também pode ser modelado, porém de forma mais limitada, como um **Problema de Designação** que consiste em designar de maneira ótima cada uma das origens a cada um dos destinos. Suponha r Recursos humanos a distribuir por t Tarefas de maneira que cada trabalhador execute apenas uma tarefa, e que cada tarefa seja executada apenas por um trabalhador. Suponha também o conhecimento do gerente dos custos da realização de cada tarefa por cada trabalhador. Deseja-se então designar os trabalhadores às tarefas de forma a minimizar os custos (financeiro ou temporal). Alguns exemplos práticos de sua aplicabilidade são:

- Várias fábricas e vários destinos sendo que uma fábrica abastecerá apenas um destino;
- Vários funcionários e várias tarefas, designar um funcionário para cada tarefa;
- Várias obras e várias empresas, uma obra para cada empresa.

Na modelagem do Problema de Alocação de Equipes como um Problema de Designação, é considerada apenas a alocação de um único recurso por tarefa. Nesse caso, um recurso despende toda sua carga horária na execução dessa tarefa até que ela seja concluída e, portanto, não poderá trabalhar em mais de uma tarefa no mesmo dia antes da conclusão da tarefa corrente. Essa é uma abordagem mais simples e fácil de ser implementada, porém limitada, pois na prática sabemos que pode haver mais de um recurso trabalhando em uma determinada tarefa e que um recurso pode estar trabalhando paralelamente em mais de uma tarefa no mesmo dia. Ela pode ser representada por um vetor de tamanho t, onde t é a quantidade total de tarefas e a célula i informa o recurso alocado na i-ésima tarefa (ver Figura 4).

No entanto, sabemos que muitas vezes o gerente precisa alocar mais de um recurso em uma tarefa e que também um recurso poder desempenhar diferentes atividades paralelamente.

Este tipo de alocação de múltiplos recursos, mais completa e condizente com a realidade, é o adotado por essa pesquisa. (ALBA; CHICANO, 2007) apresentaram essa abordagem representando as alocações em uma matriz $t \times r$, onde r é a quantidade de recursos humanos envolvidos no projeto e o conteúdo de cada célula dessa matriz representa o percentual de dedicação do recurso na tarefa relativo à sua carga horária. Para uma carga horária de 8 horas, a célula com valor de 0.75 significa que o recurso dedicará 6 horas por dia nessa tarefa (consultar Figura 5). A única diferença é que discretizamos a dedicação em termos de horas trabalhadas diariamente, variando em um número inteiro entre zero e dez. Dessa forma, *0* significa que o recurso não está alocado na tarefa, *4* que o recurso dedica 4 horas diárias na resolução da atividade e *10* que o recurso trabalha suas 8 horas normais mais 2 horas extras por dia nessa tarefa. Ver Seção 2.4 para maiores informações sobre carga horária e hora extra.

R_3	R_2	R_4	R_1
T_1	T_2	T_3	T_1

Figura 4: Representação da Alocação de um Único Recurso por Tarefa

	$\boldsymbol{T_1}$	$\boldsymbol{T_2}$	$\boldsymbol{T_3}$	$\boldsymbol{T_4}$
$\boldsymbol{R_1}$	5	8	2	2
$\boldsymbol{R_2}$	3	1	3	6
$\boldsymbol{R_3}$	6	5	3	0

Figura 5: Representação da Alocação de Múltiplos Recursos por Tarefa

Seja qual for a modelagem utilizada, a complexidade do problema torna-o ideal à aplicação de técnicas de otimização. Para maiores detalhes sobre técnicas de otimização ver o Capítulo 3.

Otimização em Engenharia de Software (*Search Based Software Engineering - SBSE*) é uma área emergente da Ciência da Computação que utiliza técnicas de otimização, geralmente metaheurísticas, para resolver problemas da Engenharia de Software em atividades do ciclo de vida da engenharia de requisitos, planejamento de projeto e estimação de custo. Para obter maiores detalhes sobre SBSE consultar Capítulo 3.

Para o Problema de Alocação de Equipes, geralmente considera-se que a ordem de execução das tarefas faz parte dos dados de entrada do problema. Dessa forma, uma vez determinada previamente a sequência de execução das tarefas, o problema de alocação de recursos

resume-se em definir quais recursos são usados para executar quais tarefas a fim de alcançar boas soluções.

Recurso é uma palavra muito genérica que pode representar vários elementos necessários para a execução das atividades do projeto, tais como funcionário, máquina, tempo e dinheiro. No entanto, consideramos apenas os recursos humanos, pois eles são fatores-chave de sucesso em um projeto e normalmente considerados a conta de maior custo no orçamento.

Uma vez que a tarefa é formada por uma equipe, é importante levar em consideração o *overhead* de comunicação, isto é, o tempo gasto pelos membros da equipe para se comunicarem entre si na tentativa de resolver uma tarefa. Esse assunto é detalhado na Subseção 2.4.4.

Sabemos que muitas vezes pode ser útil remover recursos humanos alocados em uma tarefa em andamento ou alocar outros recursos em uma tarefa em execução. No entanto, assumimos nessa modelagem que a equipe alocada em uma tarefa trabalha nela desde o início até sua conclusão, em outras palavras, não é permitido alterar os membros de uma equipe alocada em uma atividade em execução. Manter a equipe inalterada também pode constituir vantagens para o projeto, tais como aumentar a experiência da equipe na atividade em execução e aumentar a afinidade entre os integrantes da equipe.

Ao passo que reter os recursos em uma tarefa até sua conclusão constitui vantagens para o projeto, e manter a afinidade entre seus integrantes, bem como diminuir as curvas de aprendizado e o *overhead* de transição entre tarefas, além de manter a estabilidade do projeto, algumas vezes pode ser útil remover uma pessoa de uma tarefa em andamento e alocá-lo em outra. Porém, este modelo considera que, uma vez que um recurso é alocado em uma atividade, ele deve permanecer nela até sua conclusão.

2.3 O Problema de Escalonamento de Tarefas com Restrição de Recursos

A ordem de processos executados na CPU, o sequenciamento de atividades em uma linha de montagem, a ordem de execução das atividades no desenvolvimento de um projeto de software são exemplos de problemas que podem ser modelados como um problema de escalo-

namento.

No caso do desenvolvimento de um projeto de software, Escalonar Tarefas consiste em determinar a melhor ordem de execução dentre um conjunto de tarefas com durações pré-estabelecidas, respeitando suas precedências, a fim de produzir maior benefício à organização. Geralmente deseja-se minimizar o tempo total de execução de todas as tarefas, conhecido como ***makespan***.

Muitos tipos de escalonamento exigem o consumo obrigatório de recursos, tanto materiais (papel, madeira, trator, dinheiro) como pessoais (mão de obra). Para maiores detalhes dos tipos de recursos, consultar Subseção 2.4.2. Além da restrição de precedência entre as atividades (ver Subseção 2.4.1), existe também a limitação de recursos consumidos nas atividades. Para esse tipo de problema chamamos de **Problema de Escalonamento de Tarefas com Restrição de Recursos - PPPRR** (*Resource-Constrained Project Scheduling Problem - RCPSP*). Esse problema também é chamado de Problema da Programação de Projetos com Limitação de Recursos ou Problema do Sequenciamento de Projetos com Recursos Limitados.

RCPSP é o Problema de Escalonamento de Projetos mais básico. Ele é uma generalização do problema de ***Job-Shop*** estático e, portanto, pertence à classe dos problemas de otimização NP-Completo (BLAZEWICZ; LENSTRA; KAN, 1983). Diversos problemas reais não podem ser modelados através do RCPSP, e diversas extensões têm sido consideradas na literatura. O Problema de Elaboração de Cronograma proposto nesse trabalho envolve a junção do Problema de Alocação de Equipe e uma das extensões do RCPSP, denominada **Problema Generalizado de Programação de Projetos com Restrição de Recursos (PGPPRR)**. A extensão do problema generalizado abordado nesse trabalho é a disponibilidade de recursos humanos ao longo do projeto, isto é, cada recurso pode estar indisponível por algumas horas por dia ou integralmente por um determinado período do projeto. Outro exemplo de extensão, porém não implementado nesse trabalho, é o tempo de preparação de cada atividade. Dentre os tipos de recursos existentes, consideramos apenas os recursos humanos, pelo fato de serem os responsáveis diretos pelo sucesso do projeto e por ser o fator de maior custo dentro do orçamento.

O RCPSP geralmente parte do princípio que já foi definida as alocações dos recursos nas tarefas. Dessa forma, o RCPSP resume-se em definir a melhor ordem de execução das atividades a fim de construir bons cronogramas. Uma vez que os recursos já foram designados em suas atividades e a ordem de execução das atividades já foram definidas, é possível descobrir o instante de início de cada atividade para a montagem do cronograma. O conceito de bom cronograma está relacionado com os objetivos da organização, que podem ser a diminuição da duração do projeto (*makespan*), a redução do custo do projeto, entre outros.

Diagrama de Rede do Projeto é um esquema de apresentação das atividades do projeto e dos inter-relacionamentos (dependências) entre elas. O RCPSP recebe como entrada o Diagrama de Rede do Projeto além de outros atributos, tais como a estimativa de esforço em homens-hora de cada tarefa e as habilidades exigidas, os recursos humanos e suas propriedades (salário, carga horária, habilidades e disponibilidade). A Subseção 2.4.4 explica com maiores detalhes como o projetista alcança a estimativa de esforço das atividades.

A Figura 6 exibe um exemplo desse diagrama. Cada tarefa possui seu esforço estimado em **Homens-Hora**. Homens-hora é uma unidade de medida que representa as horas de duração de uma tarefa ao ser executada por um único recurso. Quando executada por mais de um recurso, o esforço estimado da tarefa deve aumentar devido ao *overhead* de comunicação entre os membros da equipe alocada. As setas sinalizam suas interdependências. Nesse exemplo, a tarefa T_4 depende da conclusão de T_2 para que ela possa ser inicializada (ligação Final-Início) e mais dois dias de latência, ou seja, T_4 só pode ser inicializada pelo menos após 2 dias que T_2 tenha sido concluída. T_3 depende da conclusão de T_1 para que ela possa ser iniciada e depende da conclusão de T_2 para que ela possa ser finalizada (ligação Final-Final). A tarefa T_2 precede T_3 e T_4, mas ela não depende de nenhuma outra. Da mesma forma, T_1 não depende de nenhuma atividade e precede T_3. H_1, H_2, H_3 e H_4 , situados abaixo das tarefas, indicam as habilidades exigidas por cada atividade. Enquanto a atividade T_1 exige apenas a habilidade H_4, T_4 requer pessoas com as habilidades H_1 e H_2. Os tipos de ligação são discutidos na Seção 2.4.

Várias referências têm sido desenvolvidas aplicando heurísticas construtivas com base em **Regras de Prioridade** e **Esquemas de Geração de Escalonamento** (***SGS*** - *Schedule Gene-*

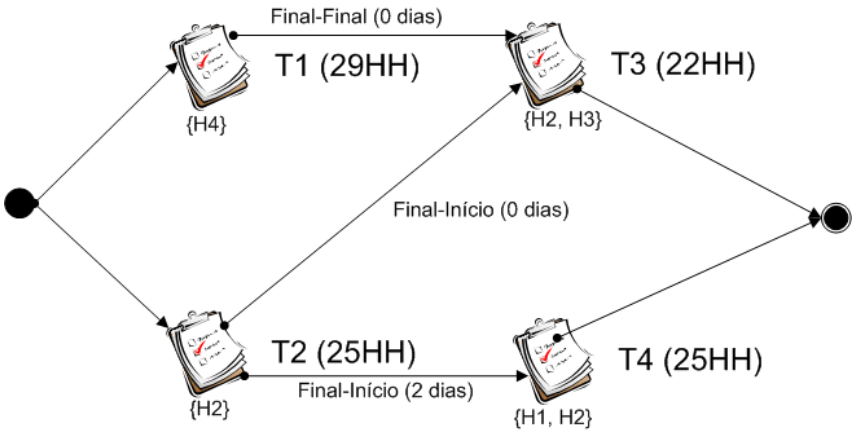


Figura 6: Representação do Diagrama de Rede de um Projeto

ration Schemes) ou **Esquema de Geração de Programa** no RCPSP, conhecido como métodos **X-Pass**. Geração de Escalonamento é uma técnica de montagem do cronograma, em que, a cada iteração, uma atividade é escolhida a partir da Regra de Prioridade definida, e então ela é programada (escalonada). Algumas Regras de Prioridade são: **Tempo de Início mais Cedo** (*Earliest Starting Time - ES*), **Tempo de Término mais Cedo** (*Earliest Finishing Time - EF*), **Tempo de Início mais Tarde** (*Latest Starting Time - LS*), **Tempo de Término mais Tarde** (*Latest Finishing Time - LF*), **Número Máximo de Atividades Imediatamente Sucessoras** (*Most Total Successors - MTS*), **Pior Caso de Folga** - se as atividades não forem escolhidas entre as elegíveis (*Worst Case Slack - WCS*), **Menor Tempo de Processamento** (*Shortest Processing Time - SPT*), **Maior Tempo de Processamento** (*Maximum Processing Time - MPT*), **Número Máximo de Candidatas Subsequentes** (*Maximum Number of Subsequent Candidates - MSC*) e **Seleção Aleatória** (*Random - RAN*).

Existem dois tipos de SGS: o **Serial** e o **Paralelo**. Ambas as gerações constroem o cronograma em várias iterações, respeitando as restrições de recursos e precedência. Para cada iteração, existe um conjunto de atividades elegíveis e uma é escolhida para ser programada. Atividades elegíveis são aquelas em que todas as atividades precedentes já foram escalonadas e, no caso do SGS Paralelo, que também possam ser programadas na unidade de tempo da iteração sem violar a restrição de recursos. Após a programação da atividade, uma nova iteração é iniciada, e portanto, um novo conjunto de atividades elegíveis é calculado. Esse processo continua até que todas as atividades estejam escalonadas. A diferença entre as gerações é que o SGS

Serial constrói a solução incrementando e sequenciando as atividades elegíveis, programando a atividade selecionada o mais cedo possível respeitando as restrições de recursos e precedência. Enquanto que o SGS Paralelo constrói a solução incrementando o tempo e avaliando todas as atividades elegíveis de iniciar em cada próximo instante de tempo. A unidade de tempo é definida na modelagem do problema e pode ser, por exemplo, em horas, dias, semanas etc. Na modelagem proposta utilizamos essa unidade em minutos.

Segundo (GONçALVES; MENDES, 2001), existem três tipos de sequenciamentos ou classes de programa:

1. **Sequenciamento Semi-ativo**: é um sequenciamento admissível que assegura que cada atividade é iniciada o mais cedo possível, isto é, não existem tempos de inatividade desnecessários.

2. **Sequenciamento Ativo**: é um sequenciamento admissível que assegura que nenhuma atividade pode ser iniciada mais cedo (ser antecipada) sem provocar atraso (adiantamento) em outras atividades ou violar as restrições tecnológicas ou de recursos. Nesse tipo de sequenciamento, não existem tempos de inatividade desnecessários. Os sequenciamentos ativos formam um subconjunto dos semi-ativos, isto é, um sequenciamento ativo é necessariamente semi-ativo.

3. **Sequenciamento Não-Atrasado**: é um sequenciamento admissível que assegura que nenhum recurso fica ocioso quando pode ser utilizado para iniciar a execução de uma atividade. Estes sequenciamentos fazem parte do conjunto dos ativos, logo, também são semi-ativos.

Foi mostrado por (KOLISCH, 1996) que o SGS Serial gera um programa ativo. No contexto de programação de atividades em um *Job-Shop*, (BAKER, 1974) mostra que a classe dos programas ativos contém uma solução ótima. (KOLISCH; SPRECHER; DREXL, 1995) adaptam a definição em (BAKER, 1974) para o RCPSP.

(KOLISCH, 1996) demonstra que o programa gerado pelo esquema paralelo é sem

atraso e (BAKER, 1974) mostra que essa classe é um subconjunto dos programas ativos, e portanto, pode não conter a solução ótima.

Existem dois tipos de métodos X-Pass: o **Single-Pass** e o **Multi-Pass**. O Single-Pass é a heurística mais antiga que utiliza apenas um tipo de geração de escalonamento (serial ou paralelo) e uma única regra de prioridade, gerando apenas uma única solução possível (determinístico). O Multi-Pass é a heurística que pode utilizar mais de um SGS, mais de uma regra de prioridade e mais de uma ordem de programação, gerando múltiplas soluções. Para saber os tipos de ordem de programação ver o item **Representação por Lista de Atividades com Ordem de Programação** a seguir.

As heurísticas utilizadas para a resolução do problema do RCPSP pertencem a uma das duas classes: a classe dos métodos baseados em regras de prioridade ou a classe das abordagens baseadas em metaheurísticas, (KOLISCH; HARTMANN, 1999). Os métodos baseados em regras de prioridade constroem o cronograma selecionando a cada iteração uma das atividades elegíveis indicadas pela regra de prioridade adotada, e aplicando na atividade escolhida algum esquema de geração (serial ou paralelo) até que todas as atividades tenham sido escalonadas. As metaheurísticas partem de uma solução inicial ou um conjunto de soluções iniciais e buscam melhorias até que algum critério de parada seja alcançado. Nesse caso, é necessária uma representação da solução para que elas possam ser manipuladas. Percebe-se que quando usamos metaheurísticas, a regra de prioridade é substituída pela representação da solução. A solução então deve ser decodificada utilizando um Esquema de Geração de Escalonamento. Para obter o instante de início e fim de cada atividade, ou seja, o cronograma por completo, as atividades são escalonadas uma a uma - no menor tempo possível, respeitando as restrições de precedência e recursos - através de um Esquema de Geração seguindo a ordem indicada na representação. Portanto, quando uma atividade estiver sendo escalonada, todas suas predecessoras já terão sido escalonadas (*forward scheduling*).

Alguns tipos de representação de solução do Problema de Escalonamento de Tarefas para as metaheurísticas são:

- **Representação por Lista de Atividades**: uma solução é representada por uma lista de

atividades precedente-factível. Ela pode ser mostrada como um vetor do tamanho da quantidade de atividades no projeto, em que o índice do vetor indica a ordem que a atividade deve ser escalonada. Assim, a primeira tarefa a ser escalonada é a tarefa contida na primeira célula do vetor. Cada atividade pode aparecer em qualquer posição do vetor após todos seus predecessores. No caso do método Serial, basta as atividades serem escalonadas seguindo a ordem da lista. Porém com o método paralelo haveria algumas modificações. (HARTMANN, 2002) propõe que em cada iteração do SGS Paralelo, a seleção das atividades elegíveis seja a atividade com menor índice no vetor de atividades. Segundo (SIMõES, 2004), uma séria desvantagem deste tipo de representação reside no fato que várias listas podem resultar no mesmo programa após a decodificação. A Figura 7 exibe um vetor em que o conteúdo da célula é a **T**arefa e o índice é a ordem de programação.

- **Representação por Lista de Atividades com Ordem de Programação**: é a representação igual à apresentada acima com a diferença que o último elemento do vetor informa o sentido em que a lista de atividades é escalonada: para frente (**Progressivo** - ***Forward Scheduling***) ou para trás (**Regressivo** - ***Backward Scheduling***). No método progressivo, quando uma atividade é escolhida para ser escalonada, todos seus predecessores devem já ter sido escalonados e a atividade é escalonada em seu instante de início mais cedo possível. A primeira atividade escolhida para ser escalonada é a primeiro do vetor. No escalonamento regressivo, quando uma atividade é escolhida para ser escalonada, todos seus sucessores devem já ter sido escalonados e ela é escalonada com seu instante de início o mais tarde possível. A primeira atividade escolhida para ser escalonada é a última do vetor. A Figura 8 exibe a Lista de Atividades mais a última célula que indica **F** para *Forward* ou **B** para *Backward*. Essa representação é interessante pelo fato de uma mesma lista de atividades poder construir diferentes escalonamentos com diferentes *makespan*.

- **Representação por Valor de Prioridade** ou **Chave Aleatória** (*Random Key*): é uma representação similar à Lista de Atividades. Nessa representação, cada elemento *i* do vetor contém um número real entre zero e um, indicando a prioridade de escalonamento

da *i*-ésima tarefa (Figura 9). Assim, a programação de cada atividade seguirá a ordem do maior para o menor valor de prioridade. Tanto o método Serial quanto o Paralelo podem ser aplicados nesse esquema de representação. Um fato negativo nessa abordagem é a que para a programação de cada tarefa há a necessidade de percorrer todo o vetor em busca da próxima tarefa de maior prioridade, caso contrário o vetor deve ser previamente ordenado por prioridade, tornando-o similar à representação por Lista de Atividades.

- **Representação Direta**: trabalha diretamente com as unidades de tempo de início e término de cada atividade, não precisado, portanto, de decodificação da representação para geração do programa. Embora essa representação seja bastante objetiva, a principal dificuldade em se trabalhar com ela é que alterações nos seus valores frequentemente resultam em soluções infactíveis. Segundo (VILLELA, 2010), os cálculos necessários para que essas infactibilidades não existam são computacionalmente tão custosos que também não valem a pena serem executados rotineiramente. Também é representada por um vetor do mesmo tamanho da quantidade de atividades do projeto, onde a célula informa o exato instante de início da tarefa e o índice indica a tarefa que inicia naquele instante. Ver Figura 10.

T_1	T_3	T_2	T_4
1^o	2^o	3^o	4^o

Figura 7: Representação por Lista de Atividades

T_1	T_3	T_2	T_4	F/B
1^o	2^o	3^o	4^o	

Figura 8: Representação por Lista de Atividades com Modo de Escalonamento

1	0,5	0,8	0,1
T_1	T_2	T_3	T_4

Figura 9: Representação por Valor de Prioridade

A Figura 11 ilustra uma visão geral dos métodos de escalonamento.

Além de heurísticas, existem também outros tipos de soluções para o RCPSP, tais como métodos exatos baseados em “Branch-and-Bound” e Programação Dinâmica.

1	34	15	40
T_1	T_2	T_3	T_4

Figura 10: Representação Direta

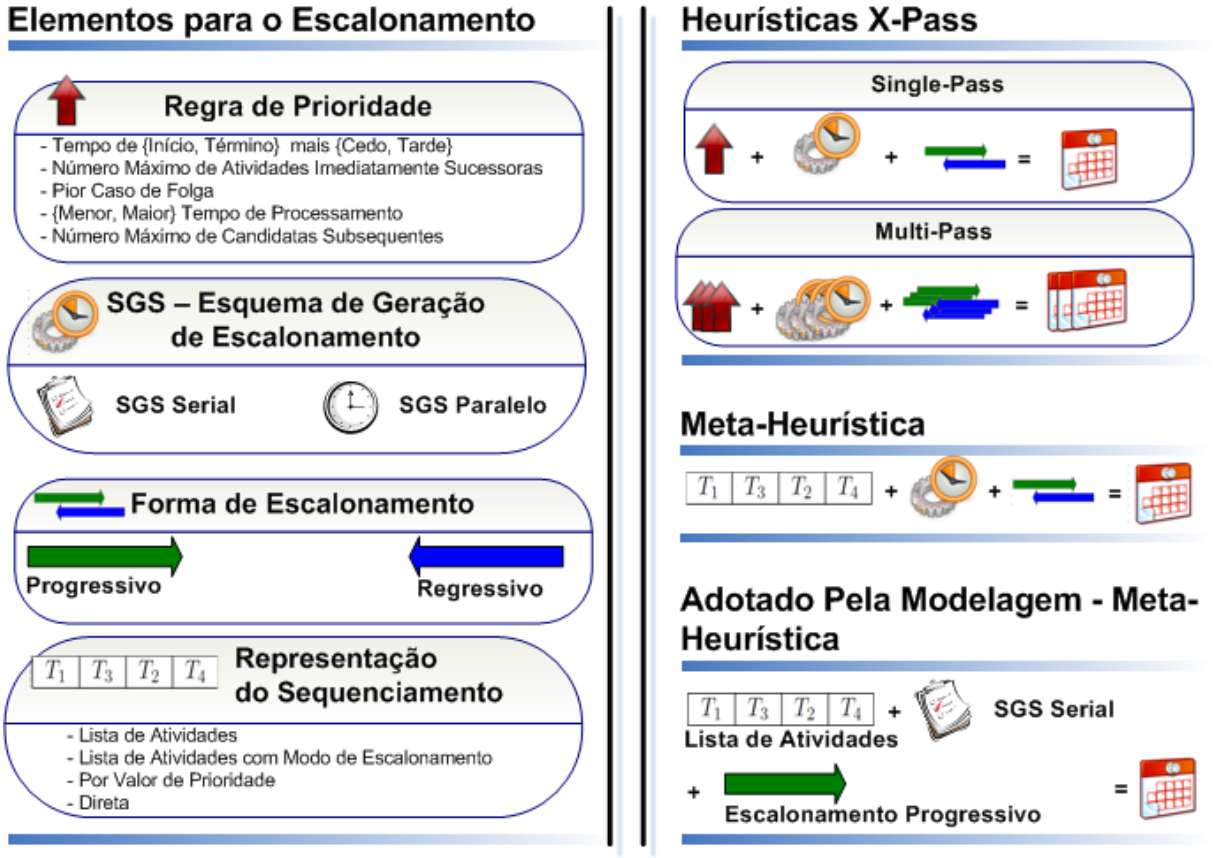


Figura 11: Visão Geral dos Métodos de Escalonamento

2.4 Conceitos Importantes

2.4.1 Tarefas

Tarefa é qualquer atividade necessária para a completude do projeto. O tamanho da tarefa vai depender da forma como os projetistas dividem do projeto. As tarefas podem ter um grau de granularidade muito pequeno a ponto de viabilizar a execução integral dessa tarefa por uma única pessoa. Muitos autores consideram esse grau de granularidade e optam pela alocação de um único recurso por tarefa.

No entanto, muitas vezes é difícil atingir tal grau de granularidade. O projetista pode não conseguir quebrar uma tarefa para formar tarefas filhas para execuções individuais. Pode-

mos citar como exemplo a programação em pares, reuniões de elicitação de requisitos e casos de usos indivisíveis. Além do mais, divisões desse tipo gerariam uma grande quantidade de tarefas de curta duração, o que tornaria entediante ou até mesmo impraticável para o projetista definir tantas tarefas. O que ocorre na prática é o gerente dividir o projeto em atividades de quantidades e tamanhos razoáveis e da mesma natureza (mesma linha de negócio ou raciocínio), ficando a atividade a cargo de um conjunto de pessoas. Outros autores consideram a divisão de tarefas em pequenas macro-atividades ou pacotes de trabalho. Porém, seja qual for o grau de granularidade, a abordagem de alocação de múltiplos recursos é capaz de resolver o Problema de Alocação de Recursos de modo mais condizente com a realidade.

Outro conceito muito importante a ser observado é a dependência entre as atividades. (MODER; PHILLIPS; DAVIS, 1995) define quatro tipos de ligações: **Início-Início** (*Start to Start*); **Início-Final** (*Start to Finish*); **Final-Início** (*Finish to Start*); e **Final-Final** (*Finish to Finish*). Além disso, uma ligação possui associada um tempo de latência, que consiste em atribuir um atraso para o início de uma tarefa sucessora. (SIMõES, 2004) chamam esse tempo de *overhead de preparo da atividade*. Chamamos esse tempo de *latência*, e o *overhead de preparo da atividade* o tempo gasto no preparo da atividade para cada recurso. Se uma tarefa T_b possui uma dependência do tipo Final-Início com a tarefa T_a e uma latência de 10 dias, significa que T_b iniciará no mínimo 10 dias depois que T_a finalizar.

Outra característica pouco utilizada nos trabalhos relacionados é o tipo da tarefa, que pode ser **Concreta**, **Marco** e de **Duração Fixa**. A tarefa do tipo Concreta é aquela em que os recursos são alocados e, quanto maior esforço compartilhado nessa tarefa, menor é seu tempo de conclusão. Consultar Seção 2.4.4 para maiores informações sobre estimação de esforços e *overhead* de comunicação. Como o próprio nome diz, a tarefa do tipo Marco marca a data de um evento significativo no cronograma. Marcos são atividades com duração zero, ou seja, não possuem esforço ou trabalhos associados, e consequentemente não possuem recursos designados. Tratam-se apenas de eventos que precisam ser monitorados, sendo útil para sinalizar a entrega de uma versão do *software* ou de um artefato. Tarefa do tipo Duração Fixa é um tipo de tarefa que mantem fixa sua duração independentemente da quantidade de recursos alocados e não é

gerado *overhead* de comunicação. Além disso, os recursos alocados possuem a mesma dedicação diária na tarefa. Podemos citar como exemplo as reuniões e as elicitações de requisitos, que necessitam tomar um tempo determinado independente da quantidade de pessoas envolvidas. Logo, para uma equipe de duas pessoas que compartilham quatro horas diárias de dedicação ou para uma equipe de dez pessoas compartilhando a mesma dedicação diária de quatro horas, a tarefa será concluída no mesmo prazo.

2.4.2 Recursos Humanos

Recurso é um nome muito genérico que pode ser aplicado praticamente a todos os itens alocáveis em uma atividade. Em outras palavras, é qualquer insumo necessário para a execução de uma atividade. Computadores, papel, madeira, pessoas e dinheiro são alguns exemplos de recursos.

Existem três categorias de recursos: os renováveis, os não-renováveis e os duplamente restritos. Recursos renováveis são aqueles que ao final da execução de uma atividade ou de um período de trabalho estão prontos para serem reutilizados. Por exemplo: computadores e empregados. Alguns deles, como os empregados, ficam disponíveis em sua totalidade a cada novo dia de trabalho. Os recursos não-renováveis são aqueles que estão disponíveis uma única vez em todo o projeto e que vão, de maneira definitiva, se desgastando ou perdendo o seu quantitativo ao longo do tempo. Uma vez que eles são utilizados ou consumidos por completo, eles não voltam a ser reutilizados. São exemplos a tinta, o papel, o dinheiro, o combustível etc. Os recursos duplamente restritos são os limitados duplamente, na soma total e por período.

Existem ainda dois tipos de recursos humanos: o empregado (ou funcionário) e o consultor (prestador de serviço). O empregado possui salário fixo (assalariado ou por contrato de trabalho) e o consultor é pago por horas trabalhadas. Esse tipo de classificação reflete no momento de se calcular o custo do projeto. Como o gasto com empregados é fixo, o que pode variar são as horas extras exercidas por eles. Já para os consultores, qualquer hora trabalhada influencia no custo do projeto.

Outro conceito importante é a **carga horária** individual do recurso. Normalmente as

pessoas trabalham oito horas por dia. Porém existem casos de contratos de quatro ou seis horas diárias, por exemplo. Nas Figuras 3 e 12, a carga horária diária é informado por um círculo em volta da hora de dedicação no eixo da ordenada. No exemplo da Figura 3, R_1 possui carga horária de quatro horas e R_2 possui carga horária de oito horas diárias. No exemplo da Figura 12, o recurso possui carga horária de seis horas.

Além da carga horária, outros conceitos são levados em conta, tais como a quantidade máxima de horas extras por dia para os recursos humanos do tipo empregado, a porcentagem de acréscimo da hora extra de trabalho sobre o valor-hora do empregado e a quantidade de horas diárias de funcionamento da empresa. Os consultores não fazem horas extra, trabalhando no limite de sua carga horária estabelecida. Esse é o motivo da **disponibilidade** de seis horas de trabalho do empregado R_1 nos primeiros 16 dias do cronograma da Figura 3 ser duas horas a mais que sua carga horária diária normal de quatro horas, devido à possibilidade do mesmo exercer duas horas de trabalho extra.

Outro fator importante é a **disponibilidade** de cada recurso ao longo de todo o andamento do projeto. Sabemos que as pessoas podem tirar férias, licenças médicas ou serem emprestadas para outros setores ou projetos. Esses e entre outros motivos tornam o recurso **indisponível** em um determinado período do projeto. E mais, um recurso pode estar indisponível parcialmente por algumas horas diárias durante certo período, não necessariamente o dia todo. Exemplo disso é o caso do funcionário ser liberado duas ou quatro horas do expediente para realizar outra atividade. Ou o caso de um consultor durante certo período poder prestar serviço por apenas algumas horas por dia. Para representar a disponibilidade dos recursos, o projetista atribui (opcionalmente) a cada recurso **períodos de indisponibilidade**, onde cada período contém a data de início e fim e as horas no qual o recurso estará indisponível. No exemplo da Figura 3, o empregado R_1 está disponível seis horas de trabalhos diários até o dia 16, sendo quatro horas de trabalho normais mais duas de extra. Entre os dias 17 e 27 ele está indisponível ao projeto (férias por exemplo). Após esse período ele volta a ter suas seis horas de trabalhos diários disponíveis.

A Figura 12 mostra um exemplo do trabalho de um empregado em um certo projeto.

Perceba que, com o acumulo das tarefas T_2, T_4 e T_5 em um mesmo período, ele irá trabalhar mais que sua dedicação máxima permitida (símbolo vermelho), ou seja, acima do limite de horas extras (linha vermelha tracejada), o que torna essa configuração parte de uma solução inválida. Os pontos de advertência mostram que ele irá trabalhar extra. A dedicação total, representada por uma linha grossa preta contínua, informa o acúmulo de trabalho e horas indisponíveis por dia. Veja que quando o recurso está indisponível quatro horas por dia, ele ainda tem mais duas horas de trabalho normal e mais duas de extra, o que possibilita a alocação em T_7. Note que o funcionário só pôde iniciar T_8 após suas férias (período de 8 horas de indisponibilidade).

A Figura 12 também mostra pontos interessantes. Exemplo disso a dedicação invariável do funcionário em uma tarefa ao longo do tempo, isto é, o recurso permanece alocado em uma tarefa com a mesma dedicação do início ao fim. Outro ponto é que, além dessa figura representar parte de uma solução inválida, por ultrapassar a dedicação máxima permitida, ela aparentemente também mostra que, para esse funcionário, sua alocação não ocorreu de forma ótima. A princípio, uma melhor alocação seria aumentar a dedicação em T_1 para quatro horas ao invés de três horas. Sua conclusão seria mais rápida, anteciparia a inicialização das outras atividades e não haveria custo adicional de horas extras. De forma análoga, a princípio o funcionário poderia se dedicar apenas duas horas em T_7 ao contrário de três horas e, portanto, eliminaria as horas extras sem prejuízo às outras alocações.

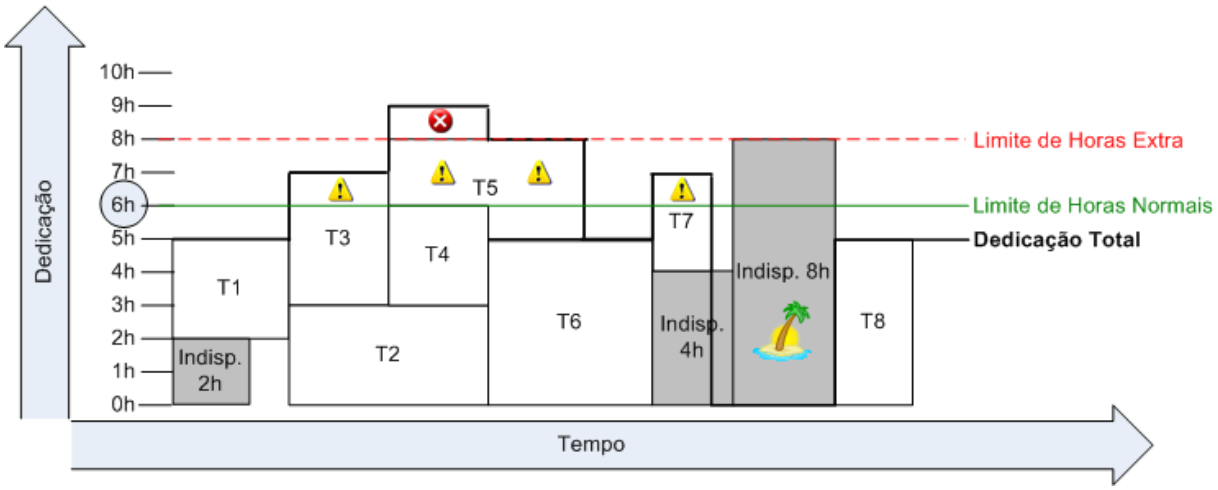


Figura 12: Exemplo do trabalho de um empregado durante o projeto

2.4.3 Habilidades

Uma pessoa só pode ser alocada em uma atividade se ela possuir todas as habilidades requeridas por essa atividade. (BARRETO; BARROS; WERNER, 2008) trabalha tanto no aspecto que um recurso precisa possuir um nível mínimo de intensidade de cada habilidade exigida pela atividade quanto pelo fato do nível da habilidade fazer parte da função objetivo que pretende formar equipes com os maiores níveis de habilidades e assim formar equipes mais capacitadas e, consequentemente, produzindo um produto final de maior qualidade.

Podemos citar como outros critérios de avaliação profissional a capacidade técnica, a atitude ou motivação, o conhecimento e a experiência. Todos esses conceitos juntos formam a **competência** do profissional. Entretanto o critério mais utilizado nas pesquisas é a habilidade, ou seja, o domínio natural de um profissional em resolver bem uma determinada tarefa. Uma pessoa que normalmente trabalha na área de gestão de processos pode ter a capacidade de implementar em uma determinada linguagem de programação, pois ela já estudou sobre o assunto, mas ela não pode ter a habilidade necessária no momento de executá-la.

2.4.4 Estimação de Esforços

Elaborar cronograma tem como pré-requisito a **estimação do esforço** necessário à execução das atividades. Vários modelos têm sido propostos para prever o esforço de se construir e manter um software, tais como **Análises de Ponto de Função** (*FPA - Function Point Analysis*), **Linhas de Código Fonte** (*SLOC - Source Lines of Code*) e **COCOMO** (***CO****nstructive* ***CO****st* ***MO****del*) (WEN; LIN, 2008). É um assunto ainda amplamente pesquisado, porém ele não faz parte do escopo desse trabalho.

Assumimos então que através de um método de estimação de esforço qualquer, como a análise de pontos de função, que considera a funcionalidade implementada sob o ponto de vista do usuário, o gerente conheça o esforço necessário para cada atividade medido em **PF** (**P**ontos de **F**unção). Assumimos também que ele conheça a **estimativa de produtividade** média das pessoas para cada tipo de trabalho. Por exemplo, o gerente pode obter a média em **HH/PF** (**H**omens-**H**ora por **P**onto de **F**unção) de produtividade dos empregados em codifica-

ção na linhagem *Java* através do histórico de projetos anteriores ou através do processo de desenvolvimento de software adotado.

Produtividade é um indicador de eficiência de um profissional ou uma equipe. É um fator que pode influenciar na quantidade de tempo requerido para conclusão de um trabalho. Cada profissional pode ter sua produtividade individual para cada tipo de atividade. Assim, quanto maior a produtividade do funcionário, mais rápido ele é capaz de concluir seu trabalho.

Homem-Hora, entre outras unidades similares como Homem-Mês, é a unidade de medida de trabalho humano que informa as horas aplicadas por uma pessoa em um trabalho. De posse dessas duas medidas - estimativa de esforço e estimativa de produtividade - podemos calcular o tempo estimado em homens-hora t_{hh} da atividade, multiplicando a estimativa de esforço em pontos de função da atividade pela estimativa de produtividade na atividade, expressada pela equação 2.1.

$$t_{hh} = \text{Produtividade (HH/PF)} \times \text{Esforço (PF)} \tag{2.1}$$

Portanto, para uma produtividade de 15HH/PF e um esforço de 50PF, serão necessários 750HH para executar uma determinada atividade. Isso significa que, para uma única pessoa, serão necessárias 150 horas de trabalho nessa tarefa. No entanto, acrescentar mais uma pessoa nessa tarefa de mesma dedicação, produtividade e habilidade, esse tempo não necessariamente cai pela metade, devido ao tempo gasto de comunicação entre eles, chamado de ***overhead* de comunicação**. A **Lei de Brooks** diz que adicionar mão de obra em um projeto atrasado faz com que ele seja mais atrasado (BROOKS, 1995).

Imagine um cenário em que um único recurso trabalhe oito horas por dia em uma tarefa. Agora considere um segundo cenário em que oito recursos de características idênticas ao recurso do primeiro cenário dividam o esforço da mesma tarefa em uma hora por dia cada um. É muito provável que a equipe envolvida do segundo cenário tenha que gastar tempo de controle e comunicação para a execução dessa tarefa, tornando a duração da tarefa maior que a estimada. Precisamos então acrescentar um tempo adicional de comunicação à estimativa de

duração da tarefa quando mais de um recurso é alocado.

Seja N o número de pessoas alocadas em uma atividade e ec o esforço médio de comunicação entre pares de pessoas, um dos possíveis modelos de *overhead* de comunicação, mostrado por (PENTA et al., 2007), calcula o tempo necessário de comunicação t_c como:

$$t_c = ec \times \frac{N(N-1)}{2} \tag{2.2}$$

Assim, se para o exemplo anterior a atividade requeria 750 horas de trabalho para uma pessoa, para uma equipe de N pessoas teremos o acréscimo de t_c. Logo, o novo tempo de duração da tarefa t_n em horas é dado pela Equação 2.3.

$$t_n = t_{hh} \times \left(1 + ec \times \frac{N(N-1)}{2}\right) \tag{2.3}$$

Essa pesquisa utiliza o modelo acima para calcular o tempo gasto na comunicação, mas existem diversos outros modelos. Alguns trabalhos, como em (CHANG et al., 2008), conduziram o problema do *overhead* de comunicação limitando a quantidade de pessoas alocados em uma atividade. Uma desvantagem de limitar o tamanho das equipes é o fato de aumentar consideravelmente a quantidade de soluções inválidas ao logo da execução do algoritmo. O que não ocorre com a abordagem adotada, pois a diminuição da quantidade da equipe apenas aumenta a duração da atividade que refletirá em um dos objetivos do problema, a duração do projeto.

Tabela 1: Comparação de Alguns Trabalhos Relacionados

Característica	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**MP**
Alocação de Recursos	sim	sim	sim	sim	sim	não	não	não	sim
Alocação de Múltip. Rec.	sim	sim	não	não*[1]	sim	não	não	não	sim
Escalonamento de Tarefas	não	não	não	não*[2]	sim	sim	sim	sim	sim
***Overhead* de Comunicação**	não	sim	sim	não	não*[3]	não	não	não	sim
***Overhead* de Preparo de Ativ.**	não	não	não	não	não	não	não	não	não
Tipos de Recurso Humano	sim	sim	não	não	sim	não	não	não	sim
Latência	não	não	não	não	não	não	sim	não	sim
Tipos de Tarefas	não	não	não	não	não	não	não	não	sim*[4]
Habilidade	sim	sim	não	não	sim	não	não	não	sim
Carga Horária Individual	sim	sim	não	não	sim	não	não	sim	sim
Hora Extra	sim	sim	não	não	sim	não	não	não	sim
Salário Individual	sim	sim	não	não	sim	não	não	não	sim
Disponibilidade dos Recursos	não	sim*[5]	não	não	sim*[5]	não	sim	não	sim
Produtividade Individual	não	sim	não	não	sim	não	não	não	não
Treinamento	não	sim	não	não	sim	não	não	não	não
Experiência	não	sim	não	não	sim	não	não	não	não
Tarefas Interdependentes	sim*[6]	sim	sim*[6]	não	sim*[6]	sim*[6]	sim*[6]	sim*[5]	sim
Prioridade nas Tarefas	não	não	não	não	sim	não	não	não	não
Alocações Uniformes	não	não	não	não	não	não	não	não	não
Alocações Mutáveis	não	não	não	não	sim	não	não	não	não
Limites Recursos Alocados p/ Tarefa	não	não	não	não	não	não	não	não	sim
Replanejamentos	não	não	não	não	não	não	não	não	não
Participação do Gerente	não	não	não	não	não	não	não	não	não
Escalonamento Progressivo	não	não	não	não	não	sim	sim	sim	sim
Escalonamento Regressivo	não	não	não	não	não	sim	não	não	não
SGS Serial	não	não	não	não	não	sim	sim	não	sim
SGS Paralelo	não	não	não	não	não	não	não	sim	não
Metaheurística	GA	NSGA-II	SC, TS	GA, SC, TS	GA	GA	BT	GA	GA, NSGA-II
Tipo de Objetivo	MpOP	MpOFP	MnO	MnO	MnO	MnO	MnO	MnO	MpOP, MpOFP
Minimização do Tempo	sim	sim	sim	sim	não	sim	sim	sim	sim*[7]
Minimização do Custo	sim	sim*[8]	não	não	sim	não	não	não	sim
Maximização da Qualidade das Equipes	não	não	não	não	não	não	não	não	sim*[9]

Tabela 2: Legenda da Tabela dos Trabalhos Relacionados

R1	(ALBA; CHICANO, 2007)
R2	(COLARES, 2010)
R3	(QURESHI; HARMAN, 2006)
R4	(ANTONIOL; PENTA; HARMAN, 2005)
R5	(CHANG et al., 2008)
R6	(ALCARAZ; MAROTO, 2001)
R7	(SIMõES, 2004)
R8	(GONçALVES; MENDES, 2001)
MP	Modelo Proposto
$*^1$	Alocação de uma Única Equipe dentre um conjunto de equipes previamente formadas
$*^2$	Ordena a execução dos pacotes de projeto a fim de buscar melhores soluções
$*^3$	Controla o *Overhead* de Comunicação limitando o tamanho da equipe
$*^4$	Tipos de Tarefa: Marco, Concreta, Duração Fixa
$*^5$	Não possibilita a indisponibilidade parcial do recurso
$*^6$	Considera apenas o Tipo de Ligação Final-Início
$*^7$	Na versão mono-objetiva proposta, a função está diretamente relacionada com o tempo do projeto e a quantidade de horas extras
$*^8$	Além do custo, minimiza a quantidade de horas extras
$*^9$	Disponível apenas na versão multi-objetiva da modelagem proposta
MnO	Mono Objetivo
MpOP	Multi-Objetivo - Ponderado
MpOFP	Multi-Objetivo - Frente *Pareto*
GA	Algoritmo Genético
TS	Têmpera Simulada
BT	Busca Tabu
SC	Subida da Colina
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*

Tabela 3: Aspectos Relativos ao Problema de Elaboração de Cronograma

Nome	**Trabalhado**	**Descrição**
Alocação de Recursos Humanos	Sim	Parte do problema de elaboração de cronograma que considera predeterminada a sequência de tarefas.
Alocação de Múltiplos Recursos	Sim	Permite a alocação de mais de um recurso por tarefa.
Recurso Trabalhando em Paralelo	Sim	Um recurso pode trabalhar paralelamente em mais de uma atividade no mesmo dia.
Alocações Uniformes	Não	Condição que procura igualar a distribuição das horas efetivas de trabalho entre os empregados, proporcionalmente a carga horária individual.
Alocações Mutáveis	Não	Possibilita a desalocação de um recurso em uma tarefa em andamento e, se conveniente, realocá-lo em outra tarefa.
Limites Recursos Alocados p/ Tarefa	Sim	Permite o gerente configurar limites mínimos e máximos da quantidade de recursos alocados nas tarefas.
Escalonamento de Tarefas	Sim	Parte do problema de elaboração de cronograma que considera predeterminada as alocações de recursos humanos.
Planejamento Multi-Projetos	Não	Elaboração de Cronograma de múltiplos projetos simultâneos com recursos compartilhados em uma organização.
Replanejamentos	Não	Possibilidade de recalcular o cronograma no decorrer do projeto.
Participação do Gerente	Não	Permite o gerente fixar algumas alocações e algumas sequências entre tarefas.
Tarefas Preemptivas	Não	Possibilidade de pausar uma determinada tarefa e reiniciá-las em outro momento.
Disponibilidade dos Recursos	Sim	Informa o período em que cada recurso estará indisponível ao longo do projeto.
Carga Horária Individual	Sim	Considera que os empregados podem ter carga horária diferenciadas.
Salário Individual	Sim	Os recursos possuem salários individuais.
Hora Extra	Sim	Possibilidade dos empregados trabalharem acima de sua jornada de trabalho e ganharem uma porcentagem a mais por isso.
Multi-Objetivo	Sim	Trabalha com mais de um objetivo simultaneamente, tais como a diminuição do custo e da duração do projeto e aumento da qualidade das equipes.
Qualidade das Equipes	Sim	Alocar as pessoas mais habilidosas para as tarefas certas torna a equipe mais qualificada e eficiente e consequentemente um produto de maior qualidade.
Experiências Profissionais Adquiridas	Não	Característica que envolve o aumento de nível das habilidades das atividades em exercício.
Curva de Aprendizado	Não	Fato que considera o ganho de produtividade do empregado ao longo de seu trabalho.
Treinamento	Não	Treinamento é uma atividade que o empregado é alocado durante certo tempo para ganhar mais tipos e níveis de habilidades.
Produtividade Individual	Não	Fator que considera a eficiência individual de cada profissional em cada tipo de atividade.
Overhead de Comunicação	Sim	Perda de tempo durante a execução de uma atividade para a troca de informações entre os membros da equipe.
Overhead de Preparo de Atividade	Não	Tempo necessário de preparo que um funcionário leva para inicializar uma atividade.
Tarefas com Prioridades Diferenciadas	Não	Para contornar riscos, os gestores costumam priorizar a execução de algumas tarefas.
Interdependência entre Tarefas	Sim	Uma tarefa pode depender de uma ou mais tarefas nos tipos de ligação Início-Início, Final-Início, Início-Final e Final-Final.
Latência	Sim	Tempo de atraso de início de uma tarefa sucessora.
Tipos de Recursos Humanos	Sim	Alguns recursos podem ser empregados assalariados, que ganham seus salários mensalmente, e outros consultores, que recebem por horas trabalhadas.
Tipos de Tarefas	Sim	Classificação das tarefas do tipo Concreta, Marco e Duração Fixa.

3 OTIMIZAÇÃO E ENGENHARIA DE SOFTWARE

Um **Problema** é uma dificuldade que impede que uma vontade seja concretizada. Solucionar problemas exige a capacidade de criar adequadas representações da realidade (**modelos**) e, com ajuda delas, encontrar um **algoritmo de solução** que explique como remover ou superar tal dificuldade.

Existem três Tipos de Problemas:

- Problemas de Decisão: o objetivo desse tipo de problema é verificar a veracidade ou não de uma determinada questão para o problema (“verdadeiro” ou “falso”).
- Problemas de Localização: consiste em identificar uma solução que satisfaça todas as propriedades ou mostrar que ela não existe.
- Problemas de Otimização: consiste em encontrar a melhor solução factível, dentro de todas as soluções localizadas. A melhor solução depende do **critério de otimização**. Para problemas de **minimização**, deve-se encontrar a solução de menor valor e, em problemas de **maximização**, deve-se encontrar o maior valor.

Um problema de otimização possui embutido um problema de localização que por sua vez possui embutido um problema de decisão. A ordem de dificuldade desses problemas é:

Decisão $\leq$ **Localização** $\leq$ **Otimização**.

Quanto aos algoritmos, eles podem ser classificados em **Determinístico** ou **Não-Determinístico**. Algoritmos determinísticos são aqueles que sempre retornam a mesma solução de um problema para os mesmos dados de entrada, enquanto algoritmos não-determinísticos podem retornar, com os mesmos dados de entrada, resultados diferentes a cada vez que o algoritmo é executado.

Os problemas também podem ser classificados como **Indecidíveis**, que não possuem solução computacional e **Decidíveis**, que possuem. Aos problemas decidíveis, podemos ainda

subclassificá-los como **Tratáveis** e **Intratáveis**. (VIANA, 1998) diz que, de uma maneira geral, um problema é considerado tratável quando existe um algoritmo de **complexidade polinomial** que o resolva, e seria intratável quando todos os algoritmos conhecidos para resolvê-lo possuem **complexidade exponencial**, ou **não polinomial**.

Uma observação é que normalmente as **complexidades computacionais** mais importantes escolhidas são o **tempo** e o **espaço**. No entanto, adotamos para esse trabalho apenas o tempo por ser a métrica mais importante para o problema proposto. A complexidade de espaço vem sendo menos preocupante nos últimos anos devido ao aumento da capacidade de armazenamento dos computadores.

Seja ***n*** o tamanho da entrada do problema, ***f*** a função de complexidade do algoritmo e ***O*** a notação usada para expressar a complexidade de tempo de execução do algoritmo, normalmente considerada no pior caso, então ***O(f(n))*** indica que o tempo necessário para obter um resultado do algoritmo é proporcional a $f(n)$. Segundo (VIANA, 1998), complexidade de $O(n^m)$, para $m > 3$, ou $O(x^n)$, para $x > 1$, ou ainda $O(c \cdot n^2)$, para grandes valores da constante c ($c > 10^6$) são indesejáveis. A Tabela 4 classifica e ordena de forma crescente as funções de acordo com suas complexidades.

Tabela 4: Função de Complexidade em Ordem Crescente

Classificação	Ordem de Complexidade	Exemplo
Polinomial	lgN	Pesquisa binária
	N	Função Linear
	$N \cdot lgN$	Ordenação *Quick Sort*
	N^2	Ordenação por inserção
	N^3	Multiplicação de matrizes
Super Polinomial	$N^K, K \geq 4$	
Exponencial ou Não-Polinomial	2^N	Problema do Caixeiro Viajante
	3^N	
	$K^N, N \geq 4$	
	$N!$	
	N^N	

Ainda segundo (VIANA, 1998), problemas de decisão que admitem algoritmos **determinísticos polinomiais** formam a classe **P - Deterministicamente Polinomial**, enquanto que algoritmos **não-determinísticos polinomiais** formam a classe **NP - Não-Deterministicamente Polinomial**.

Se a verificação de factibilidade de uma solução qualquer de uma determinada instância de um problema ocorrer em tempo polinomial, então podemos dizer que o problema pertence a NP. Assumimos aqui que $P \neq NP$, devido ao fato que diversos pesquisadores até hoje não conseguiram elaborar algoritmos polinomiais eficientes para um grande número de problemas.

Um problema é chamado um **NP-Árduo** ou **NP-Difícil** se algum problema NP, tal como o **Problema da Satisfabilidade - SAT**, é redutível polinomialmente a ele, isto é, existe um algoritmo que transforma (reduz) em tempo polinomial qualquer problema NP nesse problema. Isto significa que um problema NP-Difícil é pelo menos tão difícil quanto qualquer problema NP. A classe **NP-Completo** é um subconjunto da classe NP que contém os problemas de decisão de maiores dificuldades (NP-Difícil) e todos os outros problemas de decisão pertencentes a NP-Completo são reduzidos polinomialmente a ele. Existe ainda a classe **NP-Intermediária**, que consistiria por problemas que ninguém conseguiu uma redução polinomial de um problema NP-Completo para eles, onde NP-Intermediária $= NP - (P \cup \text{NP-Completo})$. A Figura 13 mostra uma visão das classes de problemas assumindo a visão de que $P \neq NP$.

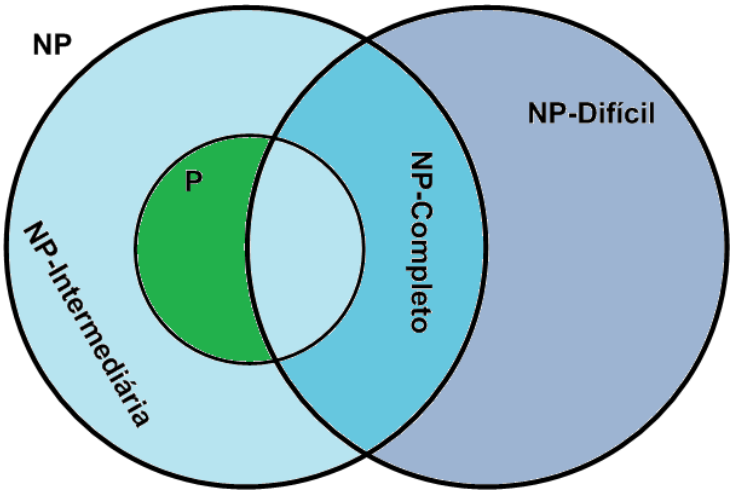


Figura 13: Classes de Problemas

(VIANA, 1998) classifica os algoritmos para resolução de problemas de otimização combinatória, excluindo os algoritmos exponenciais, como:

- **Heurísticas**: Míopes ou Gulosas, Locais e de Partição ou Grupamento;

- **Métodos Enumerativos**: não exaustivos, do tipo "Branch-and-Bound" e Programação Dinâmica;
- **Métodos de Programação Linear e Não-Linear**: Simplex, Pontos Interiores e Algoritmo dos Elipsoides;
- **Métodos Estocásticos**: Têmpera Simulada e Busca Tabu;
- **Métodos Analógicos**: Redes Neurais e Algoritmos Genéticos;

Foi visto no Capítulo 2 que o Problema de Alocação de Equipes e o Problema de Escalonamento de Tarefas pertencem a classe de problemas NP-Difícil. Portanto, dado o fato que o Problema de Elaboração de Cronograma é a junção desses dois problemas, ele é no mínimo tão difícil quanto eles. Logo, podemos considerar que o problema de Elaboração de Cronograma também pertence a classe NP-Difícil.

Para ter uma noção da quantidade de combinações possíveis do Problema de Elaboração de Cronograma, vamos simular a execução de um método exaustivo para calcular todas as combinações possíveis e também desconsiderar as restrições impostas ao problema. Digamos que todos os empregados possuam a mesma carga horária de, por exemplo, 8 horas diárias e mais duas horas de extra, totalizando uma variação de 0 a 10 horas de trabalho diários. Suponha também que não existam períodos de indisponibilidades (não existem férias nem licenças médicas). Sem as restrições impostas, todos os recursos podem ser alocados em todas as tarefas. Seja t a quantidade de tarefas, r a quantidade de recursos, ch a carga horária diária dos funcionários e he o máximo de horas extras por dia que um funcionário pode exercer, o total de combinações para o Problema de Alocação de Equipes C_{pae} é dado por:

$$C_{pae} = (ch + he)^{t \cdot r} \tag{3.1}$$

Desconsiderando também as dependências entre as tarefas, teremos que qualquer ordem de escalonamento entre elas gera uma solução factível. Portanto, o total de combinações para o Problema de Escalonamento de Tarefas C_{pet} é obtido pela seguinte equação:

$$C_{pet} = t! \tag{3.2}$$

Logo, para o Problema de Elaboração de Cronograma, que resolve simultaneamente os dois problemas acima, o total de combinações possíveis C_{pec} é dado pela multiplicação de $C_{pet} \times C_{pae}$, representado pela equação abaixo:

$$T_{pec} = t! \cdot (ch + he)^{t \cdot r} \tag{3.3}$$

Dependendo do tamanho dos dados de entrada do problema, isto é, dependendo do tamanho da **instância**, muitas vezes é impraticável a utilização de métodos exatos para resolver problemas NP-Completos de otimização combinatória devido à grande quantidade de combinações possíveis. Para esse tipo de problema, a solução ideal só pode ser alcançada através de métodos exatos em pequenos projetos. Apesar dos avanços dos métodos exatos para este tipo problema, os tempos computacionais para estes algoritmos em projetos de grande porte podem ser bastante excessivos. Nessa circunstância, torna interessante a utilização de métodos heurísticos, que não necessariamente encontram a melhor solução (**Ótimo Global**), mas que procuram encontrar uma boa solução (**Ótimo Local**) em um tempo computacional aceitável e com poucas exigências computacionais. **Heurística**, do grego "descubro", é qualquer técnica ou método criado para resolver um determinado tipo de problema. As **Metaheurísticas** são métodos heurísticos de uso geral propostos a partir da década de 70 para resolver de forma genérica (em mais alto nível) problemas de otimização. As heurísticas que sempre garantem certa **razão de aproximação** são chamados de **Algoritmos Aproximativos**.

Nos trabalhos mais recentes, avaliados nesta pesquisa, são utilizadas metaheurísticas na busca de soluções ótimas, tais como Algoritmo Genético (GA), Busca Tabu (BT), Têmpera Simulada (TS), entre outras. Algoritmos Evolutivos têm sido aplicados a uma grande quantidade de problemas de otimizações e podem oferecer vantagens significativas na metodologia de solução e desempenho otimizado. O GA e o NSGA-II se encaixam nesse perfil e provavelmente são as metaheurísticas mais conhecidas e utilizadas pela comunidade científica para

resolver problemas mono-objetivos e multiobjetivos, respectivamente.

E por fim, outra classificação seria por tipo de função objetivo:

- **Otimização Mono-Objetiva**: algoritmo que utiliza uma única função com apenas um objetivo, retornando apenas uma solução de valor numérico único. Exemplo: duração do cronograma (*makespan*).
- **Otimização Multi-Objetiva**: algoritmo que utiliza mais de um objetivo. Podem ser de dois tipos:
  - **Consolidado em uma única função**: muito similar à Otimização Mono-Objetiva em que o algoritmo possui uma única função que retorna apenas uma solução de valor numérico único. A diferença é que os objetivos são consolidados na função. Nesse tipo de otimização, pode-se desejar priorizar um objetivo em detrimento de outro, como é o caso do tempo e do custo, que são objetivos conflitantes. Dessa forma, entra-se com os valores dos pesos de cada objetivo de acordo com seu critério. Outra questão importante é a **normalização** desses objetivos, que consiste em equilibrar a ordem de grandeza dos objetivos. Por exemplo, o custo pode estar na escala de milhares de reais e a duração possuir apenas dezenas de dias. Dessa maneira o objetivo de custo monetário poderia dominar o objetivo *makespan*, podendo até mesmo chegar a "sumir" no meio da função objetiva, devido à diferença de magnitude desses objetivos.
  - **Ótimo de Pareto**: nesse tipo de otimização, cada objetivo possui uma função objetivo e o conceito de **dominância** substitui o conceito de "melhor-pior-igual". Ao contrário dos tipos anteriores, em que podemos comparar o valor da função objetivo dado que cada solução possui apenas um único valor numérico, nesse tipo de otimização, cuja solução possui um vetor de valores das funções - um valor para o tempo e um valor para o custo - devemos listar as melhores soluções que não são **dominadas** por nenhuma outra. Entende-se que uma solução S_1 domina uma solução S_2 se S_1 é melhor ou igual a S_2 em todos os valores das funções, e estritamente melhor

em pelo menos um deles. Como o retorno é um conjunto de soluções, denominado de **Frente de Pareto**, deve existir a figura de um tomador de decisão, especialista responsável em escolher, dentre esse conjunto, a solução que mais agrega valor ao negócio.

3.1 SBSE - Otimização em Engenharia de Software

Otimização em Engenharia de Software (*SBSE - Search Based Software Engineering*) é uma área emergente da Ciência da Computação que utiliza técnicas de otimização matemática, geralmente metaheurísticas, para resolver problemas complexos da Engenharia de Software em atividades do ciclo de vida da engenharia de requisitos, planejamento de projeto e estimação de custo.

Um dos primeiros trabalhos realizados utilizando otimização em engenharia de software foi feito por (MILLER; SPOONER, 1976). (XANTHAKIS et al., 1992) foram os pioneiros a publicar a resolução de um problema de engenharia de software utilizando uma metaheurística. Outros trabalhos na década de 90 foram publicados, porém em pequenas quantidades.

O termo SBSE surgiu em (HARMAN; JONES, 2001), trabalho este que pode ser considerado o primeiro *survey* relacionado ao assunto. Após esse marco, SBSE tem recebido uma crescente atenção da comunidade acadêmica em diversas áreas da Engenharia de Software, tais como testes, estimação, refatoração e design, dentre outras. Um recente survey discutindo o presente e o futuro da área é apresentado em (HARMAN, 2007). A Figura 14 ilustra o crescimento quantitativo de SBSE ao longo dos últimos anos desde a confirmação de Harman.

Recentemente (FREITAS et al., 2009) apresentaram o estágio atual de pesquisas em SBSE que aplicam metaheurísticas em problemas da Engenharia de Software. Nesse artigo, os autores exibiram a Tabela 5 indicando trabalhos de aplicação de metaheurísticas em diversas áreas da Engenharia de Software.

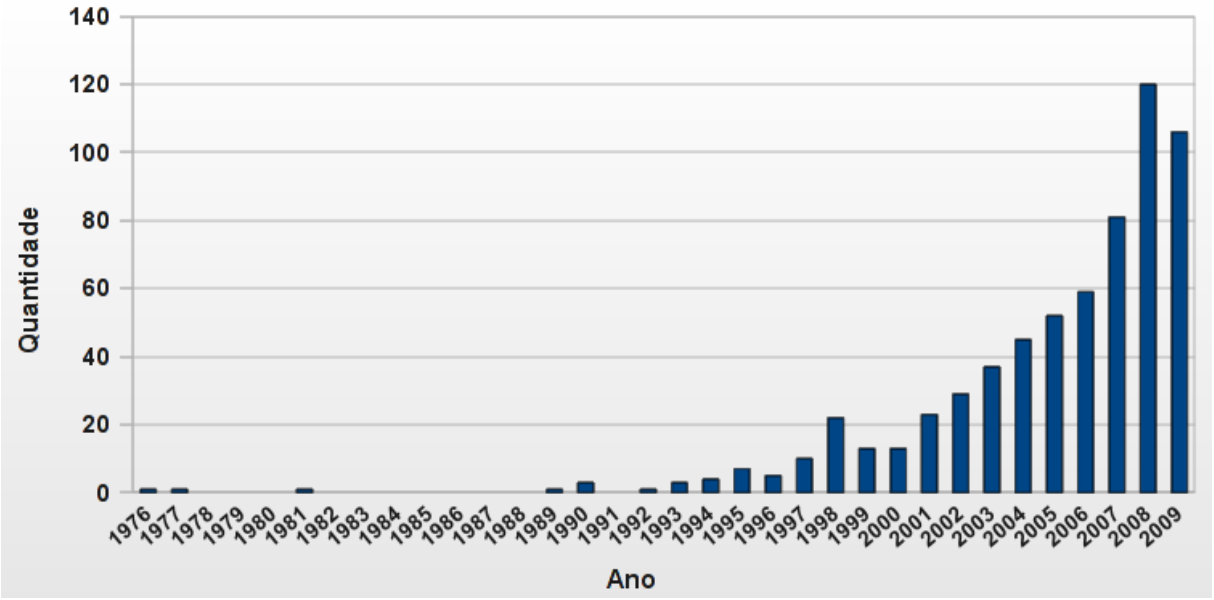


Figura 14: Crescimento quantitativo de SBSE

Tabela 5: Áreas de atuação de SBSE

Área da Engenharia de Software	**Subárea**
Engenharia de Requisitos	Análise de Requisitos Seleção de Requisitos
Teste de Software	Seleção de Casos de Teste Priorização de Casos de Teste Geração de Casos de Teste
Estimativa de Software	Estimativa de Tamanho Estimativa de Custo de Software
Planejamento de Projeto	Alocação de Recursos
Otimização de Código-Fonte	Paralelização
Manutenção de Software	Re-Engenharia de Software *Automated Maintenance*
Engenharia de Software Orientada a Serviço	Desenvolvimento de Software
Otimização de Compilador	Alocação em *Heap* Tamanho de Código

3.2 Algoritmo Genético - GA

O Algoritmo Genético é um método de busca estocástico e evolucionário inspirado na **Teoria da Evolução Natural** de **Charles Darwin**. Introduzido por John Holland em 1975 e popularizado por um dos seus alunos, David Goldberg em 1989, ele tem sido aplicado com sucesso em muitas pesquisas e atualmente é um dos algoritmos mais utilizados para resolver problemas de otimização em problemas NP-Completos.

O GA explora o espaço de busca na tentativa de encontrar boas soluções para proble-

mas de otimização. Ele não efetua uma pesquisa exaustiva em todo o espaço de solução. Ao invés disso o GA inicia com uma população de possíveis soluções (indivíduos) que são manipulados por operações genéticas a fim de evoluírem ao longo de gerações. Cada indivíduo, também chamado de cromossomo, é uma estrutura de dados capaz de representar uma possível solução.

O GA inicia com uma população inicial de N indivíduos (cromossomos), valor que é um parâmetro de entrada do algoritmo e deve ser determinado de acordo com cada problema. Essa população pode ser gerada de forma aleatória ou a partir de conhecimentos prévios do problema. Para o segundo caso, muitas vezes aplica-se uma heurística (algoritmo de baixo nível do problema) com o intuito de gerar uma população inicial de bons indivíduos.

Depois de criada a população inicial, são aplicadas operações genéticas aos indivíduos da população, chamadas de **Cruzamento** (*Crossover*) e **Mutação** (*Mutation*). A probabilidade de um indivíduo sofrer cruzamento e mutação é chamada de **Taxa de Cruzamento -** TX_c e **Taxa de Mutação -** TX_m, respectivamente, e também são fornecidas como entrada no algoritmo. Geralmente TX_c varia entre 60 e 90%, para a evolução ou aprendizado, e TX_m até 5%, para evitar a convergência de ótimos locais.

O **Cruzamento** é um processo de combinação de dois cromossomos escolhidos aleatoriamente, os pais, para a geração de um ou mais novos cromossomos, os filhos. Esse procedimento garante a propagação de material genético e espera-se que, ao longo das gerações, esse procedimento gere filhos mais aptos, isto é, soluções de melhor qualidade (melhoria do *fitness* - valor da função objetivo). Geralmente os pais são substituídos pelos filhos na população, mas uma variante dessa operação é a possibilidade de mantê-los na população.

Existem várias formas de cruzamento, porém os mais conhecidos são o cruzamento de um ponto e o de dois pontos. Nesses cruzamentos, realiza-se um ou dois cortes (dependendo do tipo de cruzamento) em cada cromossomo e combina os materiais genéticos individuais. Também existem cruzamentos aritméticos, que aplicam operações aritméticas entre os genes de mesma posição dos pais. **Gene** é a representação de um valor no cromossomo, como o conteúdo de uma posição do vetor ou uma célula na matriz.

(ALCARAZ; MAROTO, 2001) apresentaram um tipo de cruzamento chamado de ***Precedence Set Crossover*** de um ponto e ***Two-Point Precedence Set Crossover*** de dois pontos. A ideia desta técnica de cruzamento é gerar o filho copiando os genes do pai até o primeiro ponto de corte, partindo sempre do gene da esquerda. Após essa cópia inicial, inicia a cópia dos genes da mãe ignorando os genes já copiados pelo pai, também da esquerda para a direita, até encontrar o segundo ponto de corte. E finalmente o filho recebe do pai os últimos genes que ainda não formam copiados anteriormente, partindo do primeiro ponto até o final do cromossomo. A geração da filha segue processo análogo. Esse tipo de cruzamento garante a geração de filhos com listas de atividades precedente-factível, partindo do princípio que os pais também os são. Esses passos podem ser percebidos no cruzamento da Figura 16.

Da mesma forma que o cruzamento *Precedence Set Crossover* garante gerar listas recendente-factíveis, o cruzamento de dois pontos utilizado na matriz de alocação garante a geração de soluções válidas. Isso é perceptível visto que o cruzamento de dois pontos em matrizes não altera os valores dos genes, o que poderia ocorrer em cruzamentos aritméticos. Outro fato é que o cruzamento de dois pontos mantém os mesmos recursos alocados em cada tarefa, devido à cópia das colunas da matriz, evitando com isso de alguma tarefa não possuir recurso alocado.

As Figuras 15 e 16 representam exemplos de cruzamentos do Problema de Alocação de Equipes e pelo Problema de Escalonamento de Tarefas, respectivamente. Em ambos os cruzamentos foram utilizados cortes de dois pontos.

A etapa de **Mutação** vem logo após a etapa de cruzamento, processo pelo qual cada indivíduo pode sofrer alteração genética aleatória. Ele permite a adição de material genético inédito, e portanto, aumenta as chances de diversificação da população, melhorando, dessa forma, as chances do algoritmo fugir do mínimo (ou máximo) global. Mutação é o nome que se da à **perturbação** em outras metaheurísticas. Através da perturbação, uma solução pode ser "ligeiramente" transformada em outra solução de sua **vizinhança**. (SIMõES, 2004) citaram uma lista de operações de perturbação para o RCPSP na representação por Lista de Atividades, tais como:

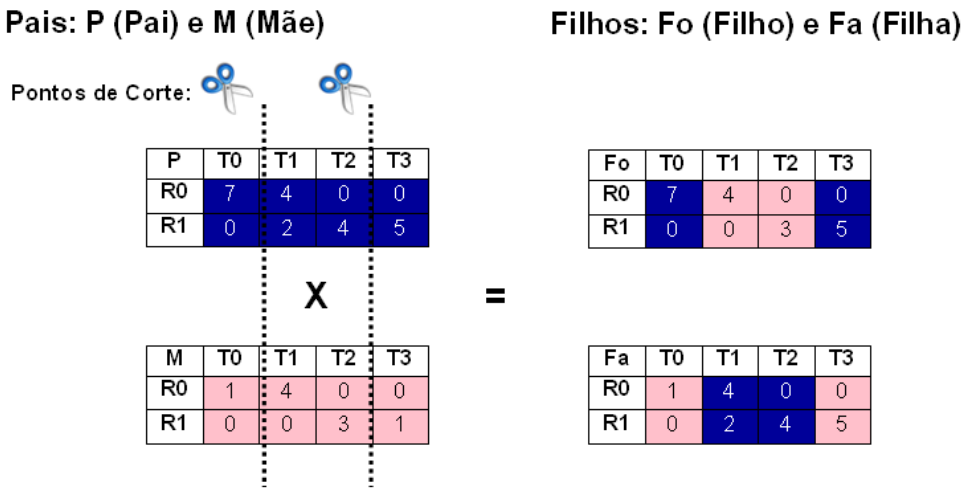


Figura 15: Cruzamento de dois pontos na Matriz de Alocação

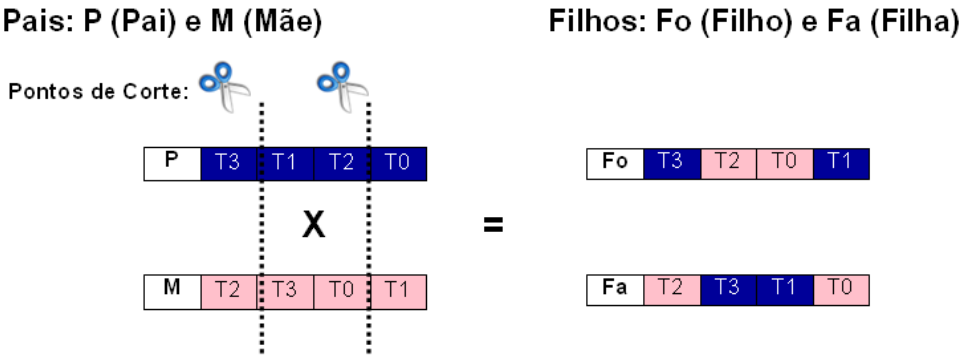


Figura 16: Cruzamento *Two-Point Precedence Set Crossover* na Lista de Atividades

- **Operador Troca**: Duas atividades da lista de atividades são escolhidas e suas posições são trocadas. Tratamentos devem ser tomados nos sucessores e predecessores (imediatos ou não) para preservar a lista precedente-factível

- **Operador Inserção**: É definido pela inserção de uma atividade na posição de outra atividade na lista de atividades. A fim de gerar somente listas precedente-factíveis, a nova posição da atividade escolhida deve ser maior que qualquer de seus predecessores (imediatos ou não) e menor que qualquer de seus sucessores (imediatos ou não).

As Figuras 17 e 18 representam exemplos de mutações do Problema de Alocação de Equipes e do Problema de Escalonamento de Tarefas, respectivamente.

Uma observação importante é o fato do cruzamento ser realizado entre pares de indivíduos à uma taxa TX_c, enquanto a mutação é realizada a uma taxa TX_m para todos os genes

do cromossomo. Dessa forma, um mesmo cromossomo pode sofrer alteração genética em seus genes nenhuma, uma ou mais de uma vez. Perceba que na Figura 17 houve duas mutações e na Figura 18 houve apenas uma mutação.

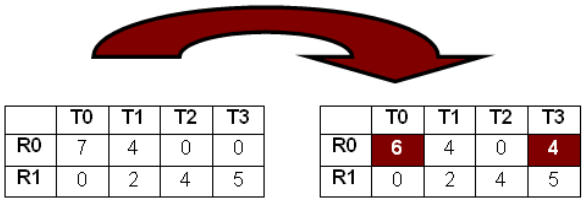


Figura 17: Duas Mutações no Cromossomo da Alocação de Equipes

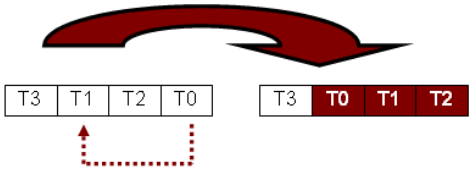


Figura 18: Uma Mutação no Cromossomo do Escalonamento de Tarefas

Após essas operações, a próxima etapa avalia cada indivíduo e o melhor deles é selecionado e armazenado. O algoritmo inicia uma nova iteração ou geração, gerando uma nova população. O processo que gera a nova população a partir da geração corrente é a **Reprodução** que se utiliza de um processo chamado **Seleção**. Essa fase foi inspirada na **seleção natural** dos seres vivos, em que os indivíduos mais aptos (Seleção) terão maiores chances de continuarem vivos na população. Os operadores de seleção são aplicados de forma estocástica, assim, mesmo com uma baixa probabilidade, um cromossomo de baixo *fitness* pode permanecer na população e um cromossomo de alto *fitness* pode não ser selecionado para a próxima geração. Os principais métodos de seleção são:

- **Método da Roleta Viciada**: seleciona um cromossomo com probabilidade proporcional a sua aptidão.
- **Método de Torneio**: seleciona um cromossomo a partir de competições, normalmente envolvendo dois cromossomos (Torneio Binário). No Torneio Binário, dois indivíduos são tomados ao acaso e o melhor deles é selecionado, como pode ser visto na Figura 19.

Figura 19: Torneio Binário

O método da roleta pode ser aplicando utilizando a estratégia apresentada por (VIANA, 1998). A ideia é gerar um **Número de Sorteio** aleatório no intervalo [0,1] e comparar seu valor com a **Probabilidade Acumulada** de cada solução. O indivíduo que tiver o número sorteado entre o **Intervalo de Seleção** será o selecionado pelo sorteio. Perceba que o número de sorteios é igual ao tamanho da população. A Figura 20 mostra a estratégia implementada. Nessa figura, o indivíduo I_1 é mais apto e portanto tem mais chances de ser selecionado e mantido para a próxima geração. Esse mesmo indivíduo foi reproduzido duas vezes, o indivíduo I_2 uma vez e o indivíduo I_3, pelo princípio da seleção natural, não faz mais parte da nova população.

Indivíduo	*Fitness*	1 / Fitness	Prob. Seleção	Prob. Acumulada	Intervalo Seleção
I1	10	0,10	0,527 (52,7%)	0,527	[0 a 0,527]
I2	20	0,05	0,263 (26,3%)	0,789	]0,527 a 0,789]
I3	25	0,04	0,210 (21,0%)	1	]0,789 a 1]

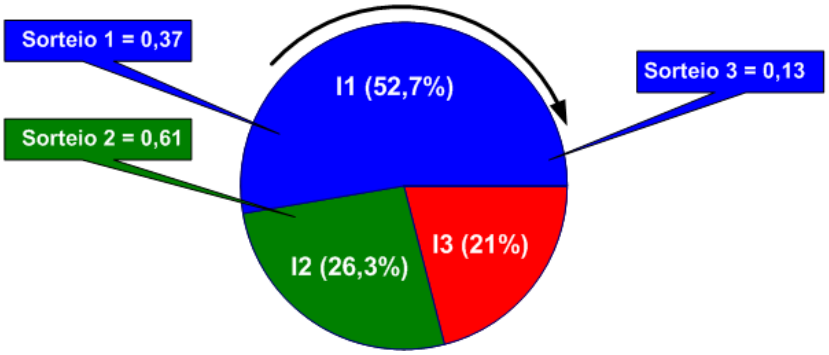


Figura 20: Uma Simulação do Método da Roleta Viciada

Todos esses processos são então repetidos até que o algoritmo atinja um critério de parada. Geralmente esses critérios são:

- **Número de Gerações**: O algoritmo pára após N gerações. N é determinado *a priori*.
- **Qualidade da Solução**: O algoritmo pára quando encontra uma solução de qualidade melhor ou igual a Q. Q é conhecida *a priori*.

- **Ausência de Evolução**: A identidade do melhor indivíduo não é mudada por um número *N* consecutivo de gerações. *N* é determinado *a priori*.

- **Tempo**: O algoritmo pára após ter passado um tempo *T* determinado *a priori*.

Uma boa prática é aplicar a combinação deles de acordo com as necessidades do negócio. O pseudocódigo do GA pode ser lido no Algoritmo 1.

Algoritmo 1 Pseudocódigo do Algoritmo Genético

```
populalacao ⇐ gerarPopulacaoInicial(instancia);
avaliar(populacao);
melhorSolucao ⇐ populacao.melhorSolucao();
repeat
  populacao ⇐ reproducao(populacao);
  cruzamento(populacao);
  mutacao(populacao);
  avaliar(populacao);
  if populacao.melhorSolucao().aptidao() < melhorSolucao.aptidao() then
    melhorSolucao ⇐ populacao.melhorSolucao();
  end if
until condicaoDeParada()
imprimir(melhorSolucao);
```

Perceba que o GA possui vários parâmetros de entrada. Esses parâmetros devem ser bem calibrados a fim de tornar o GA mais eficiente. Há estudos que aplicam GA dentro de outro GA apenas para determinar os melhores parâmetros de entrada. Outros realizam diversos testes computacionais que escolhem os melhores parâmetros de acordo com a instância.

3.3 Non-dominated Sorting Genetic Algorithm II - NSGA-II

Muitos problemas do mundo real apresentam uma coleção de objetivos a serem maximizados/minimizados, que são na maioria das vezes conflitantes entre si, a melhoria de algum(uns) objetivo(s) causa(m) consequentemente a deterioração de outro(s). É o caso do objetivo tempo e custo. Quanto mais recursos disponíveis em um projeto, maior é o custo monetário, contudo quanto mais recursos alocados nas atividades de um projeto (até certa proporção), menor é o esforço dessas atividades e consequentemente menor é a duração do projeto. Portanto,

muitas vezes não existe uma solução ótima única e sim um conjunto de soluções ótimas, porque não existem outras soluções, no espaço de busca, melhores do que estas, quando todos os objetivos são simultaneamente considerados. Essas soluções que não são **dominadas** por nenhuma outra solução são conhecidas como soluções **ótimas de Pareto** ou simplesmente **Frente de Pareto**. Nesse caso, existe a necessidade da figura de um tomador de decisão, especialista responsável em escolher, dentre esse conjunto, a solução que mais agrega valor ao negócio.

Entende-se que uma solução S_1 domina uma solução S_2 se S_1 é melhor ou igual a S_2 em todos os valores das funções, e estritamente melhor em pelo menos um deles. A Figura 21 ilustra o conceito de dominância. Na figura, existem duas funções objetivos que deseja minimizar *f1* e *f2*. Cada quadrado representa uma solução. Percebe-se que a solução *A* é melhor que *B* em relação à *f2*, mas *B* é melhor que A em relação à *f1*. Logo, *A* não domina *B* e *B* não é dominada por *A*. Porém, *C* é dominado por *A*, pois este é melhor que aquele em relação aos objetivos *f1* e *f2* e não pior que nenhum outro objetivo. Da mesma forma, *B* também domina *C* nos dois objetivos. *A* e *B* não são dominadas por nenhuma outra solução e portanto fazem parte da Frente de Pareto.

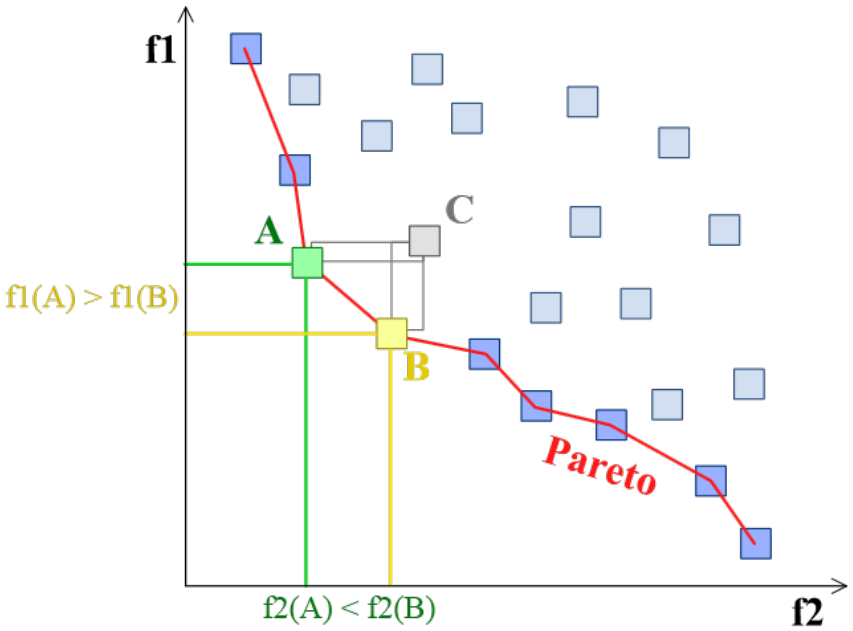


Figura 21: Conceito de Dominância

O Algoritmo Genético de Ordenação Não Dominante (*Non-dominated Sorting Genetic Algorithm - NSGA*) é um algoritmo evolucionário baseado no conceito de **não-dominância**.

Inicialmente proposto por (SRINIVAS; DEB, 1994) e posteriormente por (DEB et al., 2002) em sua versão mais recente, o NSGA-II é provavelmente a mais utilizada meta-heurística para otimização multiobjetivo. Ele é baseado no algoritmo genético clássico apresentado na Seção 3.2, incluindo em suas características conceitos como seleção, cruzamento e mutação.

A ideia básica desse algoritmo é a utilização de um procedimento de ordenação com base em um nível de não-dominância dos indivíduos e a classificação da população total em **Frentes** (*Fronts*) de acordo com o grau de dominância. Uma Frente é o conjunto de soluções de mesmo nível de não-dominância, conforme pode ser visto na Figura 22. A Frente 1 é constituída de todas as soluções não-dominadas. A Frente 2 pode ser conseguida considerando todas as soluções não-dominadas excluídas as soluções da Frente 1. Para determinação da Frente 3, excluem-se as soluções previamente classificadas na Frente 1 e 2, e assim por diante até que todos os indivíduos tenham sido classificados em alguma Frente. Os indivíduos que estão localizados na primeira Frente são considerados as melhores soluções daquela geração, enquanto que na última Frente encontram-se as piores.

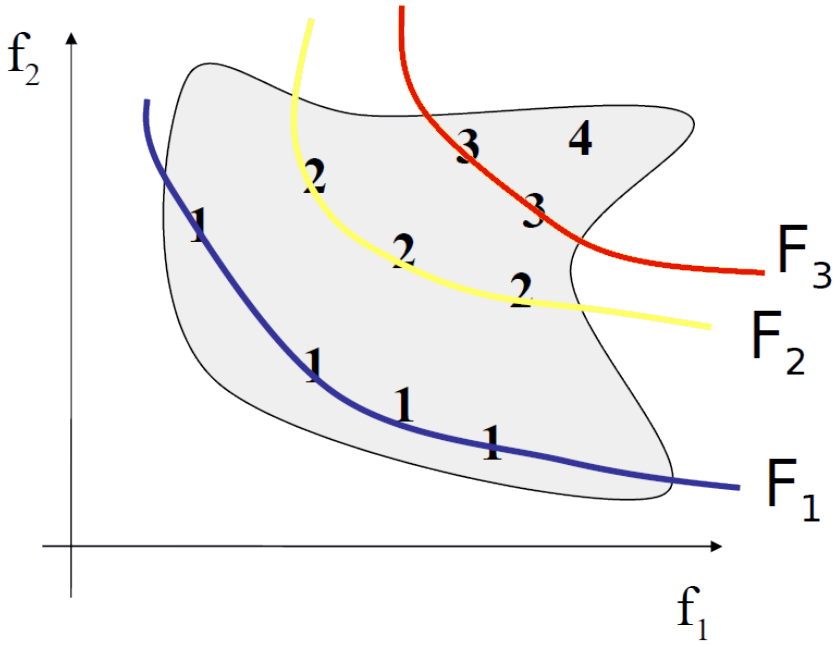


Figura 22: Diferentes Níveis de Frentes

Além de sua grande aceitação na comunidade científica, da existência de boas referências e de seu bom valor de complexidade de execução, alguns dos fatores que levam a escolha

desta metaheurística são sua eficiência computacional e conceitos, tais como elitismo e diversidade das soluções. Apesar das vantagens, a técnica também apresenta desvantagens, tais como limitação do número de soluções encontradas e nas últimas iterações a convergência das soluções pode ser afetada.

O conceito de elitismo quer dizer que a melhor solução encontrada em cada iteração (ou geração) do algoritmo irá ser preservada para a próxima iteração. Embora essa técnica garanta que as melhores soluções encontradas sejam sempre mantidas, existem técnicas de otimização que não utilizam esse conceito, como é o caso do Algoritmo Genético tradicional apresentado na Seção 3.2.

A métrica de diversidade (*crowding distance*) mede o grau de cobertura entre as soluções obtidas e as soluções de extremidade em uma mesma Frente, a fim de garantir um maior espalhamento dos resultados dessa Frente. Esta medida reflete a boa separação das soluções evitando a concentração de soluções em cima de um mesmo ponto ou região. Essa métrica também é utilizada no método de ordenação das soluções de mesma Frente, útil quando se deseja selecionar o melhor indivíduo entre dois de mesma Frente escolhidos pelo método de Seleção Torneio Binário, por exemplo. A Figura 23 exibe a distância de cada solução dentro da mesma Frente.

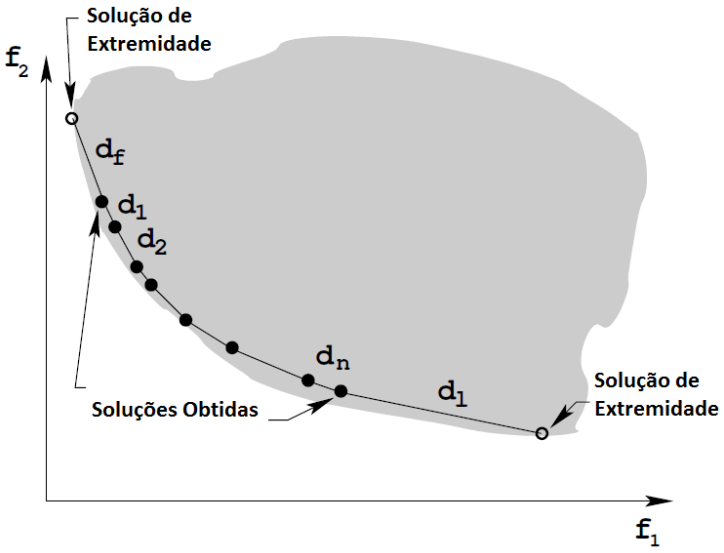


Figura 23: Diversidade de Soluções

O pseudocódigo do NSGA-II pode ser visto no Algoritmo 2. O algoritmo inicia ge-

rando a população inicial de indivíduos (soluções). A população é então submetida à avaliação e à ordenação. A avaliação consiste em calcular os valores das funções objetivo de cada indivíduo. A ordenação consiste em ordenar os indivíduos pela não dominância, isto é, classificar e ordenar os indivíduos por Frentes e em ordenar pela medida de diversidade os indivíduos de mesma Frente. O método de ordenação segue o fluxo indicado pela Figura 24. Para maiores informações de como efetuar a ordenação e classificação dos indivíduos em frente e de como calcular as distâncias de diversidade consultar (DEB et al., 2002) e (MARINHO, 2010).

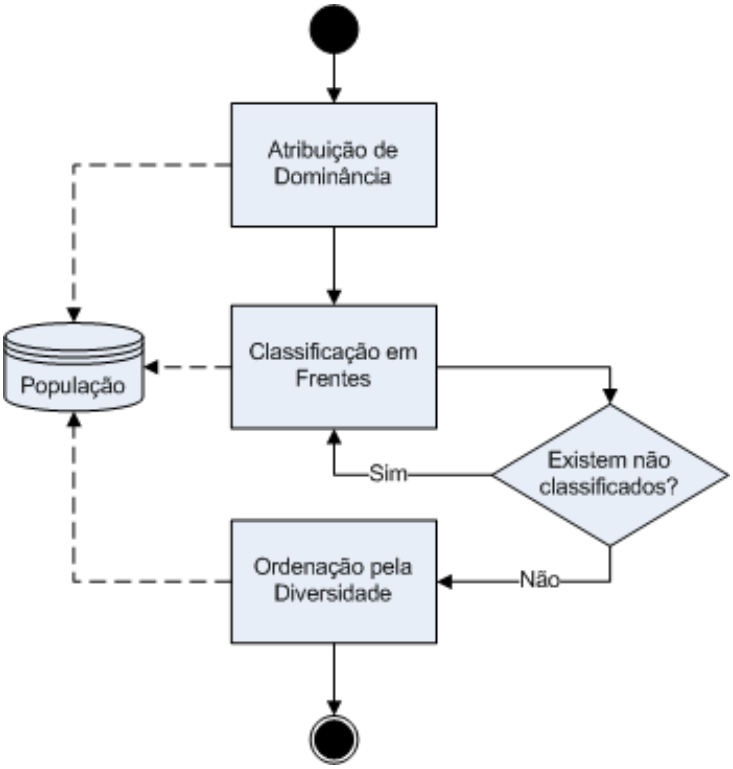


Figura 24: Fluxograma da Ordenação do NSGA-II

Inicia-se então a primeira iteração (geração) de um laço de *G* iterações, onde *G* é definido *a priori*. Um novo laço de *N/2* iterações é iniciado, onde *N* é o tamanho da população. O primeiro passo é a operação de cruzamento, em que dois indivíduos (pais) são selecionados da população atual - geralmente empregando o método de Seleção por Torneio - e cruzados a uma probabilidade previamente estabelecida. Se o cruzamento não ocorrer, os filhos serão a cópia idêntica dos pais. Posteriormente os filhos são submetidos à mutação a uma probabilidade definida previamente. Ao contrário do GA convencional, em que os filhos são substituídos pelos pais, os novos indivíduos são agregados à população anterior pelo conceito de elitismo. Para

isso, a população atual permanece inalterada, permitindo que um mesmo indivíduo seja escolhido para cruzamento ou mutação. Os filhos são adicionados na lista auxiliar *listaDeFilhos*. Após o término do segundo laço, os indivíduos de *listaDeFilhos* serão adicionados na população atual. Novamente a população atual é submetida à avaliação e ordenação. O processo entra na próxima iteração (*g+1*) e termina após *G* iterações. Por fim, o algoritmo imprime a primeira frente da última população. A Figura 25 ilustra uma iteração do algoritmo NSGA-II.

Algoritmo 2 Pseudocódigo do NSGA-II

```
g ⇐ 0;
populacao ⇐ gerarPopulacaoInicial(instancia);
avaliar(populacao);
ordenar(populacao);
for g = 1 to G do
  for n = 1 to N do
    pai ⇐ selecionar(populacao);
    mãe ⇐ selecionar(populacao);
    filho ⇐ pai.copia();
    filha ⇐ mae.copia();
    cruzamento(pai, mae, filho, filha);
    mutacao(filho);
    mutacao(filha);
    listaDeFilhos.adicionar(filho);
    listaDeFilhos.adicionar(filha);
  end for
  populacao.adicionar(listaDeFilhos);
  avaliar(populacao);
  ordenar(populacao);
end for
imprimir(populacao.frente(1));
```

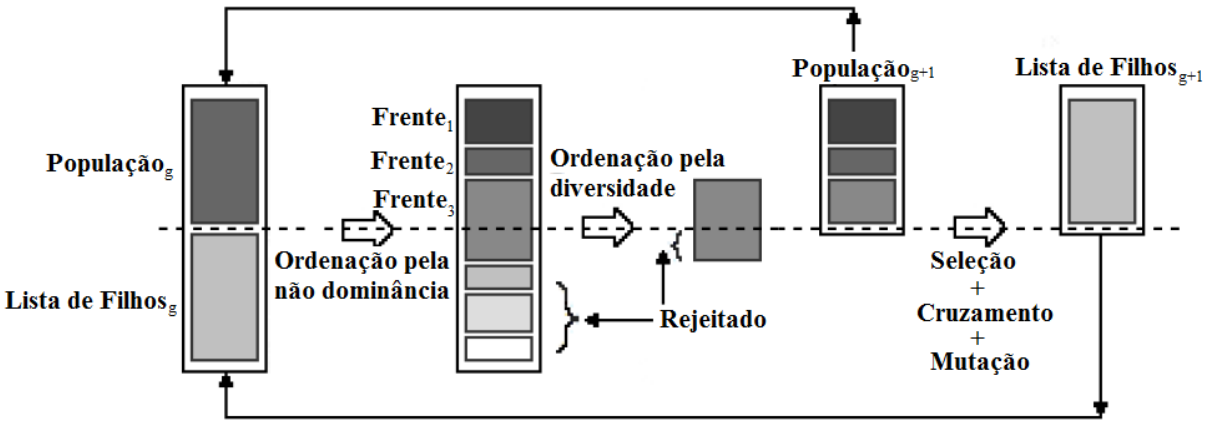


Figura 25: Uma Iteração do NSGA-II

4 MODELAGEM PROPOSTA PARA PROBLEMA DE ELABORAÇÃO DE CRONOGRAMA

Este capítulo apresenta duas versões da modelagem proposta para o Problema de Elaboração de Cronograma. A Seção 4.1 apresenta a versão mono-objetiva da modelagem, onde o Algoritmo Genético é utilizado como base da implementação. De forma análoga, a Seção 4.2 apresenta a versão multiobjetiva, onde é implementado o algoritmo NSGA-II. A maioria dos conceitos incorporados na modelagem já foram apresentados nos capítulos anteriores. Portanto, enfatizaremos como esses conceitos se encaixam e detalhamos todo o funcionamento dos algoritmos implementados. Ambas as versões da modelagem recebem como entrada as tarefas, os recursos e informações sobre o projeto e a empresa amplamente discutidos no Capítulo 2.

Consideramos apenas os recursos humanos devido ao seu papel fundamental na implementação de softwares. Eles também influenciam diretamente na qualidade e no tempo de conclusão do projeto. Além disso, eles geralmente representam a conta de maior custo para um projeto de software e possuem a maioria das características exigidas pelas atividades, tais como habilidades, capacidades técnicas e experiências.

Cada recurso possui salário, habilidades, carga horária e disponibilidade individual. Além disso, eles podem ser do tipo empregado, que recebem por mês e pelo adicional de horas extra, e do tipo consultor, que recebem por hora trabalhada e não cumprem horas extra. A disponibilidade de um recurso é representado pela diferença entre sua carga horária e sua capacidade de exercer horas extra menos as horas de indisponibilidade no período. A representação permite que um recurso esteja indisponível parcial (algumas horas por dia) ou integralmente durante um período. Ver linha grossa contínua no cronograma da Figura 3.

Cada pessoa possui habilidades específicas e são classificadas em níveis. Uma pessoa só pode ser alocada em uma atividade se ela possuir todas as habilidades requeridas por essa atividade, independente de seus níveis de habilidades. A restrição por nível de habilidade não foi considerada dado o fato dela diminuir consideravelmente a quantidade de soluções factíveis. Os níveis das habilidades são utilizados para agruparmos os recursos mais habilidosos em uma

determinada tarefa.

Podemos citar como outros critérios de avaliação profissional a capacidade técnica, a atitude ou motivação, o conhecimento, a produtividade individual e a experiência. Todos esses conceitos juntos formam a **competência** do profissional. Entretanto essa pesquisa lida apenas com a habilidade, ou seja, o domínio natural de um profissional em resolver bem uma determinada tarefa. Uma pessoa que normalmente trabalha na área de gestão de processos pode ter a capacidade de implementar em uma determinada linguagem de programação, pois ela já estudou sobre o assunto, mas ela não pode ter a habilidade necessária no momento de executá-la.

Apesar de sabermos que cada profissional pode ter sua produtividade individual, consideramos que todos os profissionais têm a mesma produtividade, uma média da equipe para cada tipo de atividade informado pelo gerente. Essa pesquisa utiliza a Equação 2.2 do Capítulo 2 para calcular o tempo gasto na comunicação entre os recursos alocados na atividade.

As tarefas possuem suas estimativas de duração, habilidades requeridas e suas interdependências. Elas podem ser do tipo Concreta, Marco ou de Duração Fixa. A duração de uma atividade é estimada considerando ela executada por apenas um único recurso. Caso mais de recurso for alocado nessa atividade, é acrescentado na estimativa de duração o *overhead* de comunicação entre os membros da equipe designada, calculado pela Equação 2.3 do Capítulo 2. As interdependências podem ter os tipos de ligação Início-Início, Final-Início, Início-Final e Final-Final. Além disso, as dependências podem possuir tempo de latência (atraso), em dias, para que a atividade sucessora seja iniciada.

Observamos que muitas vezes é vantajoso atrasar uma tarefa ou interrompê-la (preempção). Entretanto esse trabalho considera que as tarefas não podem ser interrompidas e que elas iniciam o mais cedo possível em que todas as pré-condições estão satisfeitas. Sabemos também que uma das mitigações realizadas no gerenciamento de risco é priorizar a execução das tarefas que possuem maior risco. Entretanto esta pesquisa trata todas as atividades com a mesma prioridade.

Outra característica importante adotada pela modelagem é a possibilidade do gerente

limitar a quantidade mínima e máxima de recursos que uma tarefa pode ser alocada. Essa é uma informação opcional ao projetista, mas útil quando ele deseja limitar, por exemplo, que uma atividade contenha um número excessivo de recursos alocados ao saber que é impraticável dividir e compartilhar essa atividade com muitas pessoas. Ou então de atividades que necessitem de um número exato de pessoas envolvidas, como é o caso de atividades de programação por pares, que envolvem necessariamente dois recursos.

O cromossomo dos algoritmos implementados é representado pela Matriz de Alocação (PAE) e pela Lista de Atividades (RCPSP) apresentados no Capítulo 2 através das Figuras 5 e 7, respectivamente.

O cruzamento escolhido para a Matriz de Alocação foi o de Dois Pontos de Corte. Nesse cruzamento, colunas dos pais são copiadas aos filhos por inteiro, permitindo, dessa forma, a geração de filhos válidos. O cruzamento utilizado na Lista de Atividades foi o *Precedence Set Crossover*, que garante a geração de listas recendente-factíveis. Uma observação importante é que os algoritmos trabalham do princípio ao fim apenas com soluções válidas. As Figuras 15 e 16 do Capítulo 3 representam os respectivos cruzamentos.

A mutação na Matriz de Alocação altera as células (genes) da matriz, mutano as horas de dedicação diária dos recursos nas tarefas. Os algoritmos devem se certificar se a operação gerou um indivíduo inválido. A mutação para a Lista de Atividades desloca a posição de uma tarefa no vetor para a esquerda ou para a direita, respeitando os limites de precedência. Chamamos essa perturbação de Operador Inserção. As Figuras 17 e 18 do Capítulo 3 representam as respectivas mutações.

O método de seleção escolhido na modelagem mono-objetivo foi o Método da Roleta e para a modelagem multi-objetiva, o Método do Torneio Binário, detalhados no Capítulo 3.

Foi escolhido o método X-Pass **Single-Pass**, pelo fato dele ser determinístico e assim ajudar a obter o mesmo cronograma com base na sequência da configuração encontrada. Quanto à forma de escalonamento foi escolhido a **Geração de Escalonamento Serial** por gerar sequenciamentos ativos e assim poder gerar a solução ótima. A representação adotada foi a **Lista de Atividades** devido à eficiência e à fácil aplicabilidade ao SGS Serial. O modo de

escalonamento é sempre o **Progressivo** pela fácil integração com o SGS Serial.

4.1 Versão Mono-Objetiva da Modelagem Proposta

O algoritmo recebe o projeto como instância de entrada, com os recursos e tarefas definidos. Inicia-se gerando a população inicial totalmente aleatória, representando a primeira geração. Os indivíduos da população são representados pelos seus cromossomos através da **Matriz de Alocação** e da **Lista de Atividades**. A Matriz de Alocação define as horas de dedicação diária de cada recurso para cada tarefa, e a Lista de Atividades informa a ordem em que as tarefas serão programadas (escalonadas). A geração da população inicial, assim como em qualquer outra operação do algoritmo, garante sempre gerar ou manter os indivíduos válidos. Logo, não deve haver nenhuma coluna zerada na Matriz de Alocação, isto é, não pode haver uma tarefa com nenhum recurso alocado e a Lista de Atividades deve ser precedente-factível, ou seja, nenhuma atividade pode ser programada antes de todas suas predecessoras.

Cada indivíduo, uma vez que seu cronograma foi elaborado, é uma potencial solução para o problema. A partir da Matriz de Alocação e da Lista de Atividades, a elaboração do cronograma é realizada através do **Esquema de Geração de Escalonamento Serial** e da **Forma de Escalonamento Progressivo**. A elaboração do cronograma é realizada em todos os indivíduos da população inicial e naqueles que foram modificados em alguma operação do algoritmo genético. A construção do cronograma é necessária para que se tenha conhecimento da data de término do projeto (***makespan***) e das horas de trabalho de cada recurso, utilizados no cálculo da função objetivo. A Seção 4.3 detalha uma proposta de melhoria para o Esquema de Geração de Escalonamento Serial, em que as tarefas sucessoras são programadas para qualquer tipo de ligação (Final-Final, Final-Início, Início-Final e Início-Início), e o processo é realizado de forma mais eficiente utilizando um conceito que nomeamos de **Data de Liberação**.

A avaliação da população consiste em calcular a aptidão de cada indivíduo modificado ou, no caso da população inicial, indivíduos ainda não avaliados. O atributo *avaliado* informa se o indivíduo deve ou não ser avaliado. O valor *verdadeiro* é atribuído a todos os indivíduos da primeira geração ou para aqueles que foram modificados em alguma operação do algoritmo.

Após a avaliação de um indivíduo, o valor desse atributo passa a ser *falso* e ele não precisa ser mais avaliado enquanto permanecer inalterado. Durante a avaliação de um indivíduo, seu cronograma deve ser construído para que seja possível conhecer sua aptidão.

A aptidão do indivíduo é obtida pelo cálculo da função objetivo. Essa função é dada pela Equação 4.1 e sua legenda pela Tabela 6. A primeira expressão da equação refere-se ao custo salarial das horas normais trabalhadas pelos recursos humanos do tipo Empregado independentemente da inoperância, isto é, independentemente se o funcionário está sobrecarregado ou ocioso. Para evitar cálculos desnecessários, calculamos previamente a soma dos salários de todos os empregados E por dia no início do algoritmo, pois esse somatório será igualmente reutilizado no cálculo de todos os indivíduos. De posse dessa informação, multiplicamos esse somatório com o total de dias D de duração do projeto (*makespan*) referente ao indivíduo em avaliação. A segunda expressão significa o adicional das horas extra dos Empregados. E finalmente a terceira expressão refere-se ao valor pago por cada hora de trabalho dos recursos humanos do tipo Consultor.

$$\begin{aligned} \text{aptidão} \; = & \Big(\sum_{e\leftarrow 1}^{E} \text{SalárioDiário}_e\Big) \times Makespan + \\ & \Big(\sum_{e\leftarrow 1}^{E} \Big(\sum_{t\leftarrow 1}^{T} \sum_{d\leftarrow 1}^{D} \text{HorasExtra}_{etd}\Big) \times \text{SalárioHora}_e\Big) \times pHE + \\ & \sum_{c\leftarrow 1}^{C} \Big(\sum_{t\leftarrow 1}^{T} \sum_{d\leftarrow 1}^{D} \text{Horas}_{ctd}\Big) \times \text{SalárioHora}_c \end{aligned} \tag{4.1}$$

Tabela 6: Legenda da Função Objetivo

e = Empregado	c = Consultor	t = Tarefa
d = Dia	E = Total de Empregados	C = Total de Consultores
T = Total de Tarefas	D = Total de Dias	pHE = % de acréscimo à hora normal
Makespan = Dia de conclusão (número real)		

Perceba que a aptidão representa o custo do projeto bruto, isto é, o custo pago até mesmo aos empregados, e que esse custo está em função da duração do projeto.

Após a avaliação da população, o melhor indivíduo da geração atual é comparado com o melhor indivíduo de todas a gerações já encontradas e, caso o melhor indivíduo dessa geração

seja melhor que o melhor global, ele é salvo para posterior comparação. Logo após, se o critério de parada for alcançado, o algoritmo retorna, como solução para o problema, o melhor indivíduo global salvo. Caso contrário, o processo continua.

Se o critério de parada não for alcançado, inicia-se então a próxima geração. O primeiro passo é realizar a **Reprodução**, que consiste em gerar a população com base na população da geração anterior, cujos indivíduos mais aptos possuem maiores chances de serem selecionados. O método de **Seleção** escolhido foi a **Roleta Viciada**. A etapa seguinte é executar operações de **Cruzamento** e **Mutação** na população. Esses conceitos foram discutidos no Capítulo 3.

Outra operação realizada na população é a aplicação das heurísticas de melhoria, que consiste em realizar ajustes finos na qualidade das soluções a uma probabilidade previamente determinada (parâmetro de entrada). As heurísticas são:

- Aumentar dedicação sem horas extra: Procura aumentar as horas de dedicação na Matriz de Alocação sem que isso gere horas extra.
- Aumentar dedicação com horas extra: Procura aumentar as horas de dedicação na Matriz de Alocação mesmo que isso gere horas extra.
- Redução de horas extra: Escolhe algumas alocações na Matriz de Alocação e reduz as horas de dedicação se aquelas alocações geraram horas extra. Essas horas extras podem ser visualizadas após a construção do cronograma. Essa heurística pode remover horas de alocação de um recurso a ponto de zerar sua alocação. Nesse caso deve-se conferir se a tarefa possui pelo menos um recurso associado a ela.
- Troca de dedicação entre a equipe: Troca dedicações na Matriz de Alocação entre os membros da equipe de uma tarefa, aumentando a dedicação de uns e diminuindo de outros. Essa heurística pode inclusive passar toda a dedicação de um recurso para outro. Por exemplo, suponha 4 horas de dedicação de R_1 e 3 horas de dedicação de R_2 em uma tarefa qualquer. Após a execução dessa heurística, R_1 pode ficar com 7 horas e R_2 com 0 horas de dedicação.

- Probabilidade de não alocação: Essa heurística é realizada durante a criação da população inicial, mutação da Matriz de Alocação ou qualquer outro procedimento que defina aleatoriamente a dedicação de um recurso em uma tarefa. Sabemos que a dedicação pode variar de zero a carga horária diária máxima do recurso. Nos testes realizados, os recursos possuem carga de 420 minutos (sete horas) mais 120 minutos (2 horas) de extra, totalizando carga máxima de 540 minutos. Nessa caso, a probabilidade de um recurso estar alocado em uma tarefa é muito alta, 540/541 [1..540] contra 1/541 para nenhuma associação [0]. A consequência disso é uma maior probabilidade de formação de equipes numerosas numa tarefa, aumentando consideravelmente o *overhead* de comunicação. A ideia então é aumentar a probabilidade de atribuir zero minutos durante a escolha aleatória de dedicação de um recurso. Utilizamos como padrão 50% de probabilidade de um recurso não estar associado à tarefa e nos outros 50% a dedicação do recurso pode variar entre um e sua carga diária máxima.

Elas são aplicadas seguindo o fluxo da Figura 27. O processo de execução de heurísticas de melhoria inicia com uma probabilidade parametrizada. Uma heurística é escolhida e removida aleatoriamente de uma lista de possíveis heurísticas. A heurística escolhida é aplicada seguindo uma probabilidade parametrizada para aquela heurística. O processo continua até que todas as heurísticas tenham sido removidas da lista.

Por fim, os indivíduos alterados nas operações de cruzamento, mutação ou pelas heurísticas de melhoria são submetidos novamente à avaliação e o processo continua até que o critério de parada seja alcançado. Uma visão geral da modelagem mono-objetiva pode ser vista na Figura 26.

4.2 Versão Multiobjetiva da Modelagem Proposta

Essa seção apresenta a versão multiobjetiva da modelagem. Foram escolhidos três objetivos:

- Minimização da duração do projeto, isto é, a data de término da tarefa mais tarde, conhe-

cido como *makespan*, representado por um número real em dias.

- Minimização do custo do projeto. Diferentemente da versão mono-objetiva, esse objetivo considera apenas o custo das horas extras realizadas pelos recursos humanos do tipo Empregado e pelas horas trabalhadas dos recursos do tipo Consultor.

- Maximização da qualidade das equipes alocadas. Sejam todos os recursos, todas as tarefas e todas as habilidades, essa função objetivo é dada pelo somatório da multiplicação do nível de habilidade pela dedicação total do recurso na tarefa que exige tal habilidade. As funções objetivos encontram-se na Equação 4.2 e a legenda na Tabela 6.

$$
\begin{aligned}
\text{Tempo} &= Makespan \\
\text{Custo} &= (\sum_{e\leftarrow 1}^{E}(\sum_{t\leftarrow 1}^{T}\sum_{d\leftarrow 1}^{D}\text{HorasExtra}_{etd}) \times \text{SalárioHora}_{e}) \times pHE + \\
&\quad \sum_{c\leftarrow 1}^{C}(\sum_{t\leftarrow 1}^{T}\sum_{d\leftarrow 1}^{D}\text{Horas}_{ctd}) \times \text{SalárioHora}_{c} \\
\text{Qualidade Equipes} &= \sum_{r\leftarrow 1}^{R}\sum_{t\leftarrow 1}^{T}\sum_{h\leftarrow 1}^{H}\text{NívelHabilidade}_{rh} \times \text{DedicaçãoTotal}_{rt}
\end{aligned}
\tag{4.2}
$$

A fim de se trabalhar com esses três objetivos independentemente, optamos pelo algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm II*), uma versão multiobjetiva do Algoritmo Genético, apresentada no Capítulo 3. Esta meta-heurística aplica o conceito de **dominância** e, portanto, seu resultado não será necessariamente uma única solução. O que ocorre na prática é obtermos um conjunto de soluções não-dominadas, chamado de **Frente de Pareto**, cabendo a um especialista no assunto decidir qual delas traz maior benefício ao negócio.

O métodos do NSGA-II relacionados à ordenação da população dispostos no *framework* **jMetal** foram incorporados à implementação da modelagem. Esse métodos são a atribuição de dominância, a classificação em frentes e a ordenação de diversidade. jMetal é um *framework* orientado a objeto implementado em Java destinado a facilitar o desenvolvimento de meta-heurísticas para resolução de problemas de otimização multiobjetivos. Para maiores detalhes sobre o *framework* consultar (DURILLO et al., 2006).

Essa versão possui muitos passos idênticos à versão mono-objetiva. Por exemplo, o recebimento da instância de entrada, a geração da população inicial, as operações de cruzamento, mutação e heurísticas de melhoria, a estrutura do cromossomo e a construção do cronograma a partir do cromossomo do indivíduo, da Forma de Escalonamento Progressivo e do Método de Escalonamento de Tarefas Serial proposto. No restante dessa seção, evitaremos reescrever os conceitos por trás desses métodos, concentro apenas naqueles que diferenciam essa versão.

Após o recebimento da instância de entrada, o algoritmo gera a população inicial conforme discutido na seção anterior. O processo de avaliação se diferencia da versão mono-objetiva apenas no cálculo dos objetivos, mantendo a mesma forma de criação do cronograma. Após o cronograma construído, um indivíduo pode ter seu *fitness* calculado. Enquanto a versão mono-objetiva calcula o *fitness* baseado em um único objetivo, o custo, esta versão calcula os três objetivos de forma independente.

A população é então submetida à ordenação, ordenando os indivíduos pela não-dominância e pela diversidade. Esse ponto marca o início da primeira geração. Caso o critério de parada não seja alcançado, o processo continua iniciando um laço de $N/2$ iterações, onde N é o tamanho da população. Esse laço consiste em aplicar as operações de cruzamento, mutação e heurísticas de melhoria nos pais escolhidos de cada iteração. O pai é escolhido através do Método do Torneio Binário, onde dois indivíduos são escolhidos aleatoriamente e o melhor deles será o pai. A mãe segue o mesmo método de Seleção. Definidos o pai e a mãe dessa iteração, o filho e a filha inicialmente serão, respectivamente, as cópias dos pais. Os quatros indivíduos serão submetidos à operação de cruzamento com uma probabilidade previamente definida, e caso a operação realmente ocorra, os filhos serão alterados. Os filhos são submetidos à mutação em cada gene do cromossomo com uma probabilidade atribuída previamente. Enquanto na versão mono os pais são substituídos pelos filhos, nesta versão, pelo conceito de elitismo, os filhos são adicionados a uma lista temporária. Terminado esse laço, os filhos que sofreram alterações são submetidos à Avaliação e adicionados à população corrente.

A população é novamente submetida à ordenação. Perceba que o acréscimo dos filhos na população excederá o limite N do tamanho da população previamente estabelecido. Logo,

o processo de Ordenação descarta os piores indivíduos da população, eliminando os últimos indivíduos da população ordenada. Lembrando que os indivíduos menos aptos são aqueles situados nas últimas frentes e, dentro da última frente eliminada, os que possuem as piores medidas de diversidade. Dá-se então o início da segunda geração. O processo continua até que o ponto de parada seja alcançado. Em caso positivo, o algoritmo retorna a primeira frente da população, conjunto dos melhores indivíduos não-dominados. A Figura 28 ilustra uma visão geral da modelagem multiobjetiva.

4.3 Esquema de Geração de Escalonamento Proposto

O Esquema de Geração de Escalonamento Serial programa cada tarefa no seu tempo de início mais cedo possível, respeitando restrição de precedências e recursos. Porém, esta técnica normalmente é encontrada na literatura apenas para tipos de ligações do tipo Final-Início, quando a atividade sucessora só pode ser alocado após o término mais tarde das predecessoras imediatas.

Outro fato a ser observado é que a técnica de programação serial normalmente não é trabalhada de forma otimizada. Muitos autores programam uma tarefa percorrendo do início da unidade de tempo (em minutos para o nosso caso) até o tempo em que ela pode ser programada. A unidade de tempo horizontal utilizada nessa modelagem é em dias, representando a duração do projeto, e em minutos para a vertical, representando os minutos de dedicação dentro de cada dia. Evitamos a necessidade de percorrer todas unidades de tempo até encontrar aquela em que seja factível de ser programada através de um conceito que chamamos de **Data de Liberação**. Datas de Liberação de um recurso *r* são as datas em que *r* concluiu atividades do tipo *Concreta* ou de *Duração Fixa* (já programadas), as datas de execução das *Tarefas Marco* e as datas em que houve aumento de disponibilidade de *r*. Devemos registrar essas datas em uma lista de forma crescente e sem repetição. Ao iniciar o algoritmo, já temos o conjunto de datas de término dos Períodos de Indisponibilidade e, portanto, ordenamos esse conjunto apenas uma única vez. Posteriormente, para cada tarefa programada adicionamos a data em que cada recurso concluiu sua parte na Tarefa Concreta ou Duração Fixa à lista de Datas de Liberação ou simplesmente

a data de execução da Tarefa Marco, respeitando a ordenação crescente de datas. A ideia é testar se uma tarefa pode ser programada percorrendo apenas as Datas de Liberação, pois, se uma atividade não pode ser programada devido à restrição de superalocação de um recurso, por exemplo, esse quadro só irá mudar após esse recurso concluir outra atividade ou ganhar mais disponibilidade no período.

No último dia de execução de uma atividade, podem sobrar poucas horas de conclusão, não sendo necessário que todos os recursos envolvidos tenham que trabalhar as horas previamente designadas. O algoritmo resolve esse impasse escolhendo aleatoriamente um por um os membros da equipe que irão concluir a tarefa no último dia até que ela seja totalmente concluída, respeitando os limites de alocações anteriormente atribuídos. Logo, no último dia de trabalho, um empregado pode trabalhar menos ou mais minutos que outros. Ou ainda, recursos podem trabalhar apenas no penúltimo dia enquanto outros trabalham no último dia para concluir a tarefa. Esse é o motivo pelo qual o registro da Data de Liberação em Tarefas Concreta é feita pelas datas de conclusão dos recursos e não simplesmente pela data de conclusão da tarefa, pois nem sempre todos os recursos são liberados da tarefa no mesmo tempo. Uma Data de Liberação possui associado a ela um conjunto de recursos. Assim, além de percorrer apenas as Datas de Liberação, procura-se tentar programar apenas nas datas pelas quais está associado o recurso que sofreu a restrição de superalocação.

A Figura 29 ilustra um exemplo de uma programação de um projeto pelo método SGS Serial proposto. Depois que as sete primeiras atividades foram programadas, é a vez de programar T_{cinza}. Ela possui um esforço de 16 Homens-Hora e apenas o recurso R_2 alocado nela durante 4 horas por dia, o que dá uma duração de 4 dias. Ela possui dependência imediata Final-Início em $T_{amarelo}$, Início-Final em T_{azul} com 1 dia de latência, Final-Final em T_{verde} e Início-Início em $T_{vermelho}$. Iniciamos definindo a data de início da primeira tentativa de programar a tarefa. Para isso procuramos pela maior data de término das predecessoras com dependência do tipo Final-Início e pela maior data de início das predecessoras com dependência do tipo Início-Início. A maior dessas datas será a data da primeira tentativa de programação, chamada de *dtInicioLimite*. Para saber do limite mínimo em que T_{cinza} deve terminar, *dtTerminoLimite*,

procuramos pela maior data de término das predecessoras com dependência Final-Final e pela maior data de início das predecessoras com dependência do tipo Início-Final. Definidas as datas limites, iniciamos as tentativas de programação de T_{cinza}. A primeira tentativa de programação gerou uma restrição de superalocação do recurso R_2 no Dia 6. O padrão escolhido é aproveitar o tempo restante de alocação no primeiro dia, alocando a equipe até o final de funcionamento da empresa, respeitando os limites máximos de dedicação de cada recurso previamente definidos. Logo, nessa primeira tentativa de programação, R_2 trabalharia 2 horas no primeiro dia e nos próximos dias seguiria alocando 4 horas por dia. Outra restrição gerada foi de dependência Início-Final em T_{azul}, pois se a tarefa fosse programada desse dia ela terminaria no Dia 9 e Hora 2 é menor que a data de término mínima, *dtTerminoLimite*. O processo continua tentando programar em mais duas Datas de Liberação e novamente foram geradas as duas restrições. Na quarta tentativa, foi gerada apenas a restrição de dependência. A solução é avançar a data de início de programação a fim que a data de término coincida com *dtTerminoLimite*, acrescentando a diferença entre *dtTerminoLimite* e a data de término da última tentativa. Não encontrando mais nenhuma restrição, T_{cinza} pode ser programada no Dia 18 e Hora 6. O Esquema de Geração de Escalonamento Serial proposto é mostrado no Algoritmo 3.

Algoritmo 3 Proposta para Esquema de Geração de Escalonamento Serial

```
for Tarefa t : listaDeAtividades do
  dtInicioLimite = retornarMaiorDataInicio(t)
  dtInicioLimite = retornarMaiorDataTermino(t)
  dtProgramacao ⇐ dtInicioLimite;
  idDtLiberacao ⇐ 0;
  idDtLiberacao ⇐ indiceDataLiberacao(solucao, dtProgramacao, idDtLiberacao);
  resultadoPg ⇐ programar(solucao, t, dtProgramacao, dtTerminoLimite);
  {-1 = programado com sucesso, ≥ 0 = id do recurso que sofreu restrição de superalocação,
  -2 = restrição por dependência}
  while resultadoPg ≠ −1 do
    if resultadoPg ≥ 0 then
      {Restrição por Recurso}
      encontrouDataParaProgramar ⇐ falso;
      repeat
        idDtLiberacao ⇐ idDtLiberacao + 1;
        if idDtLiberacao ≥ solucao.listaDataLiberacao.tamanho then
          dtProgramacao ⇐ DataHora(dtProgramacao.dia + 1, 0);
          encontrouDataParaProgramar ⇐ verdadeiro;
        else
          dtLiberacao ⇐ solucao.listaDataLiberacao[idDtLiberacao]
          if dtLiberacao.listaRecursos.contem(resultadoPg) then
            dtProgramacao ⇐ dtLiberacao.data;
            encontrouDataParaProgramar ⇐ verdadeiro;
          end if
        end if
      until encontrouDataParaProgramar = falso
    else
      {Restrição por Dependência (data de término)}
      dtProgramacao ⇐ dtProgramacao + (dtTerminoLimite − t.dtTermino);
      idDtLiberacao ⇐ indiceDataLiberacao(solucao, dtProgramacao, idDtLiberacao);
    end if
    resultadoPg ⇐ programar(solucao, t, dtProgramacao, dtTerminoLimite);
  end while;
  adicionarDatasLiberacao(solucao, t);
end for
```

Algoritmo 4 Retornar Maior Data de Início

```
dtInicioLimite ⇐ DiaHora(0,0);
predecessoras ⇐ t.predecessoras(TipoDependencia.InicioInicio);
for Tarefa p : predecessoras do
  if p.dtInicio < dtInicioLimite then
    dtInicioLimite ⇐ p.dtInicio;
  end if
end for
predecessoras ⇐ t.predecessoras(TipoDependencia.FinalInicio);
for Tarefa p : predecessoras do
  if p.dtFinal < dtInicioLimite then
    dtInicioLimite ⇐ p.dtFinal;
  end if
end for
retornardtInicioLimite;
```

Algoritmo 5 Retornar Maior Data de Término

```
dtTerminoLimite ⇐ DiaHora(0,0);
predecessoras ⇐ t.predecessoras(TipoDependencia.FinalFinal);
for Tarefa p : predecessoras do
  if p.dtFinal < dtTerminoLimite then
    dtTerminoLimite ⇐ p.dtFinal;
  end if
end for
predecessoras ⇐ t.predecessoras(TipoDependencia.InicioFinal);
for Tarefa p : predecessoras do
  if p.dtInicio < dtTerminoLimite then
    dtTerminoLimite ⇐ p.dtInicio;
  end if
end for
retornardtTerminoLimite;
```

Algoritmo 6 Adicionar Datas de Liberação em Ordem Crescente

```
if t.tarefaMarco then
  adicionarOrdenado(solucao.listaDataLiberacao, t.dtTermino);
else
  for Recurso r : t.equipe do
    dedicacao ⇐ solucao.alocacao(t.dtTermino.dia, t, r);
    if dedicacao > 0 then
      {O recurso trabalhou no último dia}
      dtTerminoRecurso ⇐ Data(t.dtTermino.dia, dedicacao);
    else
      dedicacao ⇐ solucao.alocacao(t.dtTermino.dia − 1, t, r);
      if dedicacao > 0 then
        {O recurso trabalhou no penúltimo dia}
        dtTerminoRecurso ⇐ Data(t.dtTermino.dia − 1, dedicacao);
      end if
    end if
    adicionarOrdenado(solucao.listaDataLiberacao, dtTerminoRecurso, r.id);
  end for
end if
```

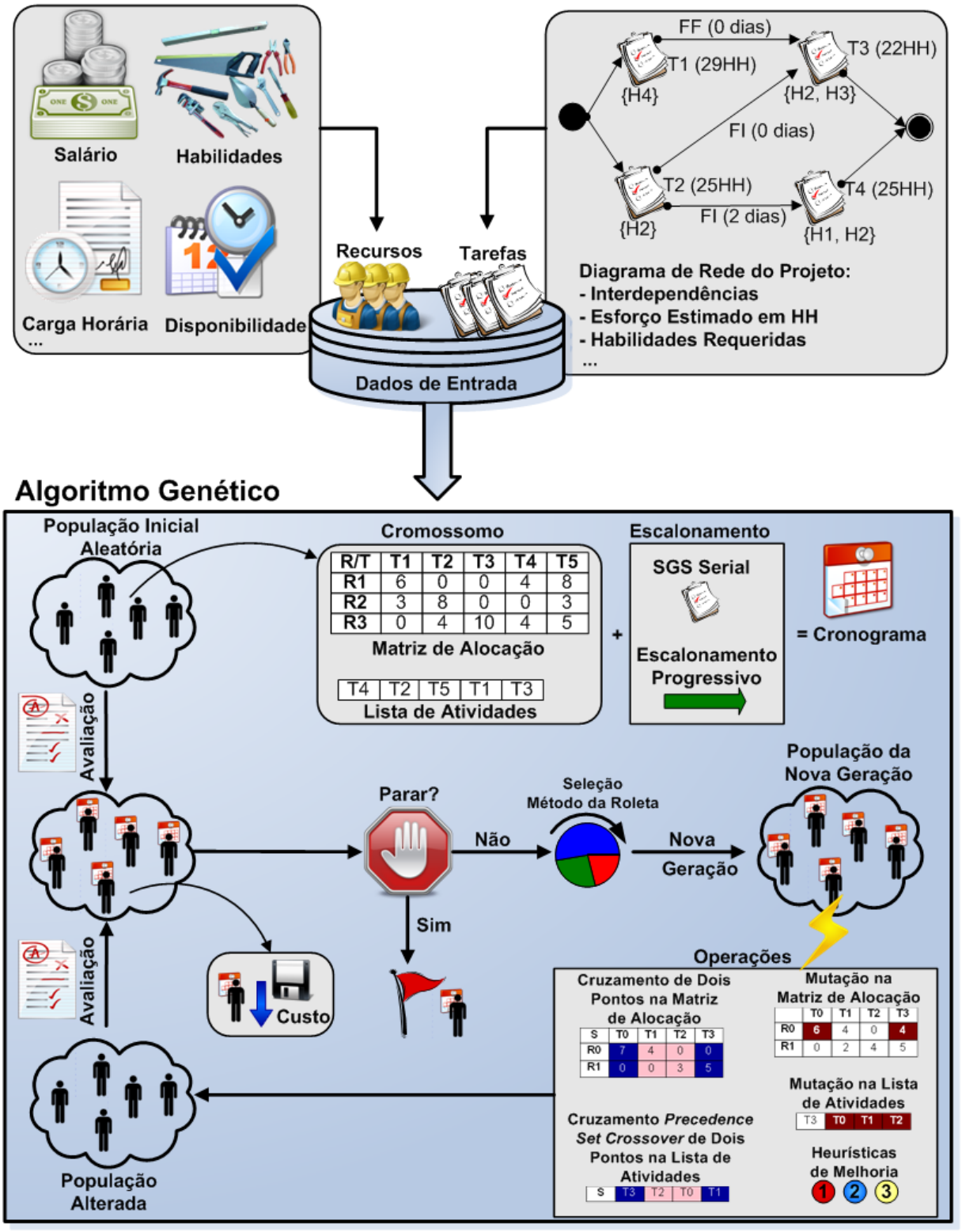


Figura 26: Visão Geral da Modelagem Proposta - Versão Mono-Objetiva (GA)

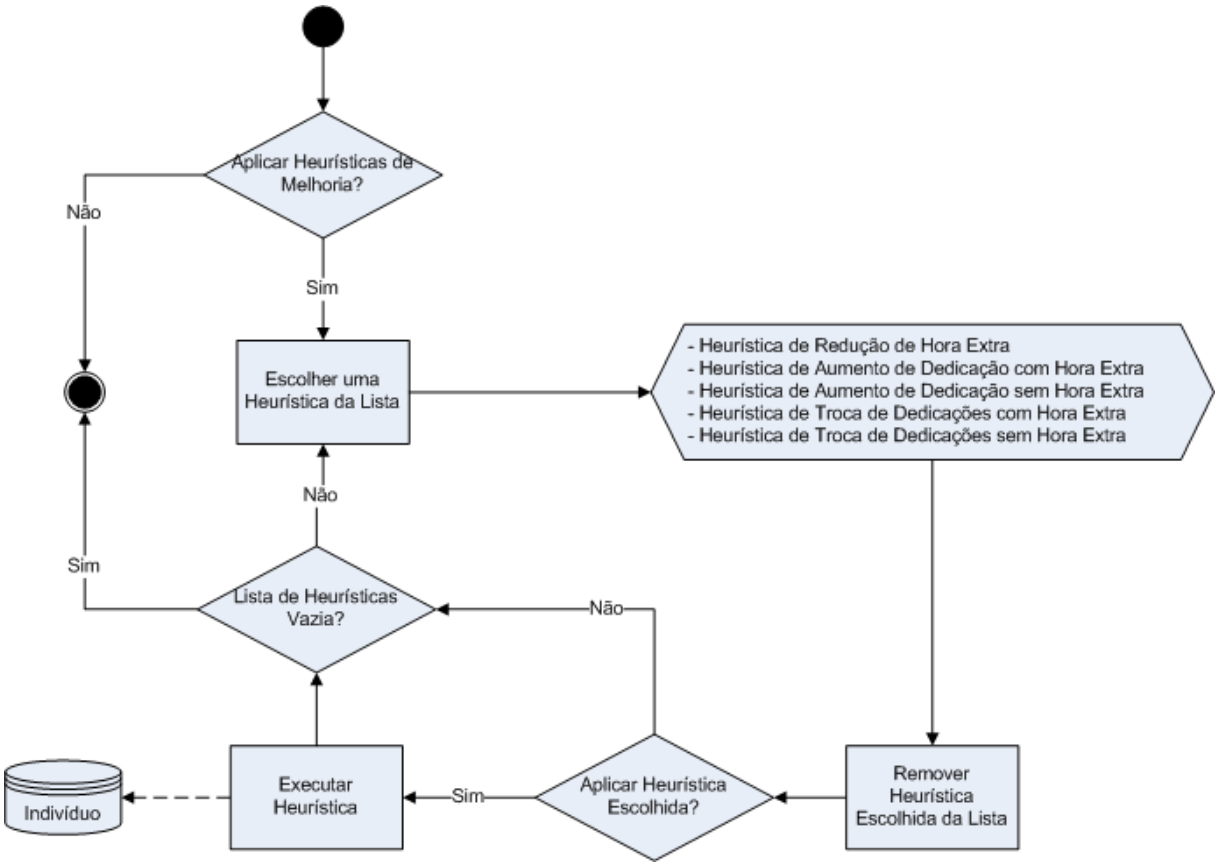


Figura 27: Fluxograma da Aplicação de Heurísticas de Melhoria

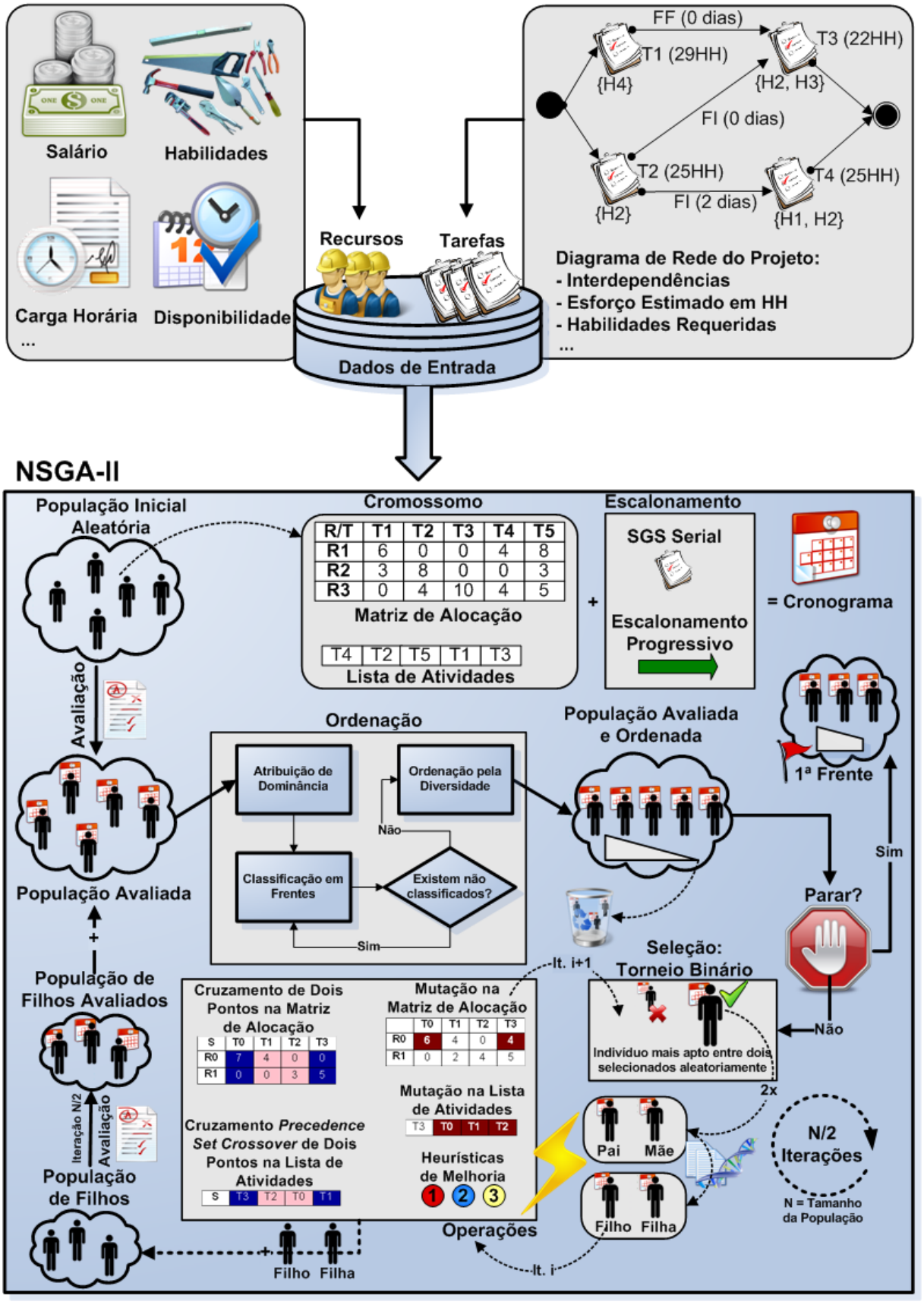


Figura 28: Visão Geral da Modelagem Proposta - Versão Multiobjetiva (NSGA-II)

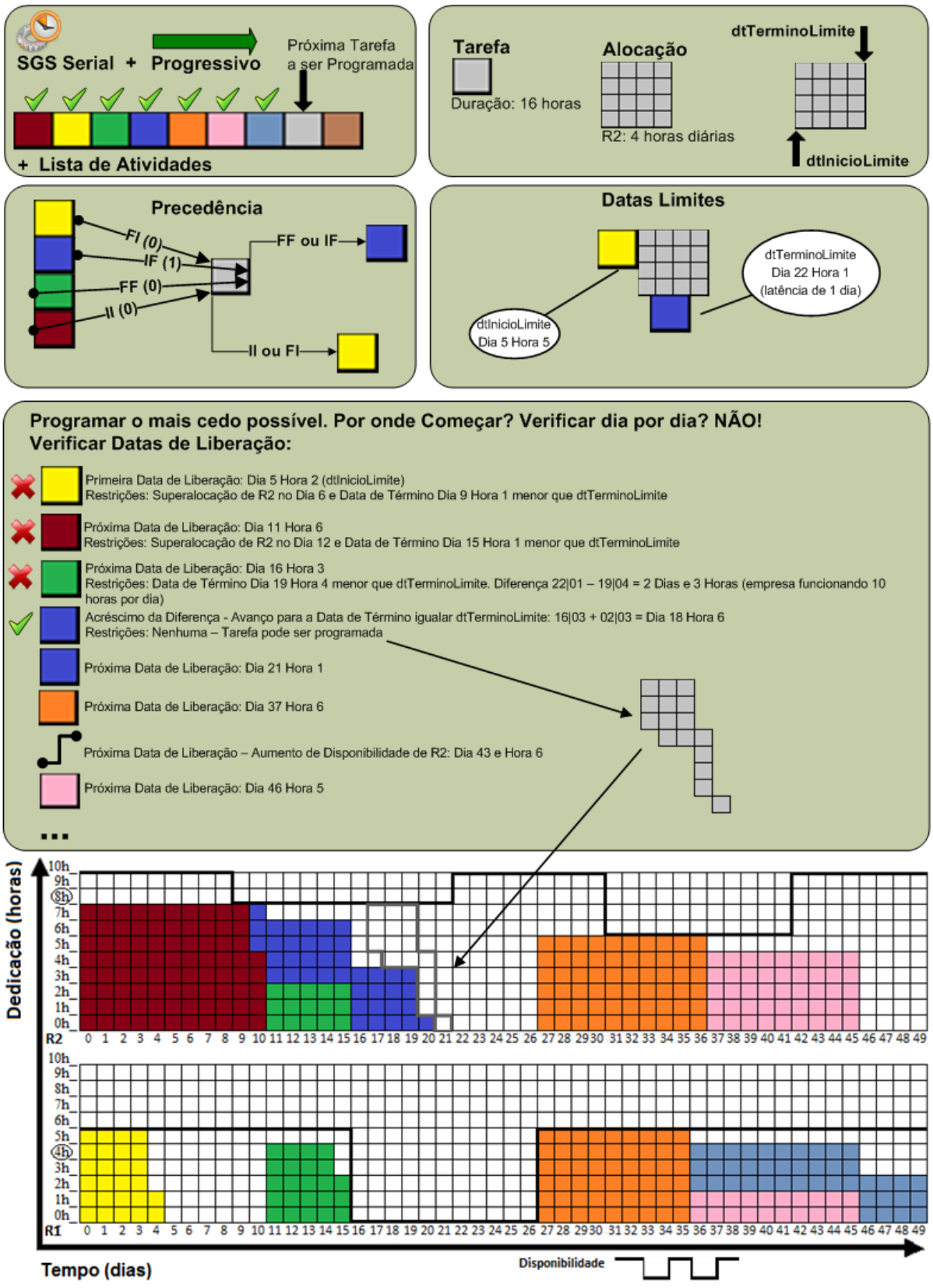


Figura 29: Exemplo de Programação Utilizando o Esquema de Geração de Escalonamento Serial Proposto

5 RESULTADOS COMPUTACIONAIS

5.1 Projeto Sigecom

A modelagem proposta foi aplicada em uma das iterações do projeto Sigecom, um projeto do SERPRO - Serviço Federal de Processamentos de Dados. Nesta empresa, assim como na maioria das fábricas de software, a elaboração do cronograma é realizada baseada no conhecimento do gerente do projeto e nas experiências em projetos anteriores. Esta pesquisa propõe a utilização de uma abordagem otimizada que auxilie o gerente de projeto na construção de um cronograma "ideal" (ótimo) ou próximo do ideal. Na versão mono-objetiva, o cronograma ideal é aquele com menor custo financeiro e na versão multi aqueles com menor duração, menor custo financeiro e maior qualidade na formação das equipes.

As Figuras 30 a 32 exibem o *Cronograma Planejado pelo Gerente - CPG* de uma iteração do Sigecom auxiliada pela ferramenta *Microsoft Project*. Foi adicionado o campo *Overhead de Comunicação* apenas para demonstrar o acréscimo de 5% sobre o esforço estimado a cada par de recursos da equipe referente ao *overhead* de controle e comunicação. Esse campo é apenas para ilustração, pois esse cálculo será efetuado automaticamente durante a execução dos algoritmos. Também foi incluído o campo *Id* que identifica a tarefa, seja ela do tipo Concreta, Marco ou Duração Fixa. Os agrupamentos, tais como *Gestão de Projeto de Software* e *Garantia da Qualidade*, não são tarefas e portanto não possuem identificação. O campo *Ordem de Execução* representa o sequenciamento escolhido pelo gerente, isto é, a ordem de execução das atividades. Esse campo foi alimentado baseado em três critérios: pelas datas de *Início* de execução, pelas *Precedências* e pelas ordens em que aparecem no cronograma (de cima para baixo - Código do Microsoft Project). O primeiro critério é o mais importante, assim, a primeira tarefa a ser programada será aquela com menor data de início. Caso existam tarefas com a mesma data de início, verificam-se as precedências entre elas e, caso não compartilhem nenhuma relação de dependência, a tarefa escolhida será aquela com menor identificação do *Microsoft Project*.

Para realizar os testes computacionais, devemos viabilizar essa iteração do projeto Sigecom à modelagem como instância do problema. Disponibilizamos ao gerente uma interface

	Id	Nome da tarefa	Esforço Estimado	Esf. Estim. Overhead	Ordem Execução	Start	Finish	Predecessors	Resource Names
1		**SIGECOM-093800**	**39137 mins**	**41.385 mins**		**Thu 12/08/10**	**Thu 02/12/10**		
2		**Gestão de Projeto de Software**	**5262 mins**	**5.262 mins**		**Wed 15/09/10**	**Thu 07/10/10**		
3	0	Planejar Projeto	1052 mins	1.052 mins	8	Wed 15/09/10	Thu 23/09/10		Maiza
4	1	Acompanhar Projeto	3684 mins	3.684 mins	12	Thu 23/09/10	Tue 05/10/10	3	Maiza
5	2	Encerrar Projeto	526 mins	526 mins	34	Tue 05/10/10	Thu 07/10/10	4	Maiza
6		**Garantia da Qualidade**	**439 mins**	**439 mins**		**Tue 28/09/10**	**Mon 01/11/10**		
7	3	Planejar revisões	60 mins	60 mins	16	Tue 28/09/10	Tue 28/09/10	3FS+3 days	Selma
8	4	Revisão Outubro/10	379 mins	379 mins	52	Wed 27/10/10	Mon 01/11/10	7FS+20 days	Selma
9		**Gestão de Configuração**	**1530 mins**	**1.530 mins**		**Fri 10/09/10**	**Thu 02/12/10**		
10	5	Criar Ambientes de Configuração	90 mins	90 mins	2	Fri 10/09/10	Fri 10/09/10		Maiza
11	36	Gerar baseline requisitos SIGECOM-093800-01_00-REQ-00	30 mins	30 mins	18	Tue 28/09/10	Tue 28/09/10	40	Rejane
12	37	Auditar baseline requisitos SIGECOM-093800-01_00-REQ-00	240 mins	240 mins	33	Tue 05/10/10	Tue 05/10/10	11FS+5 days	Jairo
13	38	Gerar baseline A&P SIGECOM-093800-01_00-AeP-00	30 mins	30 mins	40	Mon 11/10/10	Mon 11/10/10	48	George
14	39	Auditar baseline A&P SIGECOM-093800-01_00-AeP-00	240 mins	240 mins	45	Tue 19/10/10	Wed 20/10/10	13FS+5 days	Jairo
15	40	Build 04.01.00 - Testes Ciclo #4.1.0	120 mins	120 mins	42	Tue 19/10/10	Tue 19/10/10	62	George
16	43	Build 04.01.01 - Testes Ciclo #4.1.1	60 mins	60 mins	46	Thu 21/10/10	Thu 21/10/10	64	George
17	46	Build 04.01.02 - Testes Ciclo #4.1.2	60 mins	60 mins	49	Tue 26/10/10	Tue 26/10/10	65	George
18	49	Build e publicação homol 04.01.03 - Testes Ciclo #4.1.3	60 mins	60 mins	53	Fri 29/10/10	Fri 29/10/10	66	George
19	52	Build e publicação homol 04.01.04 - Testes Ciclo #4.1.4	60 mins	60 mins	56	Wed 03/11/10	Wed 03/11/10	67	George
20	65	Build e publicação homol 04.01.05 - Testes Ciclo #4.1.5	60 mins	60 mins	65	Thu 11/11/10	Fri 12/11/10	68	George
21	54	Gerar e promover baseline Implem SIGECOM-093800-04_01_05 para Compilada e	60 mins	60 mins	62	Wed 10/11/10	Wed 10/11/10	77FS+3 days	George
22	68	Promover baseline SIGECOM-093800-04_01_05 para Homologada	30 mins	30 mins	71	Wed 24/11/10	Thu 25/11/10	88FS+3 days	George
23	69	Auditar baseline Implem SIGECOM-093800-04_01_05	360 mins	360 mins	76	Thu 02/12/10	Thu 02/12/10	22FS+5 days	Jairo
24	75	Promover baseline SIGECOM-093800-04_01_05 para Implantada	30 mins	30 mins	74	Tue 30/11/10	Tue 30/11/10	92FS+3 days	George
25		**Engenharia**	**31906 mins**	**34.154 mins**		**Thu 12/08/10**	**Tue 30/11/10**		
26		**Requisitos**	**8282 mins**	**8.654 mins**		**Thu 12/08/10**	**Mon 04/10/10**		
27	6	Elicitar Requisitos	1179 mins	1.179 mins	1	Thu 12/08/10	Mon 23/08/10		Alex[20%];Rejane[20%]
28	7	Aprovação DVP, DVS e RAI 13/08/10	0 mins	0 mins	3	Mon 13/09/10	Mon 13/09/10	27FS+14 days	
29		**Documentar e Analisar Requisitos**	**2916 mins**	**2.916 mins**		**Mon 13/09/10**	**Fri 17/09/10**	**28**	
30	8	Documentar Manter Serviços - SM 259116	1206 mins	1.206 mins	5	Tue 14/09/10	Fri 17/09/10		Sherman
31	9	Documentar Analisar, Ativar/Desativar Serviços - SM 259117	614 mins	614 mins	4	Mon 13/09/10	Tue 14/09/10		Rejane

Figura 30: Cronograma planejado pelo gerente de uma iteração do projeto Sigecom (CPG) - Parte 1/3

	Id	Nome da tarefa	Esforço Estimado	Esf. Estim. Overhead	Ordem Execução	Start	Finish	Predecessors	Resource Names
32	10	Documentar Cancelar, Analisar PFP - SM 259117	570 mins	570 mins	6	Tue 14/09/10	Fri 17/09/10		Rejane
33	11	Documentar Ativar/Desativar Exercícios - SM 259120	526 mins	526 mins	7	Wed 15/09/10	Thu 16/09/10		Rejane
34	12	Verificar Requisitos / Revisão Técnica	1169 mins	1.286 mins	9	Fri 17/09/10	Mon 20/09/10	29	Rejane;Sherman
35	13	Ajustes após revisão	1132 mins	1.245 mins	10	Mon 20/09/10	Wed 22/09/10	34	Rejane;Sherman
36	14	**Marco Entrega Validação Requisitos pelo Cliente**	0 mins	**0 mins**	11	**Wed 22/09/10**	**Wed 22/09/10**	**35**	
37	15	Ajustes após validação pelo Cliente	943 mins	1.037 mins	13	Thu 23/09/10	Fri 24/09/10	36FS+1 day	Rejane;Sherman
38	16	Revisão Ajustes após validação Cliente	472 mins	519 mins	14	Fri 24/09/10	Mon 27/09/10	37	Rejane;Sherman
39	17	**Marco de Entrega de Requisitos por Siscor**	0 mins	**0 mins**	15	**Mon 27/09/10**	**Mon 27/09/10**	**38**	
40	18	**Termo de Aceite Requisitos**	0 mins	**0 mins**	17	**Tue 28/09/10**	**Tue 28/09/10**	**39FS+1 day**	
41		**Gestão Requisitos**	**472 mins**	**472 mins**		**Thu 30/09/10**	**Mon 04/10/10**		
42	19	Atualizar Situação dos Requisitos	236 mins	236 mins	24	Thu 30/09/10	Thu 30/09/10	39FS+3 days	Rejane
43	20	Coletar Métricas de Requisitos	236 mins	236 mins	32	Mon 04/10/10	Mon 04/10/10	39FS+5 days	Rejane
44		**Análise e Projeto**	**1316 mins**	**1.316 mins**		**Tue 28/09/10**	**Thu 30/09/10**	**40**	
45	21	Especificar Modelo de Dados	395 mins	395 mins	19	Tue 28/09/10	Wed 29/09/10		Washington
46	22	Especificar Funcionalidades	461 mins	461 mins	20	Tue 28/09/10	Wed 29/09/10		George
47	23	Revisar Análise e Projeto / Revisão Técnica	131 mins	131 mins	23	Wed 29/09/10	Thu 30/09/10	45;46	Italo
48	24	Ajustes A&P após revisão técnica	66 mins	66 mins	26	Thu 30/09/10	Thu 30/09/10	47	George
49	25	Suporte ao desenvolvimento - AeP	263 mins	263 mins	22	Wed 29/09/10	Wed 29/09/10		Washington
50		**Implementação**	**13975 mins**	**14.746 mins**		**Thu 30/09/10**	**Thu 11/11/10**		
51		**Mapeamentos, scripts e pojos**	**350 mins**	**350 mins**		**Thu 30/09/10**	**Thu 30/09/10**	**44**	
52	26	Mapeamento O/R e Pojos	175 mins	175 mins	26	Thu 30/09/10	Thu 30/09/10		George
53	27	Scripts Banco	175 mins	175 mins	27	Thu 30/09/10	Thu 30/09/10		Washington
54		**Desenvolvimento**	**5905 mins**	**5.905 mins**		**Thu 30/09/10**	**Fri 08/10/10**	**51**	
55	28	Implementar Manter Serviço	2423 mins	2.423 mins	28	Thu 30/09/10	Fri 08/10/10		Jenner
56	**29**	Implementar Analisar Serviço	440 mins	440 mins	31	Fri 01/10/10	Mon 04/10/10		George
57	30	Implementar Ativar/Desativar Serviço	661 mins	661 mins	35	Tue 05/10/10	Thu 07/10/10		George
58	31	Implementar Cancelar PFP	440 mins	440 mins	36	Thu 07/10/10	Fri 08/10/10		George
59	32	Implementar Analisar PFP	881 mins	881 mins	29	Thu 30/09/10	Tue 05/10/10		George
60	33	Implementar Ativar/Desativar Exercícios Serviço	1058 mins	1.058 mins	30	Fri 01/10/10	Fri 08/10/10		Italo
61	34	Revisar Implementação / Revisão Técnica	877 mins	965 mins	38	Fri 08/10/10	Thu 14/10/10	54	George;Jenner
62	35	Ajustes implementação após Revisão Técnica	877 mins	965 mins	41	Thu 14/10/10	Tue 19/10/10	61	George;Jenner

Figura 31: Cronograma planejado pelo gerente de uma iteração do projeto Sigecom (CPG) - Parte 2/3

	Id	Nome da tarefa	Esforço Estimado	Esf. Estim. Overhead	Ordem Execução	Start	Finish	Predecessors	Resource Names
63		**Correção dos Testes**	**5965 mins**	**6.561 mins**		**Tue 19/10/10**	**Thu 11/11/10**		
64	42	Correções verificadas pela eq Req - Ciclo #4.1.0	1579 mins	1.737 mins	44	Tue 19/10/10	Thu 21/10/10	73FF+1 day	George;Jenner
65	45	Correção dos Testes - Ciclo #4.1.1	1930 mins	2.123 mins	48	Fri 22/10/10	Tue 26/10/10	74FF+1 day	George;Jenner
66	48	Correção dos Testes - Ciclo #4.1.2	1403 mins	1.543 mins	51	Tue 26/10/10	Fri 29/10/10	75FF+1 day	George;Jenner
67	51	Correção dos Testes em Homol - Ciclo #4.1.3	351 mins	386 mins	55	Wed 03/11/10	Wed 03/11/10	76FF+1 day	George;Jenner
68	60	Correção pós Homologação #4.1.4	702 mins	772 mins	64	Thu 11/11/10	Thu 11/11/10	85	George;Jenner
69		**Testes**	**6140 mins**	**6.410 mins**		**Tue 28/09/10**	**Wed 17/11/10**		
70	70	Plano e Especificação de Testes	1412 mins	1.412 mins	21	Tue 28/09/10	Mon 25/10/10	40	Leninha
71	71	Avaliar Resultados dos testes	307 mins	307 mins	67	Tue 16/11/10	Wed 17/11/10	72	Leninha
72		**Execução dos Testes**	**4420 mins**	**4.690 mins**		**Tue 19/10/10**	**Tue 16/11/10**		
73	41	Testes iniciais pela equipe req - Ciclo #4.1.0	614 mins	614 mins	43	Tue 19/10/10	Wed 20/10/10	15	Rejane
74	44	Testes - Ciclo #4.1.1	1474 mins	1.621 mins	47	Thu 21/10/10	Mon 25/10/10	16	Lima;Rejane
75	47	Testes - Ciclo #4.1.2	614 mins	675 mins	50	Tue 26/10/10	Wed 27/10/10	17	Lima;Rejane
76	50	Testes em Homologação - Ciclo #4.1.3	614 mins	675 mins	54	Fri 29/10/10	Mon 01/11/10	18	Lima;Rejane
77	53	ReTestes em Homologação - Ciclo #4.1.4	614 mins	614 mins	58	Wed 03/11/10	Fri 05/11/10	19	Rejane
78	66	Testes pós Homologação - Ciclo #4.1.5	491 mins	491 mins	66	Fri 12/11/10	Tue 16/11/10	20	Rejane
79	67	Marco de Entrega de Código	0 mins	0 mins	59	Fri 05/11/10	Fri 05/11/10	77	
80		**Homologação**	**1316 mins**	**1.814 mins**		**Mon 11/10/10**	**Fri 19/11/10**		
81	55	Planejar Homologação	420 mins	840 mins	39	Mon 11/10/10	Thu 14/10/10		Maiza
82	56	Executar script/publicar #4.1.1	536 mins	590 mins	57	Wed 03/11/10	Thu 04/11/10	75FS+3 days	Washington;George
83	57	Publicar após teste equipe #4.1.1	120 mins	120 mins	60	Fri 05/11/10	Mon 08/11/10	76FS+3 days	George
84	58	**Marco de entrega para Homologação # 4.1.1**	0 mins	**0 mins**	61	**Mon 08/11/10**	**Mon 08/11/10**	83	
85	59	**Fim 1a. Homologação pelo Cliente # 4.1.1**	0 mins	**0 mins**	63	**Thu 11/11/10**	**Thu 11/11/10**	84SS+3 days	
86	61	Executar script/publicar #.4.1.2	240 mins	264 mins	68	Wed 17/11/10	Wed 17/11/10	68FF+3 days	Washington;George
87	62	**Marco de entrega para 2a. Homologação # 4.1.2**	0 mins	**0 mins**	69	**Wed 17/11/10**	**Wed 17/11/10**	86	
88	63	**Termo de Aceite Cliente**	0 mins	**0 mins**	70	**Fri 19/11/10**	**Fri 19/11/10**	87SS+2 days	
89		**Implantação**	**877 mins**	**1.214 mins**		**Fri 08/10/10**	**Tue 30/11/10**		
90	64	Planejar Implantação	277 mins	554 mins	37	Fri 08/10/10	Mon 11/10/10		Maiza
91	72	Executar script/publicar	600 mins	660 mins	72	Wed 24/11/10	Thu 25/11/10	90;88FS+3 days	Washington;George
92	73	**Marco de Implantação**	0 mins	**0 mins**	73	**Thu 25/11/10**	**Thu 25/11/10**	91	
93	74	Validar Implantação - QA	0 mins	0 mins	75	Tue 30/11/10	Tue 30/11/10	92FS+3 days	

Figura 32: Cronograma planejado pelo gerente de uma iteração do projeto Sigecom (CPG) - Parte 3/3

gráfica para cadastramento das Habilidades, dos Recursos e das Tarefas, dispostas nas Figuras 33 a 39.

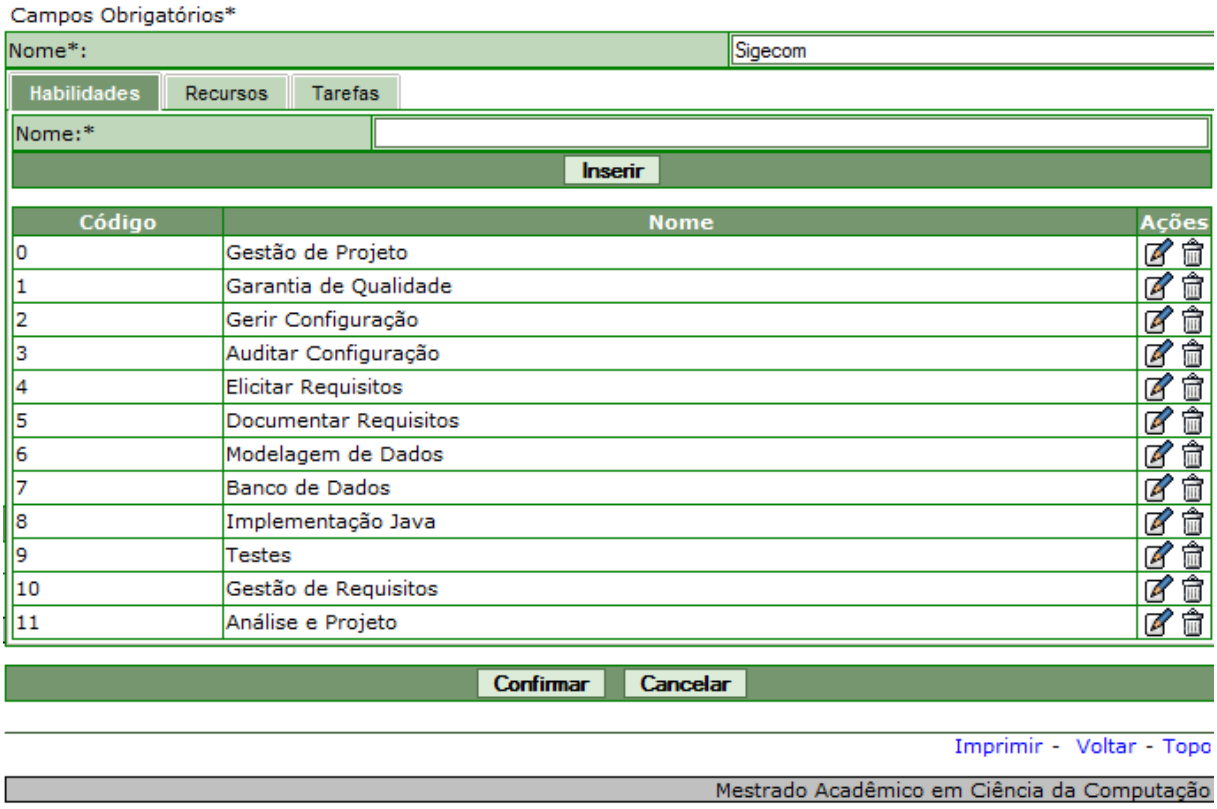


Figura 33: Cadastro de Habilidades

Uma observação importante é a relação entre datas e os dias sequenciais de planejamento. A abordagem proposta considera os dias de planejamento como números inteiros e crescentes de uma unidade, todos representando um dia útil de um cronograma real. Dessa forma, seja qual for a data em que o projeto inicie, consideramos essa data como o Dia 0, conforme ilustrado na Figura 3.

No CPG, apesar de não está visível nas figuras, o empregado Alex está disponível nessa iteração do Sigecom apenas uma hora normal por dia do dia 12 ao dia 23 de agosto e o empregado Sherman está disponível em tempo integral somente a partir do dia 14 de setembro. Devemos registrar os períodos de indisponibilidade desses dois recursos no cadastro de Recursos. Para isso, devemos calcular os dias úteis de indisponibilidade, uma vez que a modelagem considera apenas os dias úteis. Lembrando que, como o gerente considerou a carga horária de todos os funcionários de 7 horas diárias, um dia inteiro de indisponibilidade seria de 9 horas (7

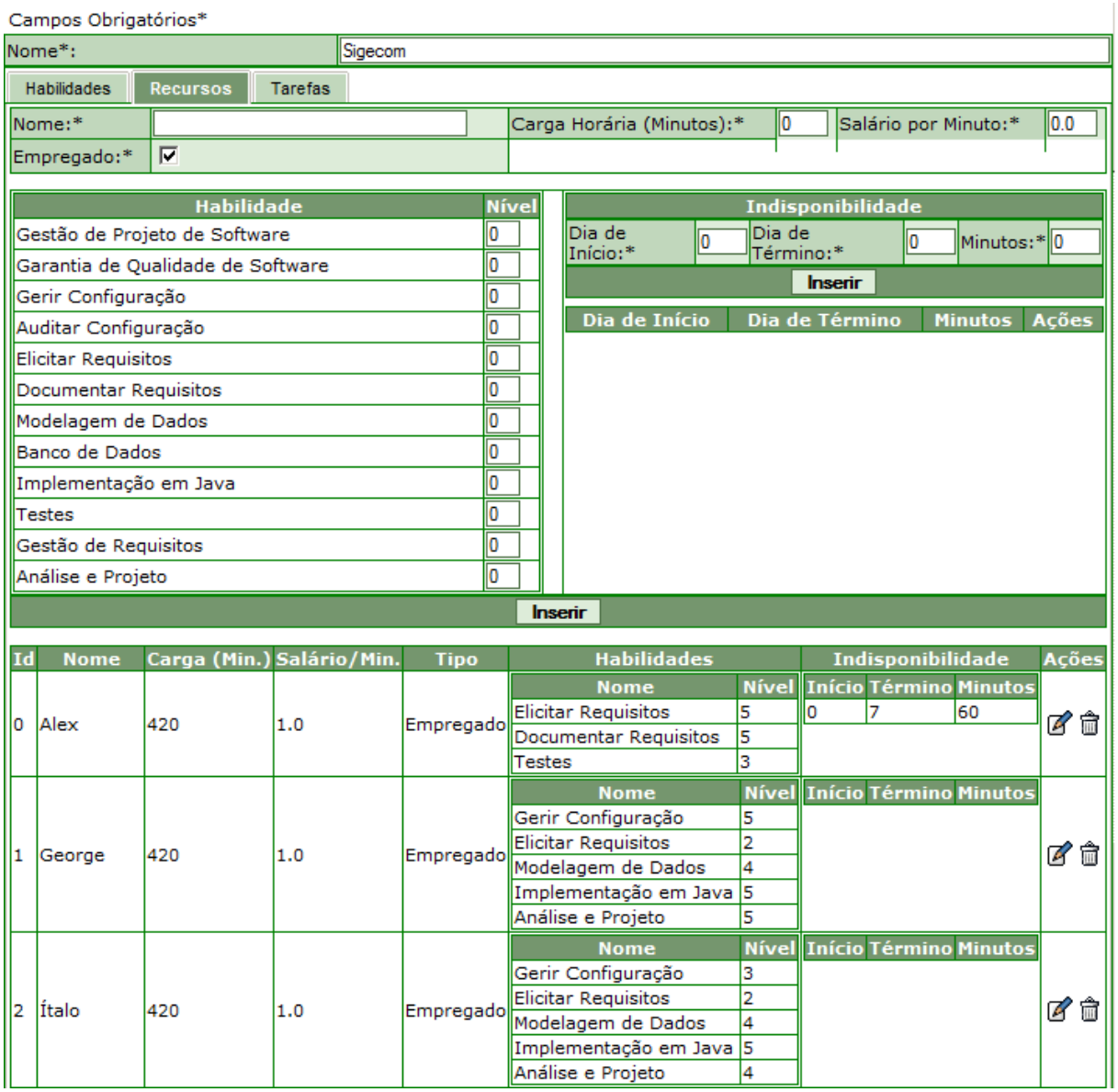


Figura 34: Cadastro de Recursos - Parte 1/2

horas normais + 2 de extra). Efetuando os cálculos temos que o empregado Alex está disponível 8 dias úteis, dos dias 12/08/2010 ao 23/09/2010, onde 12/08/2010 é o início do sistema. Logo, Alex possui um período de indisponibilidade de 6 horas por dia (permitindo ele trabalhar 1 hora normal mais 2 horas extra) do Dia 0 ao Dia 7. Sherman está indisponível desde o primeiro dia até o dia 23/09/2010 o que dá 22 dias úteis. Assim, seu período de indisponibilidade é de 9 horas por dia (indisponível totalmente) do Dia 0 ao Dia 21.

Outra informação que não está visível nas Figuras 30 e 31 é o salário individual de cada funcionário. Pela dificuldade de obter tal informação e por saber que a maioria dos funcionários

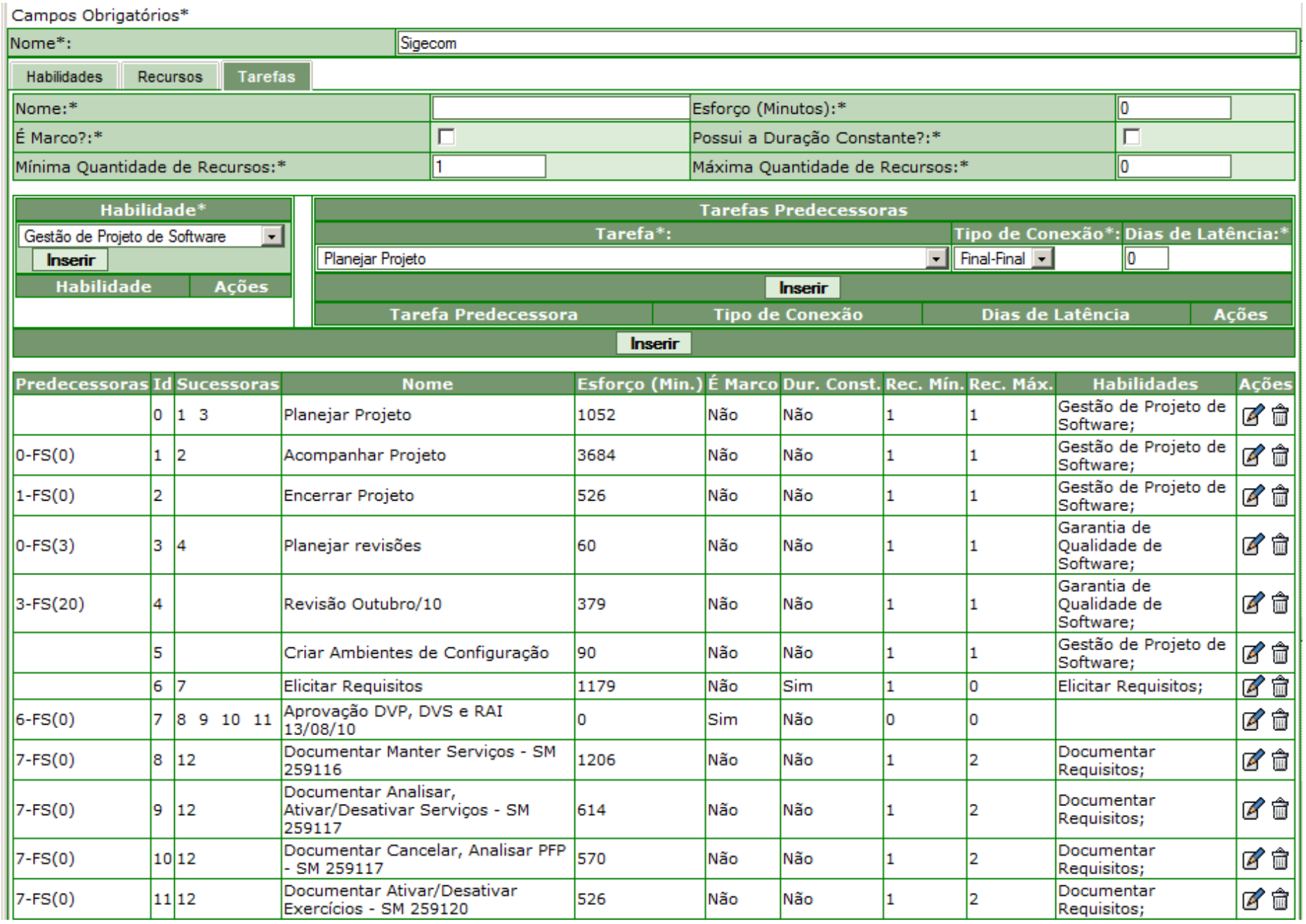
Id	Nome	Carga (Min.)	Salário/Min.	Tipo	Habilidades		Indisponibilidade			Ações
					Nome	Nível	Início	Término	Minutos	
3	Jairo	420	1.0	Empregado	Auditar Configuração	5				
4	Jenner	420	1.0	Empregado	Gerir Configuração	1				
					Elicitar Requisitos	2				
					Documentar Requisitos	3				
					Modelagem de Dados	3				
					Implementação em Java	5				
					Análise e Projeto	3				
5	Leninha	420	1.0	Empregado	Testes	5				
6	Lima	420	1.0	Empregado	Testes	5				
7	Maiza	420	1.0	Empregado	Gestão de Projeto de Software	5				
					Elicitar Requisitos	4				
					Documentar Requisitos	4				
					Testes	3				
					Gestão de Requisitos	4				
8	Rejane	420	1.0	Empregado	Gestão de Projeto de Software	4				
					Elicitar Requisitos	5				
					Documentar Requisitos	5				
					Testes	4				
					Gestão de Requisitos	5				
9	Selma	420	1.0	Empregado	Garantia de Qualidade de Software	5				
10	Sherman	420	1.0	Empregado	Elicitar Requisitos	5	0	13	540	
					Documentar Requisitos	5				
					Testes	3				
11	Washington	420	1.0	Empregado	Gerir Configuração	4				
					Elicitar Requisitos	2				
					Modelagem de Dados	5				
					Banco de Dados	5				

Confirmar Cancelar

Figura 35: Cadastro de Recursos - Parte 2/2

possui salários similares, atribuímos o mesmo valor-hora a todos os envolvidos.

Conforme informado no Capítulo 4, o gerente tem a opção de atribuir os limites mínimos e máximos do tamanho da equipe de cada tarefa. Isso é feito no Cadastro das Tarefas (Figuras 36 a 39), através dos campos *Mínima* e *Máxima Quantidade de Recursos*. Para os nossos testes, escolhemos esses limites com base na particularidade de cada atividade. Algumas atividades são muito curtas e pontuais, não fazendo sentido alocar para mais de uma pessoa. Outras são divisíveis apenas para um número pequeno de pessoas. Outras, como em programação em par, exigem o trabalho de um número mínimo de pessoas. Contudo, existem aquelas



Predecessoras	Id	Sucessoras	Nome	Esforço (Min.)	É Marco	Dur. Const.	Rec. Mín.	Rec. Máx.	Habilidades	Ações
	0	1 3	Planejar Projeto	1052	Não	Não	1	1	Gestão de Projeto de Software;	
0-FS(0)	1	2	Acompanhar Projeto	3684	Não	Não	1	1	Gestão de Projeto de Software;	
1-FS(0)	2		Encerrar Projeto	526	Não	Não	1	1	Gestão de Projeto de Software;	
0-FS(3)	3	4	Planejar revisões	60	Não	Não	1	1	Garantia de Qualidade de Software;	
3-FS(20)	4		Revisão Outubro/10	379	Não	Não	1	1	Garantia de Qualidade de Software;	
	5		Criar Ambientes de Configuração	90	Não	Não	1	1	Gestão de Projeto de Software;	
	6	7	Elicitar Requisitos	1179	Não	Sim	1	0	Elicitar Requisitos;	
6-FS(0)	7	8 9 10 11	Aprovação DVP, DVS e RAI 13/08/10	0	Sim	Não	0	0		
7-FS(0)	8	12	Documentar Manter Serviços - SM 259116	1206	Não	Não	1	2	Documentar Requisitos;	
7-FS(0)	9	12	Documentar Analisar, Ativar/Desativar Serviços - SM 259117	614	Não	Não	1	2	Documentar Requisitos;	
7-FS(0)	10	12	Documentar Cancelar, Analisar PFP - SM 259117	570	Não	Não	1	2	Documentar Requisitos;	
7-FS(0)	11	12	Documentar Ativar/Desativar Exercícios - SM 259120	526	Não	Não	1	2	Documentar Requisitos;	

Figura 36: Cadastro de Tarefas - Parte 1/4

Predecessoras	Id	Sucessoras	Nome	Esforço (Min.)	É Marco	Dur. Const.	Rec. Mín.	Rec. Máx.	Habilidades	Ações
7-FS(0)	11	12	Documentar Ativar/Desativar Exercícios - SM 259120	526	Não	Não	1	2	Documentar Requisitos;	
8-FS(0) 9-FS(0) 10-FS(0) 11-FS(0)	12	13	Verificar Requisitos / Revisão Técnica	1169	Não	Não	1	0	Documentar Requisitos;	
12-FS(0)	13	14	Ajustes de requisitos após revisão	1132	Não	Não	1	0	Documentar Requisitos;	
13-FS(0)	14	15	Marco Entrega Validação Requisitos pelo Cliente	0	Sim	Não	0	0		
14-FS(1)	15	16	Ajustes de requisitos após validação pelo Cliente	943	Não	Não	1	0	Documentar Requisitos;	
15-FS(0)	16	17	Revisão ajustes requisitos após validação Cliente	472	Não	Não	1	0	Documentar Requisitos;	
16-FS(0)	17	18 19 20	Marco de Entrega de Requisitos por Siscor	0	Sim	Não	0	0		
17-FS(1)	18	21 22 25 36 70	Termo de Aceite Requisitos	0	Sim	Não	0	0		
17-FS(3)	19		Atualizar Situação dos Requisitos	236	Não	Não	1	1	Gestão de Requisitos;	
17-FS(5)	20		Coletar Métricas de Requisitos	236	Não	Não	1	1	Gestão de Requisitos;	
18-FS(0)	21	23	Especificar Modelo de Dados	395	Não	Não	1	2	Modelagem de Dados;	
18-FS(0)	22	23	Especificar Funcionalidades	461	Não	Não	1	2	Análise e Projeto;	
21-FS(0) 22-FS(0)	23	24	Revisar Análise e Projeto / Revisão Técnica	131	Não	Não	1	1	Análise e Projeto; Modelagem de Dados;	
23-FS(0)	24	26 27 38	Ajustes A&P após revisão técnica	66	Não	Não	1	1	Análise e Projeto;	
18-FS(0)	25	26 27	Suporte ao desenvolvimento - AeP	263	Não	Não	1	2	Banco de Dados;	
24-FS(0) 25-FS(0)	26	28 29 30 31 32 33	Mapeamento O/R e Pojos	175	Não	Não	1	2	Modelagem de Dados;	
24-FS(0) 25-FS(0)	27	28 29 30 31 32 33	Scripts Banco	175	Não	Não	1	2	Banco de Dados;	
26-FS(0) 27-FS(0)	28	34 55 64	Implementar Manter Serviço	2423	Não	Não	1	0	Implementação em Java;	
26-FS(0) 27-FS(0)	29	34 55 64	Implementar Analisar Serviço	440	Não	Não	1	0	Implementação em Java;	
26-FS(0) 27-FS(0)	30	34 55 64	Implementar Ativar/Desativar Serviço	661	Não	Não	1	0	Implementação em Java;	
26-FS(0) 27-FS(0)	31	34 55 64	Implementar Cancelar PFP	440	Não	Não	1	0	Implementação em Java;	
26-FS(0) 27-FS(0)	32	34 55 64	Implementar Analisar PFP	881	Não	Não	1	0	Implementação em Java;	
26-FS(0) 27-FS(0)	33	34 55 64	Implementar Ativar/Desativar Exercícios Serviço	1058	Não	Não	1	0	Implementação em Java;	

Figura 37: Cadastro de Tarefas - Parte 2/4

Predecessoras	Id	Sucessoras	Nome	Esforço (Min.)	É Marco	Dur. Const.	Rec. Mín.	Rec. Máx.	Habilidades	Ações
28-FS(0) 29-FS(0) 30-FS(0) 31-FS(0) 32-FS(0) 33-FS(0)	34	35	Revisar Implementação / Revisão Técnica	877	Não	Não	1	0	Implementação em Java;	
34-FS(0)	35	40	Ajustes implementação após Revisão Técnica	877	Não	Não	1	0	Implementação em Java;	
18-FS(0)	36	37	Gerar baseline requisitos SIGECOM-093800-01_00-REQ-00	30	Não	Não	1	1	Gestão de Requisitos;	
36-FS(5)	37		Auditar baseline requisitos SIGECOM-093800-01_00-REQ-00	240	Não	Não	1	0	Auditar Configuração;	
24-FS(0)	38	39	Gerar baseline A&P SIGECOM-093800-01_00-AeP-00	30	Não	Não	1	1	Gerir Configuração;	
38-FS(5)	39		Auditar baseline A&P SIGECOM-093800-01_00-AeP-00	240	Não	Não	1	0	Auditar Configuração;	
35-FS(0)	40	41	Build 04.01.00 - Testes Ciclo #4.1.0	120	Não	Não	1	1	Gerir Configuração;	
40-FS(0)	41	42	Testes iniciais pela equipe req - Ciclo #4.1.0	614	Não	Não	1	0	Testes;	
41-FF(1)	42	43	Correções verificadas pela eq Req - Ciclo #4.1.0	1579	Não	Não	1	0	Implementação em Java;	
42-FS(0)	43	44	Build 04.01.01 - Testes Ciclo #4.1.1	60	Não	Não	1	1	Gerir Configuração;	
43-FS(0)	44	45	Testes - Ciclo #4.1.1	1474	Não	Não	1	0	Testes;	
44-FF(1)	45	46	Correção dos Testes - Ciclo #4.1.1	1930	Não	Não	1	0	Implementação em Java;	
45-FS(0)	46	47	Build 04.01.02 - Testes Ciclo #4.1.2	60	Não	Não	1	1	Gerir Configuração;	
46-FS(0)	47	48 56	Testes - Ciclo #4.1.2	614	Não	Não	1	0	Testes;	
47-FF(1)	48	49	Correção dos Testes - Ciclo #4.1.2	1403	Não	Não	1	0	Implementação em Java;	
48-FS(0)	49	50	Build e publicação homol 04.01.03 - Testes Ciclo #4.1.3	60	Não	Não	1	1	Gerir Configuração;	
49-FS(0)	50	51 57	Testes em Homologação Ciclo #4.1.3	614	Não	Não	1	0	Testes;	
50-FF(1)	51	52	Correção dos Testes em Homol - Ciclo #4.1.3	351	Não	Não	1	0	Implementação em Java;	
51-FS(0)	52	53	Build e publicação homol 04.01.04 - Testes Ciclo #4.1.4	60	Não	Não	1	1	Gerir Configuração;	
52-FS(0)	53	54 67	ReTestes em Homologação - Ciclo #4.1.4	614	Não	Não	1	0	Testes;	
53-FS(3)	54		Gerar e promover baseline Implem SIGECOM-093800-04_01_05 para Compilada e Testada	60	Não	Não	1	1	Gerir Configuração;	

Figura 38: Cadastro de Tarefas - Parte 3/4

Predecessoras	Id	Sucessoras	Nome	Esforço (Min.)	É Marco	Dur. Const.	Rec. Mín.	Rec. Máx.	Habilidades	Ações
28-FS(0) 29-FS(0) 30-FS(0) 31-FS(0) 32-FS(0) 33-FS(0)	55		Planejar Homologação	420	Não	Não	1	1	Gestão de Projeto de Software;	
47-FS(3)	56		Executar script/publicar #4.1.1	536	Não	Não	1	0	Gerir Configuração;	
50-FS(3)	57	58	Publicar após teste equipe #4.1.1	120	Não	Não	1	1	Gerir Configuração;	
57-FS(0)	58	59	Marco de entrega para Homologação # 4.1.1	0	Sim	Não	0	0		
58-SS(3)	59	60	Fim 1a. Homologação pelo Cliente # 4.1.1	0	Sim	Não	0	0		
59-FS(0)	60	61 65	Correção pós Homologação #4.1.4	702	Não	Não	1	0	Implementação em Java;	
60-FF(3)	61	62	Executar script/publicar #.4.1.2	240	Não	Não	1	0	Gerir Configuração;	
61-FS(0)	62	63	Marco de entrega para 2a. Homologação # 4.1.2	0	Sim	Não	0	0		
62-SS(2)	63	68 72	Termo de Aceite Cliente	0	Sim	Não	0	0		
28-FS(0) 29-FS(0) 30-FS(0) 31-FS(0) 32-FS(0) 33-FS(0)	64	72	Planejar Implantação	277	Não	Não	1	0	Gestão de Projeto de Software;	
60-FS(0)	65	66	Build e publicação homol 04.01.05 - Testes Ciclo #4.1.5	60	Não	Não	1	0	Gerir Configuração;	
65-FS(0)	66	71	Testes pós Homologação - Ciclo #4.1.5	491	Não	Não	1	0	Testes;	
53-FS(0)	67		Marco de Entrega de Código	0	Sim	Não	0	0		
63-FS(3)	68	69	Promover baseline SIGECOM-093800-04_01_05 para Homologada	30	Não	Não	1	0	Gerir Configuração;	
68-FS(5)	69		Auditar baseline Implem SIGECOM-093800-04_01_05	360	Não	Não	1	0	Auditar Configuração;	
18-FS(0)	70		Plano e Especificação de Testes	1412	Não	Não	1	0	Testes;	
66-FS(0)	71		Avaliar Resultados dos testes	307	Não	Não	1	0	Testes;	
64-FS(0) 63-FS(3)	72	73	Executar script/publicar Implantação	600	Não	Não	1	0	Gerir Configuração;	
72-FS(0)	73	74 75	Marco de Implantação	0	Sim	Não	0	0		
73-FS(3)	74		Validar Implantação - QA	0	Sim	Não	0	0		
73-FS(3)	75		Promover baseline SIGECOM-093800-04_01_05 para Implantada	30	Não	Não	1	1	Gerir Configuração;	

Confirmar Cancelar

Figura 39: Cadastro de Tarefas - Parte 4/4

que podem ser divididas para várias pessoas. Obviamente que, até certa quantidade de pessoas, a formação de equipes numerosas é indesejável pelo conceito do *overhead* de comunicação.

Uma vez definida a instância de entrada, podemos dar início à execução dos algoritmos. Criamos um indivíduo baseado no cromossomo escolhido pelo gerente, isto é, um indivíduo que contém o sequenciamento da Lista de Atividades definido pelo campo *Ordem de Execução* e pela Matriz de Alocação definida pelo campo *Recursos* dispostos no CPG. Chamaremos esse cromossomo de *CGerente*, o indivíduo de *IGerente* e a solução, ou seja, o cronograma elaborado a partir de seu cromossomo, de *SGerente*. A ideia é comparar as soluções *SGerente* com as soluções geradas pela modelagem proposta. Como a construção do cronograma a partir de um cromossomo é determinística, obtivemos uma única solução *SGerente*.

Após a construção de *SGerente*, é a vez de executamos o *Algoritmo Genético* e o *NSGA-II* livre da configuração do gerente. Em toda execução, mantivemos os mesmo parâmetros de entrada, dispostos na Figura 40. Os parâmetros são auto-explicativos e já foram previamente comentados, ficando apenas algumas observações. O GA implementa todos quatro os critérios de parada, porém o NSGA-II não considera a *Quantidade Máxima de Gerações sem Melhoria* como critério de parada, dado o princípio que pode não haver uma única solução melhor que todas as outras. Esses critérios podem ser combinados entre si e aquele que ocorrer primeiro será o que determinará a parada do algoritmo. Para ignorar um critério basta atribuir seu valor igual a zero. Para os testes, todos os algoritmos tiveram seu tempo de execução limitado em 30 minutos. Os valores dos parâmetros que podem alterar a qualidade dos resultados, tais como o tamanho da população e as probabilidades de cruzamento, mutação e heurísticas, foram escolhidos baseados na qualidade dos resultados para a instância trabalhada.

As Figuras 41, 42 e 43 ilustram comparativos entre os resultados dos algoritmos executados. Os objetivos foram combinados dois a dois para formar gráficos bidimensionais, permitindo uma melhor visualização dos resultados. Para que as melhores soluções entre pares de objetivos do NSGA-II sejam exibidas, novamente ordenamos a Frente de Pareto pelo critério de não-dominância considerando apenas dois objetivos por vez. Por exemplo, seja uma Frente de Pareto com um cronograma C_1 com duração de 50 dias, custo de \$800 e qualidade

Nome do Projeto	Opção	Ações
Sigecom	Gerar Novo Projeto Aleatório	✓
	Calcular Projeto	✓

Calcular Projeto			
Tamanho da População	100		
Probabilidade de Cruzamento da Alocação (%)	0.6	Probabilidade de Cruzamento do Escalonamento (%)	0.6
Probabilidade de Mutação da Alocação (%)	0.01	Probabilidade de Mutação do Escalonamento (%)	0.01
Probabilidade de Aplicar Heurísticas de Melhoria	0.8		
Probabilidade de Não Alocar um Recurso	0.9	Aplicar Heurística de Redução de Horas Extra	0.01
Aplicar Heurística de Aumento de Dedicação com Horas Extras	0.01	Aplicar Heurística de Aumento de Dedicação sem Horas Extra	0.9
Aplicar Heurística de Troca de Dedicação com Horas Extra	0.01	Aplicar Heurística de Troca de Dedicação sem Horas Extra	0.01
Quantidade Máxima de Gerações	0	Quantidade Máxima de Gerações sem Melhoria	0
Tempo de Execução Máximo (Minutos)	30		
Quantidade Máxima de Minutos Extras por Dia	120	Taxa Salarial Adicional do Minuto Extra (%)	0.5
Taxa de Comunicação Entre Dois Recursos	0.05	Período de Funcionamento da Empresa por Dia em Minutos	600
Salvar Histórico	☐	Calcular Média de Aptidão	☐
Executar GA	☑	Executar NSGA-II	☑
Executar Algoritmo Aleatório - Mono	☐	Executar Algoritmo Aleatório - Multi	☐
Executar Algoritmo Aleatório com Heurísticas - Mono	☐	Executar Algoritmo Aleatório com Heurísticas - Multi	☐
Mestrado Acadêmico em Ciência da Computação			

Figura 40: Parâmetros de entrada de execução do projeto Sigecom

de 2.000, um cronograma C_2 com duração de 70 dias, custo de $400 e qualidade de 1.800,00 e um cronograma C_3 de duração de 65 dias, custo de $500 e qualidade de 900,00. Para exibir o gráfico de *Custo x Tempo*, aplicamos a ordenação nessa Frente considerando apenas os objetivo Custo e Tempo e, como resultado, obtemos uma Frente com apenas os cronogramas C_1 (menor tempo) e C_2 (menor custo). Como a Qualidade das Equipes nessa comparação é desprezada, descartamos as soluções dominadas pelos dois objetivos, no caso C_{C3}. De forma análoga, para uma comparação entre *Tempo x Qualidade*, apenas os cronogramas C_1 (menor tempo) e C_3 (menor qualidade) serão exibidos. Na comparação *Custo x Qualidade*, serão impressos apenas os cronogramas C_2 (menor custo) e C_3 (menor qualidade).

Observe que o tempo está em função de dias. Logo, para um empresa que funciona 10 horas por dia, o valor de 64,56 representa 64 dias, 5 horas e 36 minutos. Quanto aos valores de Qualidade de Formação de Equipes, alteramos o sinal para que ele também seja tratado como um problema de minimização.

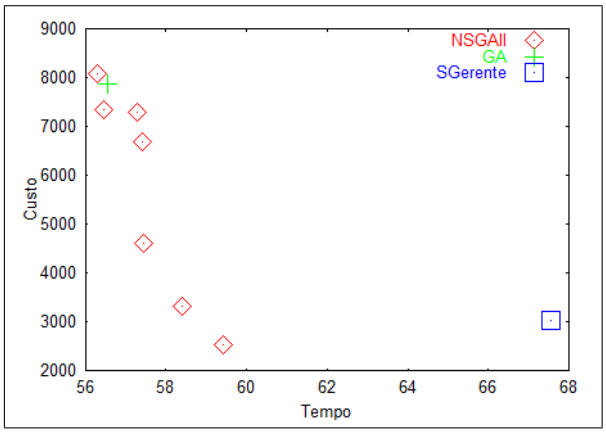


Figura 41: Projeto Sigecom - Custo x Tempo

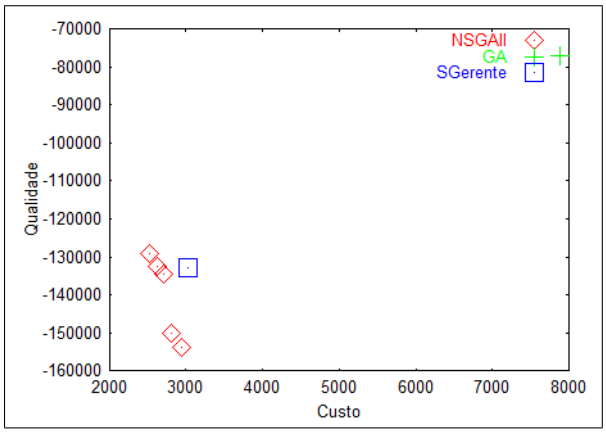


Figura 42: Projeto Sigecom - Qualidade x Custo

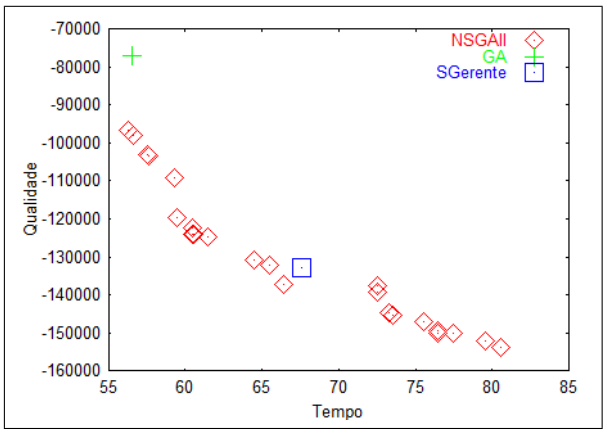


Figura 43: Projeto Sigecom - Qualidade x Tempo

Salientamos que o Algoritmo Genético, versão mono-objetivo da modelagem, se propõe apenas à minimização do custo total, que é o somatório do custo salarial dos empregados e do custo em horas extras e de consultores. Devido ao custo salarial dos empregados estar em função da duração do projeto, o GA consegue construir cronogramas com bons tempos de conclusão. Entretanto, dependendo da instância do problema, o custo salarial pago a cada empregado por dia é muitas vezes superior ao valor pago em horas extras e aos consultores. Isso aumenta as chances do GA encontrar soluções com melhores durações em detrimento ao custo de horas extra e consultores. No NSGA-II isso não ocorre, pois os objetivos são desacoplados e trabalhados de forma independente.

Adicionamos a solução do gerente (*SGerente*) nos gráficos apenas compará-lo com a

qualidade das soluções dos algoritmos. Podemos perceber que o cronograma construído a partir da configuração do gerente (*SGerente*) buscou alcançar valores intermediários em todos os aspectos, priorizando na formação das equipes e na redução do custo relativo às horas extras. Lembrando que não existem custos relativos a consultores no Sigecom. A solução encontrada pelo GA possui um dos menores tempos, porém com um dos custos monetário mais elevado. Como o GA não se propõe a formar equipes de qualidade, seu desempenho nesse objetivo foi o pior dos algoritmos. O NSGA-II, pela sua própria natureza, gerou resultados bem diversificados, encontrando sempre os melhores objetivos. Mesmo quando comparamos os três objetivos simultaneamente, pelo menos uma solução do NSGA-II é melhor que o resultado do GA e pelo menos uma é melhor que *SGerente*. Em outras palavras, pelo menos uma solução do NSGA-II domina a solução do GA e pelo menos uma solução do NSGA-II domina *SGerente*. Se a solução do GA e *SGerente* fossem incorporados no conjunto de soluções do NSGA-II, eles seriam descartados pelo conceito de dominância. A Tabela 7 mostra os valores dos objetivos do resultado do GA e *SGerente* e uma solução dominante do NSGA-II para cada um deles.

Tabela 7: Dominância do NSGA-II em relação aos outros resultados

Solução	Tempo	Custo	Qualidade
GA	56,56	7.876,50	77.066,00
NSGA-II	56,47	7.357,50	82.340,00
SGerente	67,55	3.030,00	132.874,00
NSGA-II	66,38	3.094,50	137.133,00

Conforme informado anteriormente, deve existir a figura de um especialista para decidir qual solução dentro do conjunto de soluções do NSGA-II é melhor para o projeto. Suponha que o gerente escolha a solução com *makespan* de 57,47, custo de $4.600,00 e Qualidade das Equipes de 87.688,00, o cronograma completo pode ser visualizado nas Figuras 44 a 49.

5.2 Análise de Sensibilidade

Esta seção destina-se a realizar uma análise de sensibilidade da modelagem proposta. Embora o projeto Sigecom trabalhado na seção anterior seja um projeto real, ele não possui todos os aspectos envolvidos na modelagem, tais como recursos humanos do tipo Consultor e o tipo de ligação Início-Final. Para isso, aplicamos a análise de sensibilidade em um projeto

completo criado por um gerador de instâncias aleatórias. Os parâmetros do gerador é exibido na Figura 52.

Executamos o gerador e foi criado um projeto com 10 recursos, sendo 7 Empregados e 3 Consultores e 50 tarefas, sendo 28 Concretas, 14 Marcos e 8 de Duração Fixa. Todos os recursos possuem de 1 a 5 habilidades e cada habilidade pode variar de 1 a 5 níveis. O projeto possui todos os tipos de ligação (Início-Início, Início-Final, Final-Início e Final-Final) e cada ligação pode ter ou não tempo de latência associado.

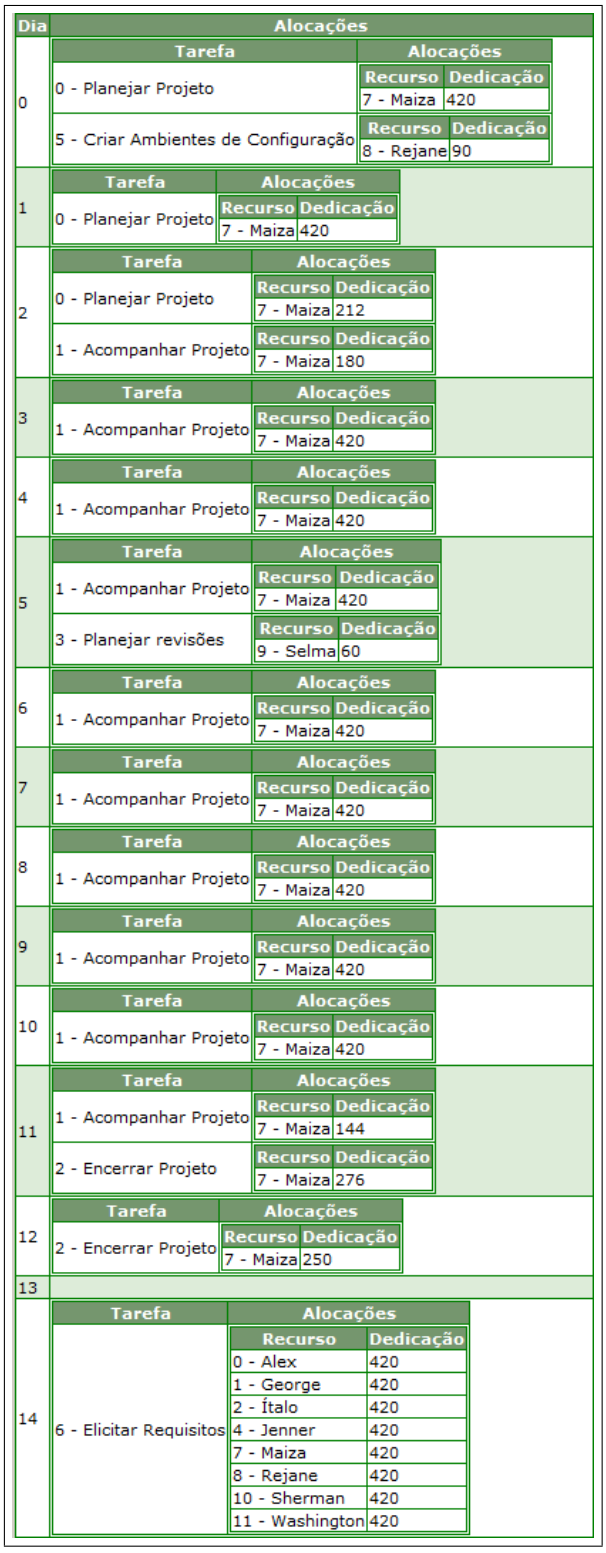
Os testes foram realizados executando o GA, o NSGA-II e um Algoritmo Aleatório Multiobjetivo, todos com os mesmo parâmetros da seção anterior (Figura 40). O Algoritmo Aleatório contém as mesmas funções objetivo da versão multiobjetiva da modelagem. Nesse algoritmo, a cada iteração, N indivíduos são gerados aleatoriamente, avaliados e adicionados à população, onde N é o tamanho da população. A população inteira é então submetida ao mesmo processo de ordenação do NSGA-II, onde os indivíduos menos aptos serão eliminados. Ele foi introduzido apenas para o GA e NSGA-II passar pelo "teste de sanidade". O comparativo dos resultados pode ser visto nas Figuras 53 a 55.

Nossa análise de sensibilidade consiste em analisar a relação entre a equipe completa e a equipe sem consultores e sem a permissão de realização de horas extra por parte dos empregados. Para facilitar nosso comparativo, resolvemos considerar apenas o NSGA-II e o efeito das modificações em cada um de seus objetivos. As Figuras 56 a 58 exibem os resultados em todos esses aspectos. Podemos perceber na relação Custo x Tempo (Figura 58) que houve uma sutil redução do custo quando os empregados não realizam horas extra e uma redução significativa quando a empresa não contrata consultores para o projeto. O tempo nessa relação praticamente não sofre alterações, apenas uma leve tendência de seu aumento quando não é permitida a realização de horas extra. As Figuras 57 e 58 mostram a possibilidade de aumento e diminuição da qualidade na formação das equipes quando os consultores são desconsiderados no projeto. Isso pode ocorrer devido o novo quadro de recursos disponíveis, em que consultores menos ou mais habilidosos foram eliminados do projeto.

Podemos concluir que a eliminação das horas extras pouco influenciou nos valores dos

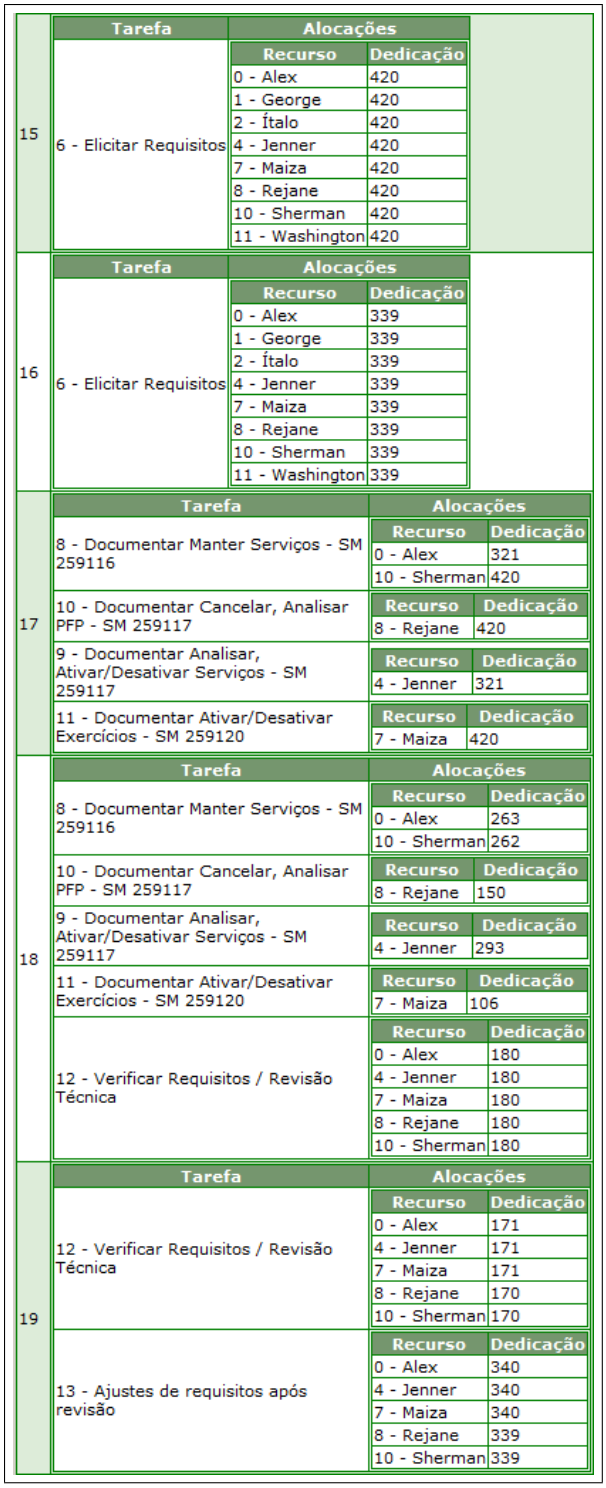
objetivos. No entanto, a eliminação dos consultores resultou em grandes reduções do custo do projeto, sem aumentos significativas no tempo. Isso mostra que a redução de 30% da quantidade de recursos humanos é suficiente para o projeto, não interferindo no tempo de conclusão do projeto. A qualidade na formação das equipes não teve relação direta com remoção das horas extras e dos consultores. A qualidade na formação das equipes pode variar dependendo dos níveis de habilidades dos recursos eliminados.

Uma observação importante é que outras análises podem ser realizadas pelo gerente através da ferramenta implementada, tais como remover/inserir recursos/tarefas do projeto, aumentar/diminuir a carga horária ou disponibilidade dos recursos etc. Dessa forma, o gerente poderá tomar decisões baseadas no comportamento das análises realizadas.



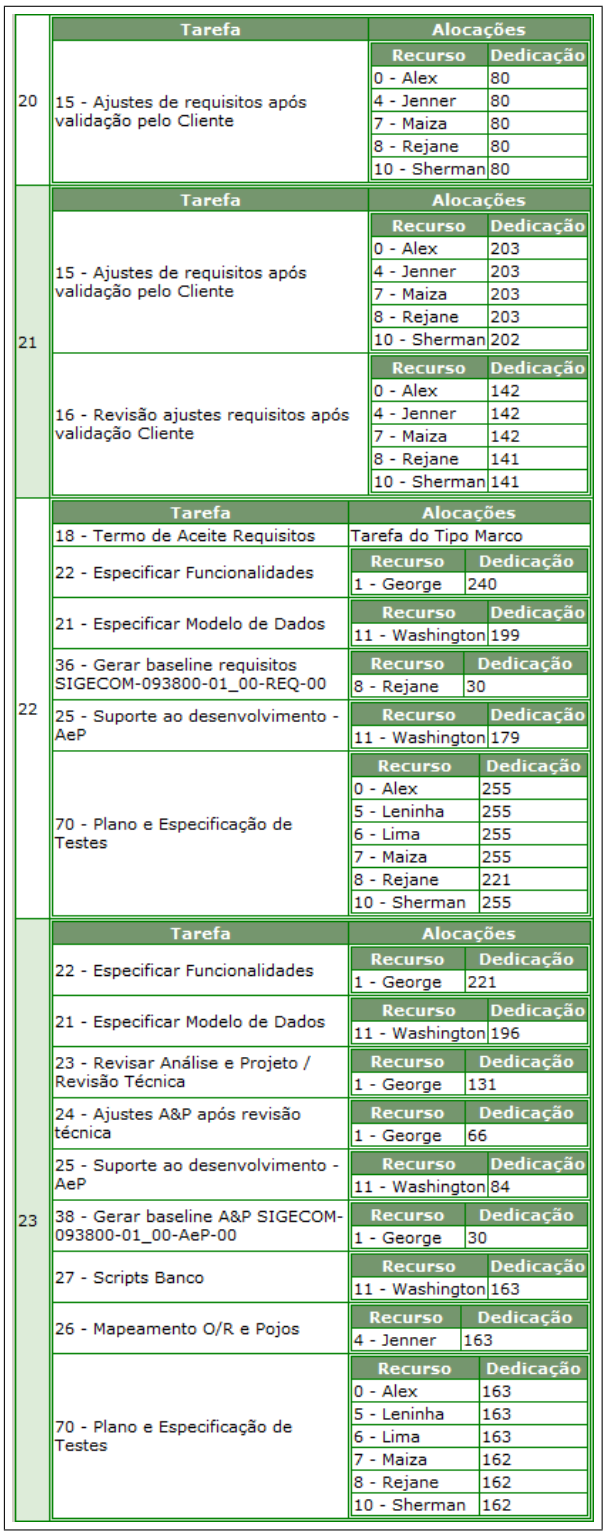
Dia	Alocações		
0	Tarefa	Alocações	
	0 - Planejar Projeto	Recurso	Dedicação
		7 - Maiza	420
	5 - Criar Ambientes de Configuração	Recurso	Dedicação
		8 - Rejane	90
1	Tarefa	Alocações	
	0 - Planejar Projeto	Recurso	Dedicação
		7 - Maiza	420
2	Tarefa	Alocações	
	0 - Planejar Projeto	Recurso	Dedicação
		7 - Maiza	212
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	180
3	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
4	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
5	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
	3 - Planejar revisões	Recurso	Dedicação
		9 - Selma	60
6	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
7	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
8	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
9	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
10	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	420
11	Tarefa	Alocações	
	1 - Acompanhar Projeto	Recurso	Dedicação
		7 - Maiza	144
	2 - Encerrar Projeto	Recurso	Dedicação
		7 - Maiza	276
12	Tarefa	Alocações	
	2 - Encerrar Projeto	Recurso	Dedicação
		7 - Maiza	250
13			
14	Tarefa	Alocações	
	6 - Elicitar Requisitos	Recurso	Dedicação
		0 - Alex	420
		1 - George	420
		2 - Ítalo	420
		4 - Jenner	420
		7 - Maiza	420
		8 - Rejane	420
		10 - Sherman	420
		11 - Washington	420

Figura 44: Exemplo de um Cronograma Elaborado - Sigecom - Parte 1/8



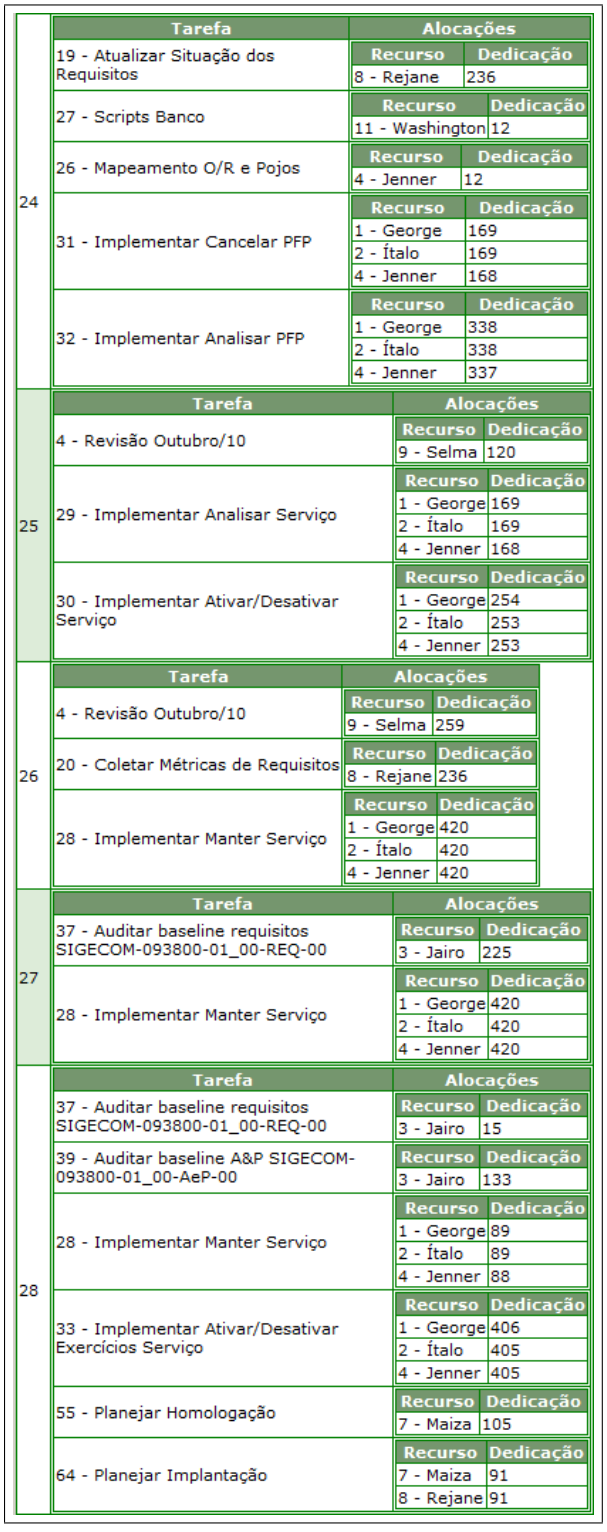
15	Tarefa	Alocações	
	6 - Elicitar Requisitos	Recurso	Dedicação
		0 - Alex	420
		1 - George	420
		2 - Ítalo	420
		4 - Jenner	420
		7 - Maiza	420
		8 - Rejane	420
		10 - Sherman	420
		11 - Washington	420
16	Tarefa	Alocações	
	6 - Elicitar Requisitos	Recurso	Dedicação
		0 - Alex	339
		1 - George	339
		2 - Ítalo	339
		4 - Jenner	339
		7 - Maiza	339
		8 - Rejane	339
		10 - Sherman	339
		11 - Washington	339
17	Tarefa	Alocações	
	8 - Documentar Manter Serviços - SM 259116	Recurso	Dedicação
		0 - Alex	321
		10 - Sherman	420
	10 - Documentar Cancelar, Analisar PFP - SM 259117	Recurso	Dedicação
		8 - Rejane	420
	9 - Documentar Analisar, Ativar/Desativar Serviços - SM 259117	Recurso	Dedicação
		4 - Jenner	321
	11 - Documentar Ativar/Desativar Exercícios - SM 259120	Recurso	Dedicação
		7 - Maiza	420
18	Tarefa	Alocações	
	8 - Documentar Manter Serviços - SM 259116	Recurso	Dedicação
		0 - Alex	263
		10 - Sherman	262
	10 - Documentar Cancelar, Analisar PFP - SM 259117	Recurso	Dedicação
		8 - Rejane	150
	9 - Documentar Analisar, Ativar/Desativar Serviços - SM 259117	Recurso	Dedicação
		4 - Jenner	293
	11 - Documentar Ativar/Desativar Exercícios - SM 259120	Recurso	Dedicação
		7 - Maiza	106
	12 - Verificar Requisitos / Revisão Técnica	Recurso	Dedicação
		0 - Alex	180
		4 - Jenner	180
		7 - Maiza	180
		8 - Rejane	180
		10 - Sherman	180
19	Tarefa	Alocações	
	12 - Verificar Requisitos / Revisão Técnica	Recurso	Dedicação
		0 - Alex	171
		4 - Jenner	171
		7 - Maiza	171
		8 - Rejane	170
		10 - Sherman	170
	13 - Ajustes de requisitos após revisão	Recurso	Dedicação
		0 - Alex	340
		4 - Jenner	340
		7 - Maiza	340
		8 - Rejane	339
		10 - Sherman	339

Figura 45: Exemplo de um Cronograma Elaborado - Parte 2/8



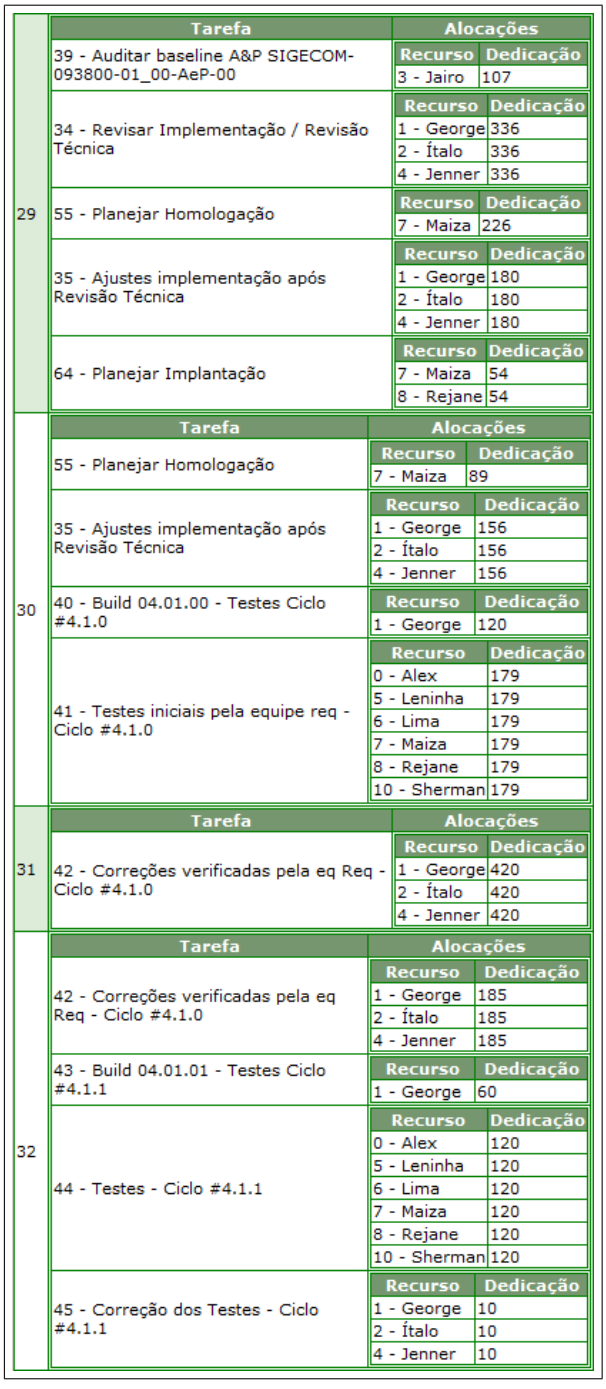
	Tarefa	Alocações	
20	15 - Ajustes de requisitos após validação pelo Cliente	Recurso	Dedicação
		0 - Alex	80
		4 - Jenner	80
		7 - Maiza	80
		8 - Rejane	80
		10 - Sherman	80
21	15 - Ajustes de requisitos após validação pelo Cliente	Recurso	Dedicação
		0 - Alex	203
		4 - Jenner	203
		7 - Maiza	203
		8 - Rejane	203
		10 - Sherman	202
	16 - Revisão ajustes requisitos após validação Cliente	Recurso	Dedicação
		0 - Alex	142
		4 - Jenner	142
		7 - Maiza	142
		8 - Rejane	141
		10 - Sherman	141
22	18 - Termo de Aceite Requisitos	Tarefa do Tipo Marco	
	22 - Especificar Funcionalidades	Recurso	Dedicação
		1 - George	240
	21 - Especificar Modelo de Dados	Recurso	Dedicação
		11 - Washington	199
	36 - Gerar baseline requisitos SIGECOM-093800-01_00-REQ-00	Recurso	Dedicação
		8 - Rejane	30
	25 - Suporte ao desenvolvimento - AeP	Recurso	Dedicação
		11 - Washington	179
	70 - Plano e Especificação de Testes	Recurso	Dedicação
		0 - Alex	255
		5 - Leninha	255
		6 - Lima	255
		7 - Maiza	255
		8 - Rejane	221
		10 - Sherman	255
23	22 - Especificar Funcionalidades	Recurso	Dedicação
		1 - George	221
	21 - Especificar Modelo de Dados	Recurso	Dedicação
		11 - Washington	196
	23 - Revisar Análise e Projeto / Revisão Técnica	Recurso	Dedicação
		1 - George	131
	24 - Ajustes A&P após revisão técnica	Recurso	Dedicação
		1 - George	66
	25 - Suporte ao desenvolvimento - AeP	Recurso	Dedicação
		11 - Washington	84
	38 - Gerar baseline A&P SIGECOM-093800-01_00-AeP-00	Recurso	Dedicação
		1 - George	30
	27 - Scripts Banco	Recurso	Dedicação
		11 - Washington	163
	26 - Mapeamento O/R e Pojos	Recurso	Dedicação
		4 - Jenner	163
	70 - Plano e Especificação de Testes	Recurso	Dedicação
		0 - Alex	163
		5 - Leninha	163
		6 - Lima	163
		7 - Maiza	162
		8 - Rejane	162
		10 - Sherman	162

Figura 46: Exemplo de um Cronograma Elaborado - Parte 3/8



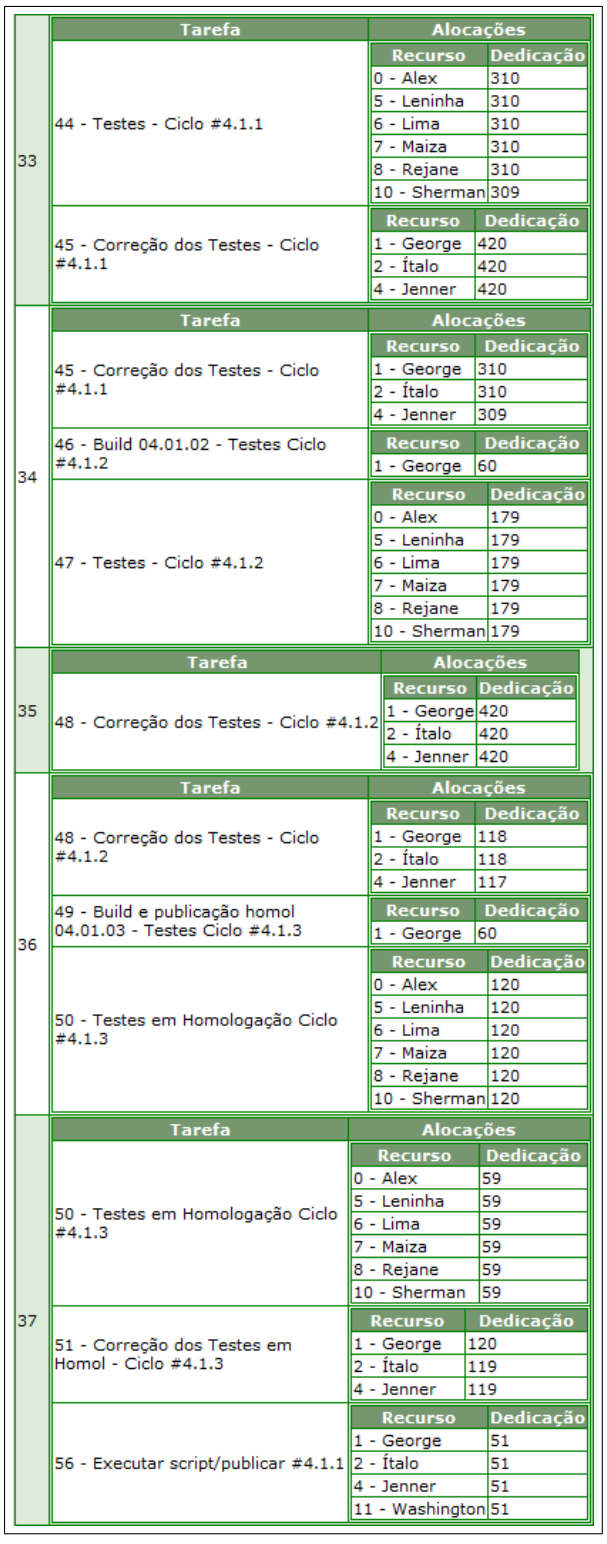
	Tarefa	Alocações	
24	19 - Atualizar Situação dos Requisitos	Recurso	Dedicação
		8 - Rejane	236
	27 - Scripts Banco	Recurso	Dedicação
		11 - Washington	12
	26 - Mapeamento O/R e Pojos	Recurso	Dedicação
		4 - Jenner	12
	31 - Implementar Cancelar PFP	Recurso	Dedicação
		1 - George	169
		2 - Ítalo	169
		4 - Jenner	168
	32 - Implementar Analisar PFP	Recurso	Dedicação
		1 - George	338
		2 - Ítalo	338
		4 - Jenner	337
25	4 - Revisão Outubro/10	Recurso	Dedicação
		9 - Selma	120
	29 - Implementar Analisar Serviço	Recurso	Dedicação
		1 - George	169
		2 - Ítalo	169
		4 - Jenner	168
	30 - Implementar Ativar/Desativar Serviço	Recurso	Dedicação
		1 - George	254
		2 - Ítalo	253
		4 - Jenner	253
26	4 - Revisão Outubro/10	Recurso	Dedicação
		9 - Selma	259
	20 - Coletar Métricas de Requisitos	Recurso	Dedicação
		8 - Rejane	236
	28 - Implementar Manter Serviço	Recurso	Dedicação
		1 - George	420
		2 - Ítalo	420
		4 - Jenner	420
27	37 - Auditar baseline requisitos SIGECOM-093800-01_00-REQ-00	Recurso	Dedicação
		3 - Jairo	225
	28 - Implementar Manter Serviço	Recurso	Dedicação
		1 - George	420
		2 - Ítalo	420
		4 - Jenner	420
28	37 - Auditar baseline requisitos SIGECOM-093800-01_00-REQ-00	Recurso	Dedicação
		3 - Jairo	15
	39 - Auditar baseline A&P SIGECOM-093800-01_00-AeP-00	Recurso	Dedicação
		3 - Jairo	133
	28 - Implementar Manter Serviço	Recurso	Dedicação
		1 - George	89
		2 - Ítalo	89
		4 - Jenner	88
	33 - Implementar Ativar/Desativar Exercícios Serviço	Recurso	Dedicação
		1 - George	406
		2 - Ítalo	405
		4 - Jenner	405
	55 - Planejar Homologação	Recurso	Dedicação
		7 - Maiza	105
	64 - Planejar Implantação	Recurso	Dedicação
		7 - Maiza	91
		8 - Rejane	91

Figura 47: Exemplo de um Cronograma Elaborado - Parte 4/8



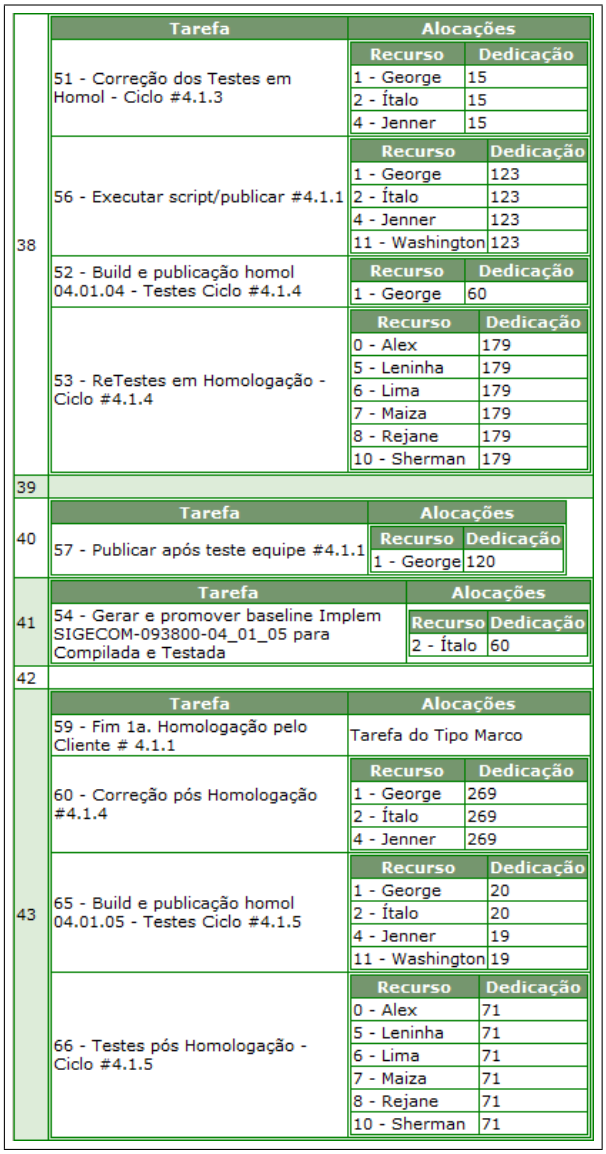
	Tarefa	Alocações	
29	39 - Auditar baseline A&P SIGECOM-093800-01_00-AeP-00	Recurso	Dedicação
		3 - Jairo	107
	34 - Revisar Implementação / Revisão Técnica	Recurso	Dedicação
		1 - George	336
		2 - Ítalo	336
		4 - Jenner	336
	55 - Planejar Homologação	Recurso	Dedicação
		7 - Maiza	226
	35 - Ajustes implementação após Revisão Técnica	Recurso	Dedicação
		1 - George	180
		2 - Ítalo	180
		4 - Jenner	180
	64 - Planejar Implantação	Recurso	Dedicação
		7 - Maiza	54
		8 - Rejane	54
30	Tarefa	Alocações	
	55 - Planejar Homologação	Recurso	Dedicação
		7 - Maiza	89
	35 - Ajustes implementação após Revisão Técnica	Recurso	Dedicação
		1 - George	156
		2 - Ítalo	156
		4 - Jenner	156
	40 - Build 04.01.00 - Testes Ciclo #4.1.0	Recurso	Dedicação
		1 - George	120
	41 - Testes iniciais pela equipe req - Ciclo #4.1.0	Recurso	Dedicação
		0 - Alex	179
		5 - Leninha	179
		6 - Lima	179
		7 - Maiza	179
		8 - Rejane	179
		10 - Sherman	179
31	Tarefa	Alocações	
	42 - Correções verificadas pela eq Req - Ciclo #4.1.0	Recurso	Dedicação
		1 - George	420
		2 - Ítalo	420
		4 - Jenner	420
32	Tarefa	Alocações	
	42 - Correções verificadas pela eq Req - Ciclo #4.1.0	Recurso	Dedicação
		1 - George	185
		2 - Ítalo	185
		4 - Jenner	185
	43 - Build 04.01.01 - Testes Ciclo #4.1.1	Recurso	Dedicação
		1 - George	60
	44 - Testes - Ciclo #4.1.1	Recurso	Dedicação
		0 - Alex	120
		5 - Leninha	120
		6 - Lima	120
		7 - Maiza	120
		8 - Rejane	120
		10 - Sherman	120
	45 - Correção dos Testes - Ciclo #4.1.1	Recurso	Dedicação
		1 - George	10
		2 - Ítalo	10
		4 - Jenner	10

Figura 48: Exemplo de um Cronograma Elaborado - Parte 5/8



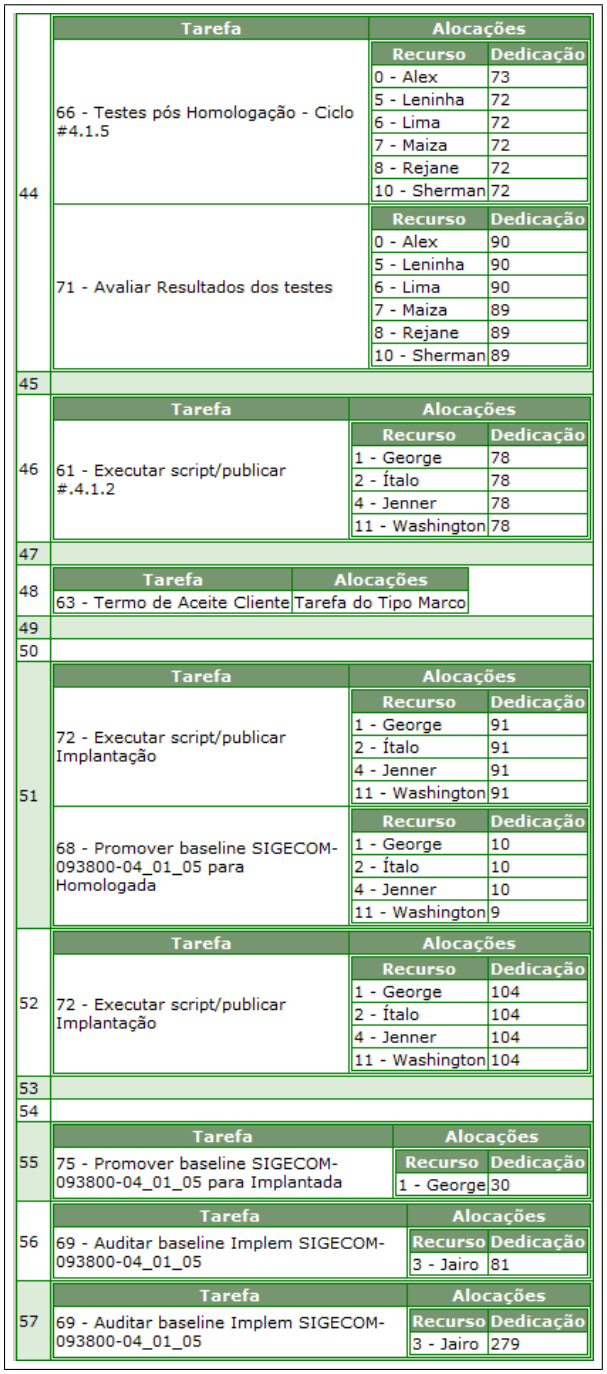
	Tarefa	Alocações	
33	44 - Testes - Ciclo #4.1.1	Recurso	Dedicação
		0 - Alex	310
		5 - Leninha	310
		6 - Lima	310
		7 - Maiza	310
		8 - Rejane	310
		10 - Sherman	309
	45 - Correção dos Testes - Ciclo #4.1.1	Recurso	Dedicação
		1 - George	420
		2 - Ítalo	420
		4 - Jenner	420
34	Tarefa	Alocações	
	45 - Correção dos Testes - Ciclo #4.1.1	Recurso	Dedicação
		1 - George	310
		2 - Ítalo	310
		4 - Jenner	309
	46 - Build 04.01.02 - Testes Ciclo #4.1.2	Recurso	Dedicação
		1 - George	60
	47 - Testes - Ciclo #4.1.2	Recurso	Dedicação
		0 - Alex	179
		5 - Leninha	179
		6 - Lima	179
		7 - Maiza	179
		8 - Rejane	179
		10 - Sherman	179
35	Tarefa	Alocações	
	48 - Correção dos Testes - Ciclo #4.1.2	Recurso	Dedicação
		1 - George	420
		2 - Ítalo	420
		4 - Jenner	420
36	Tarefa	Alocações	
	48 - Correção dos Testes - Ciclo #4.1.2	Recurso	Dedicação
		1 - George	118
		2 - Ítalo	118
		4 - Jenner	117
	49 - Build e publicação homol 04.01.03 - Testes Ciclo #4.1.3	Recurso	Dedicação
		1 - George	60
	50 - Testes em Homologação Ciclo #4.1.3	Recurso	Dedicação
		0 - Alex	120
		5 - Leninha	120
		6 - Lima	120
		7 - Maiza	120
		8 - Rejane	120
		10 - Sherman	120
37	Tarefa	Alocações	
	50 - Testes em Homologação Ciclo #4.1.3	Recurso	Dedicação
		0 - Alex	59
		5 - Leninha	59
		6 - Lima	59
		7 - Maiza	59
		8 - Rejane	59
		10 - Sherman	59
	51 - Correção dos Testes em Homol - Ciclo #4.1.3	Recurso	Dedicação
		1 - George	120
		2 - Ítalo	119
		4 - Jenner	119
	56 - Executar script/publicar #4.1.1	Recurso	Dedicação
		1 - George	51
		2 - Ítalo	51
		4 - Jenner	51
		11 - Washington	51

Figura 49: Exemplo de um Cronograma Elaborado - Parte 6/8



	Tarefa	Alocações	
38	51 - Correção dos Testes em Homol - Ciclo #4.1.3	Recurso	Dedicação
		1 - George	15
		2 - Ítalo	15
		4 - Jenner	15
	56 - Executar script/publicar #4.1.1	Recurso	Dedicação
		1 - George	123
		2 - Ítalo	123
		4 - Jenner	123
		11 - Washington	123
	52 - Build e publicação homol 04.01.04 - Testes Ciclo #4.1.4	Recurso	Dedicação
		1 - George	60
	53 - ReTestes em Homologação - Ciclo #4.1.4	Recurso	Dedicação
		0 - Alex	179
		5 - Leninha	179
		6 - Lima	179
		7 - Maiza	179
		8 - Rejane	179
		10 - Sherman	179
39			
40	Tarefa	Alocações	
	57 - Publicar após teste equipe #4.1.1	Recurso	Dedicação
		1 - George	120
41	Tarefa	Alocações	
	54 - Gerar e promover baseline Implem SIGECOM-093800-04_01_05 para Compilada e Testada	Recurso	Dedicação
		2 - Ítalo	60
42			
43	Tarefa	Alocações	
	59 - Fim 1a. Homologação pelo Cliente # 4.1.1	Tarefa do Tipo Marco	
	60 - Correção pós Homologação #4.1.4	Recurso	Dedicação
		1 - George	269
		2 - Ítalo	269
		4 - Jenner	269
	65 - Build e publicação homol 04.01.05 - Testes Ciclo #4.1.5	Recurso	Dedicação
		1 - George	20
		2 - Ítalo	20
		4 - Jenner	19
		11 - Washington	19
	66 - Testes pós Homologação - Ciclo #4.1.5	Recurso	Dedicação
		0 - Alex	71
		5 - Leninha	71
		6 - Lima	71
		7 - Maiza	71
		8 - Rejane	71
		10 - Sherman	71

Figura 50: Exemplo de um Cronograma Elaborado - Parte 7/8



	Tarefa	Alocações	
44	66 - Testes pós Homologação - Ciclo #4.1.5	Recurso	Dedicação
		0 - Alex	73
		5 - Leninha	72
		6 - Lima	72
		7 - Maiza	72
		8 - Rejane	72
		10 - Sherman	72
	71 - Avaliar Resultados dos testes	Recurso	Dedicação
		0 - Alex	90
		5 - Leninha	90
		6 - Lima	90
		7 - Maiza	89
		8 - Rejane	89
		10 - Sherman	89
45			
46	Tarefa	Alocações	
	61 - Executar script/publicar #.4.1.2	Recurso	Dedicação
		1 - George	78
		2 - Ítalo	78
		4 - Jenner	78
		11 - Washington	78
47			
48	Tarefa	Alocações	
	63 - Termo de Aceite Cliente	Tarefa do Tipo Marco	
49			
50			
51	Tarefa	Alocações	
	72 - Executar script/publicar Implantação	Recurso	Dedicação
		1 - George	91
		2 - Ítalo	91
		4 - Jenner	91
		11 - Washington	91
	68 - Promover baseline SIGECOM-093800-04_01_05 para Homologada	Recurso	Dedicação
		1 - George	10
		2 - Ítalo	10
		4 - Jenner	10
		11 - Washington	9
52	Tarefa	Alocações	
	72 - Executar script/publicar Implantação	Recurso	Dedicação
		1 - George	104
		2 - Ítalo	104
		4 - Jenner	104
		11 - Washington	104
53			
54			
55	Tarefa	Alocações	
	75 - Promover baseline SIGECOM-093800-04_01_05 para Implantada	Recurso	Dedicação
		1 - George	30
56	Tarefa	Alocações	
	69 - Auditar baseline Implem SIGECOM-093800-04_01_05	Recurso	Dedicação
		3 - Jairo	81
57	Tarefa	Alocações	
	69 - Auditar baseline Implem SIGECOM-093800-04_01_05	Recurso	Dedicação
		3 - Jairo	279

Figura 51: Exemplo de um Cronograma Elaborado - Parte 6/8

Gerar Novo Projeto Aleatório			
Quantidade Mínima de Recursos	10	Quantidade Máxima de Recursos	10
Quantidade de Tarefas Mínima	50	Quantidade de Tarefas Máxima	50
Quantidade de Habilidades	5		
Duração Mínima de Tarefa (Minutos)	30	Duração Máxima de Tarefa (Minutos)	6000
Quantidade Mínima de Habilidades por Recurso	3	Quantidade Máxima de Habilidades por Recurso	5
Quantidade Mínima de Habilidades por Tarefa	1	Quantidade Máxima de Habilidades por Tarefa	2
Salário/Minuto Mínimo	1	Salário/Minuto Máximo	10
Quantidade Mínima de Minutos Extra por Recurso	240	Quantidade Máxima de Minutos Extra por Recurso	480
Nível de Habilidade Mínimo	1	Nível de Habilidade Máximo	5

Figura 52: Parâmetros do Gerador de Instâncias Aleatórias

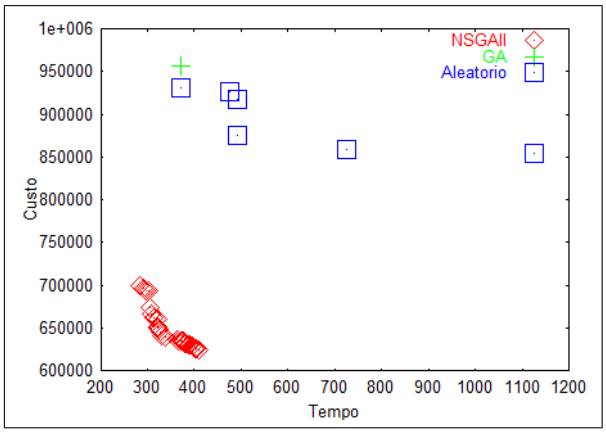


Figura 53: Projeto Sintético - Custo x Tempo

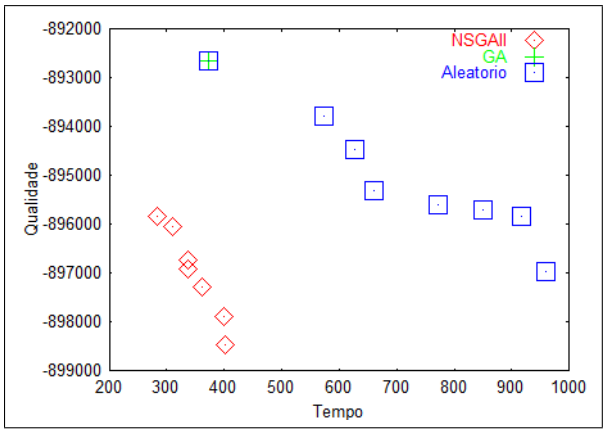


Figura 54: Projeto Sintético - Qualidade x Custo

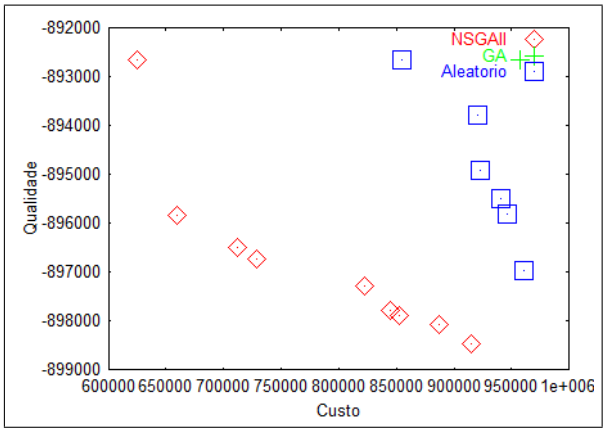


Figura 55: Projeto Sintético - Qualidade x Tempo

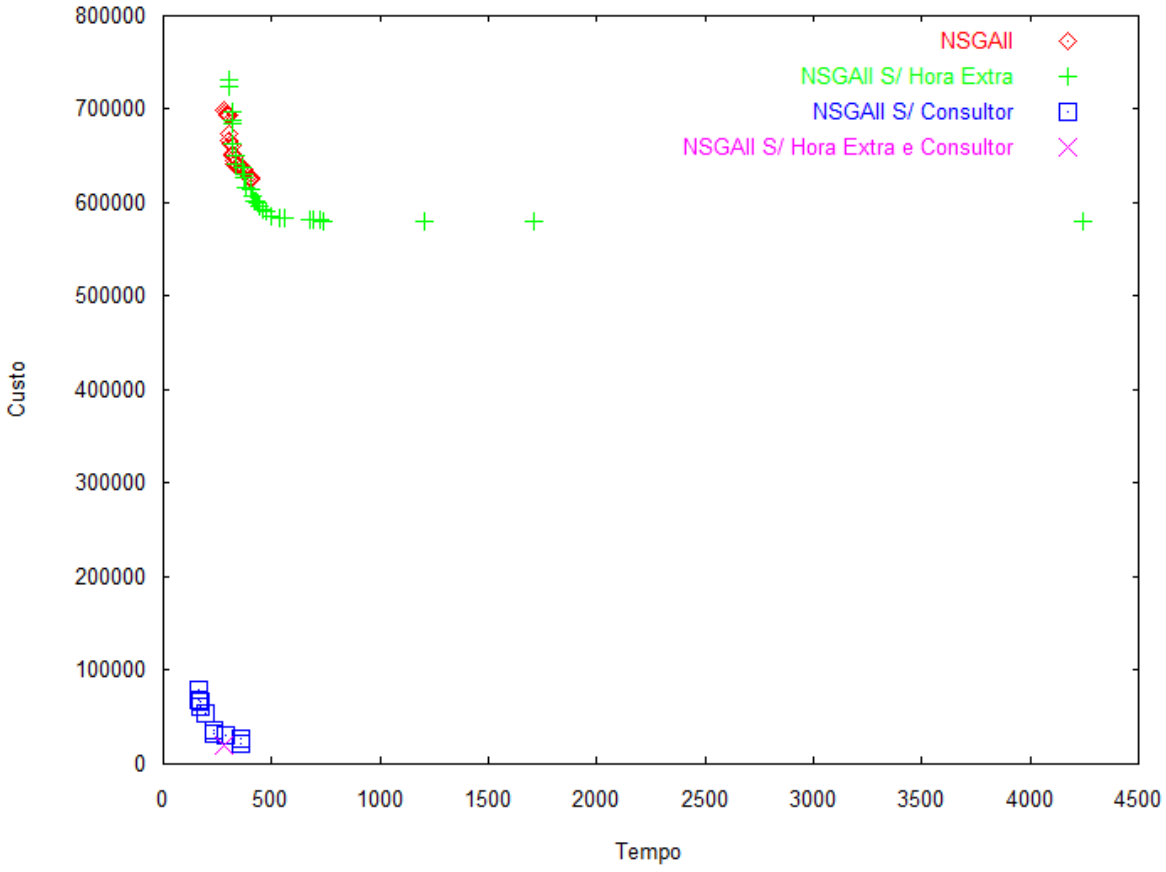


Figura 56: Projeto Sintético - Análise de Sensibilidade - Custo x Tempo

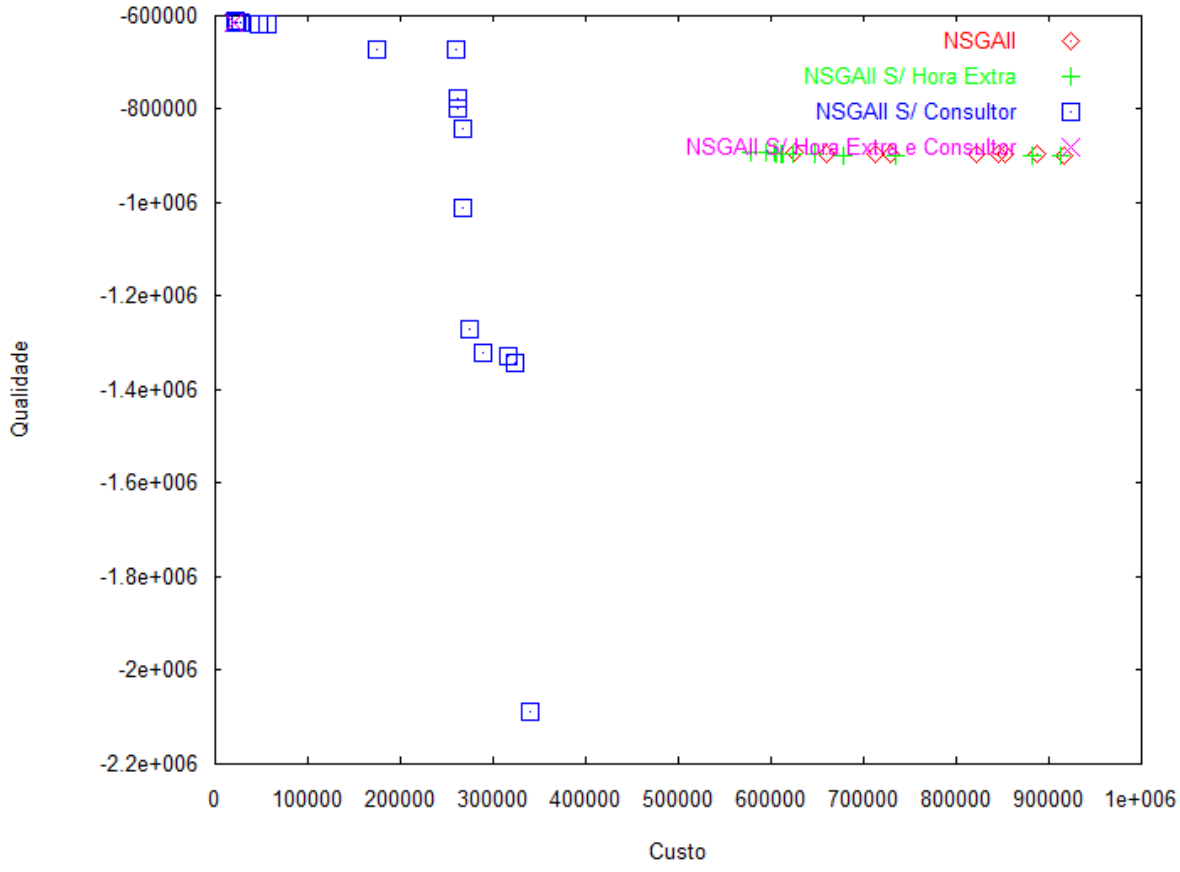


Figura 57: Projeto Sintético - Análise de Sensibilidade - Qualidade x Custo

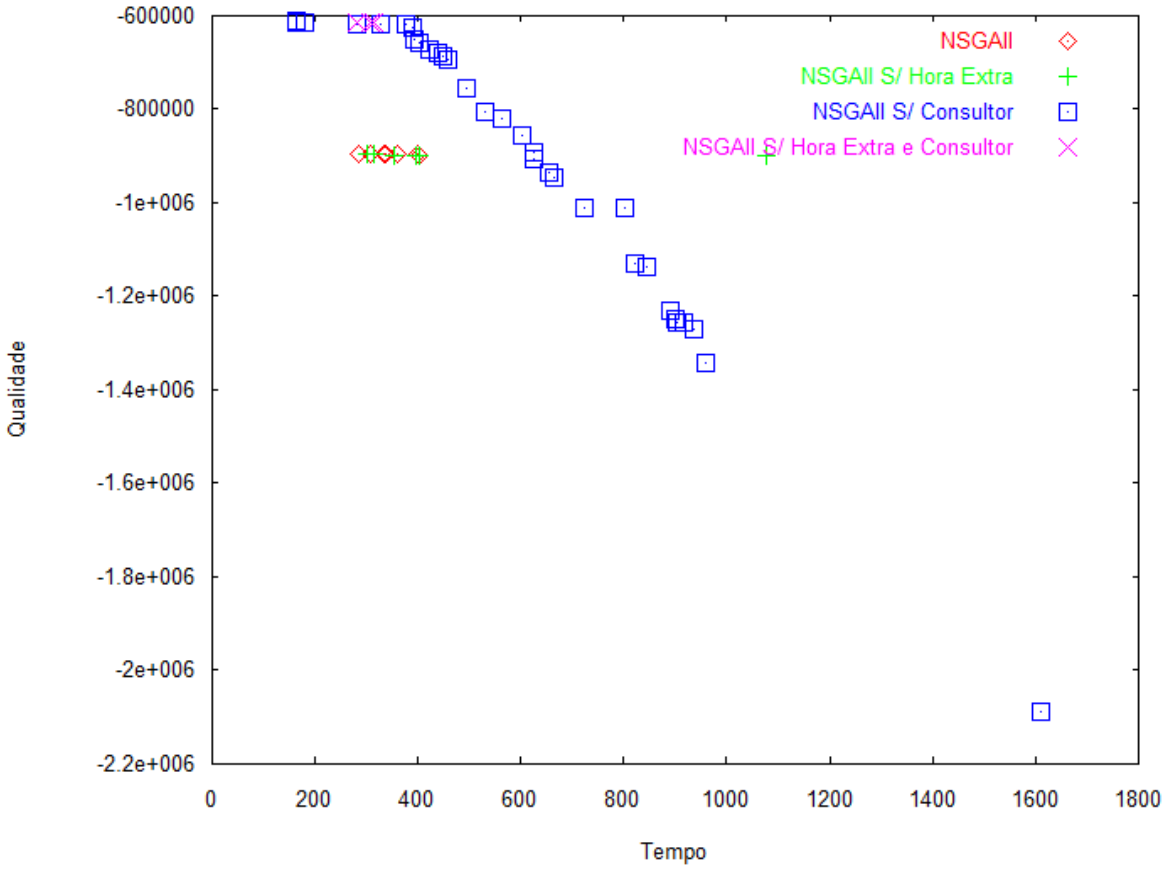


Figura 58: Projeto Sintético - Análise de Sensibilidade - Qualidade x Tempo

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 Conclusão

Neste trabalho propomos a utilização de uma abordagem otimizada híbrida, que envolve o Problema de Alocação de Equipes e o Problema de Escalonamento de Tarefas com Restrição de Recursos, a fim de elaborar bons cronogramas que minimize o tempo, o custo e maximizem a qualidade na formação das equipes nas atividades do projeto.

Duas versões para a modelagem foram propostas, uma mono-objetiva que utiliza como base de implementação o Algoritmo Genético e outra multiobjetiva que utiliza o NSGA-II. Vimos que, quando os objetivos são comparados separadamente, os resultados do NSGA-II foram bem superiores à solução construída com base na configuração planejada pelo gerente (*SGerente*). Mesmo quando comparados todos os objetivos simultaneamente, existe pelo menos uma solução do NSGA-II que é melhor (domina) em todos os sentidos (objetivos) que *SGerente*. Também observamos que a função objetivo do GA prioriza a redução do tempo em relação ao custo, e portanto, apesar de encontrar uma das melhores soluções quanto ao tempo, seu custo é um dos piores.

Foi realizada uma Análise de Sensibilidade da versão multiobjetiva proposta que analisou o efeito da eliminação dos consultores e das horas extra dos empregados em um projeto sintético. Concluiu-se que, para esse projeto, a eliminação das horas extras pouco influenciou os valores dos objetivos. No entanto, a eliminação dos consultores resultou em reduções significativas do custo do projeto, sem impactos no tempo. Isso mostra que a redução de 30% da quantidade de recursos humanos foi suficiente para o projeto trabalhado, não interferindo no seu tempo de conclusão. Concluímos também que a qualidade na formação das equipes pode variar dependendo dos níveis de habilidades dos recursos eliminados.

6.2 Contribuições

Esta pesquisa tem como principal contribuição a resolução do Problema de Elaboração de Cronograma abordando ao mesmo tempo o Problema de Alocação de Recursos Humanos e o Problema de Escalonamento de Projetos com Restrição de Recursos, aplicando para isso técnicas de otimização.

A modelagem utilizada procurou tornar a resolução condizente com situações reais, abordando conceitos como habilidades, hora extra, interdependência entre tarefas, formação de equipes em uma única tarefa, recursos alocados em tarefas paralelas (trabalhando no mesmo dia em mais de uma tarefa) e disponibilidade do recurso.

Outra importante contribuição é a apresentação de duas versões da modelagem proposta. A primeira versão utiliza o Algoritmo Genético mono-objetivo, que procura minimizar o custo do projeto. Além das horas pagas aos consultores e às horas extras pagas aos empregados, o custo do projeto também está relacionado com o pagamento salarial "diário" dos empregados. Dessa forma, a minimização do tempo também é um objetivo implícito que o algoritmo procura minimizar. A outra versão utiliza o NSGA-II, uma versão multi-objetiva do Algoritmo Genético. Nessa versão, os objetivos tempo, custo - horas extra dos empregados e horas trabalhadas pelos consultores - e qualidade das equipes alocadas são trabalhados de forma isoladas e independentes, aplicando o conceito não dominância discutidos no Capítulo 3.

Foi desenvolvida uma aplicação com interface gráfica para apoiar gerentes de projetos na elaboração de cronogramas. Nessa ferramenta, o gerente cadastra um projeto, suas atividades e os recursos humanos envolvidos. Para cada recurso humano, deve ser informada a carga horária, seu valor de hora de trabalho, os períodos de indisponibilidade ao longo do projeto e suas habilidades. Já as tarefas necessitam da duração em homens-hora, as habilidades exigidas e suas interdependências. Após um projeto ter sido completamente cadastrado, ele pode acionar o cálculo do cronograma entrando com alguns parâmetros, tais como o critério de parada e taxas de evolução.

Foi proposta a utilização de técnicas chamadas de *Datas de Liberação* e *Limites das*

Datas de Precedência a fim de evitar um custo muito alto de processamento durante o SGS Serial e permitir a utilização do minuto como unidade de tempo, obtendo uma modelagem mais flexível.

Testes computacionais foram realizados em dados artificiais (gerados aleatoriamente) e em um projeto real de uma empresa de grande porte de desenvolvimento de software. Foi realizada também uma análise de sensibilidade a fim de mostrar o comportamento da modelagem com relação às mudanças das configurações do projeto.

6.3 Trabalhos Futuros

Sugerimos como trabalho futuro a implementação de outras características pertinentes ao Problema de Elaboração de Cronograma, tais como planejamento multi-projetos, replanejamentos, alocações uniformes, alocações mutáveis, experiências profissionais adquiridas, curvas de aprendizado, treinamentos, produtividade individual, *overhead* de preparo de atividades, tarefas preemptivas e participação do gerente. Esses aspectos foram definidos na Tabela 3 do Capítulo 2.

Sabemos também que uma das mitigações realizadas no gerenciamento de risco é priorizar a execução das tarefas que possuem maior risco. Esta pesquisa tratou todas as atividades com a mesma prioridade, ficando esta questão para trabalhos futuros.

Outra questão importante é tornar a aplicação implementada nesse trabalho mais prática, permitindo ler e escrever dados diretamente de uma planilha do *Microsoft Project*.

BIBLIOGRAFIA

ALBA, Enrique; CHICANO, J. Francisco. Software project management with gas. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 177, n. 11, p. 2380–2401, 2007. ISSN 0020-0255.

ALCARAZ, J.; MAROTO, C. A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, Springer Netherlands, v. 102, p. 83–109, 2001. ISSN 0254-5330. 10.1023/A:1010949931021. Disponível em: <http://dx.doi.org/10.1023/A:1010949931021>.

ALVAREZ-VALDES, R.; TAMARIT, J. M. Heuristic algorithms for resource-constrained project scheduling: a review and na empirical analisys. In: ____. Amstrdam: R. Slowinski and J. Weglarz Ed, 1989. p. 113–134.

ANTONIOL, Giuliano; PENTA, Massimiliano Di; HARMAN, Mark. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In: *Proceedings of the Software Metrics, 10th International Symposium*. Washington, DC, USA: IEEE Computer Society, 2004. p. 172–183. ISBN 0-7695-2129-0. Disponível em: <http://portal.acm.org/citation.cfm?id=1018439.1021904>.

ANTONIOL, Giulio; PENTA, Massimiliano Di; HARMAN, Mark. Search-based techniques applied to optimization of project planning for a massive maintenance project. In: *In 21 st IEEE International Conference on Software Maintenance*. [S.l.]: IEEE Computer Society Press, 2005. p. 240–249.

BAKER, K. R. Introducing to sequencing and scheduling. In: ____. Wiley, New York: [s.n.], 1974.

BARRETO, Ahilton; BARROS, Márcio de O.; WERNER, Cláudia M.L. Staffing a software project: A constraint satisfaction and optimization-based approach. *Computers & Operations Research*, v. 35, n. 10, p. 3073 – 3089, 2008. ISSN 0305-0548. Part Special Issue: Search-based Software Engineering. Disponível em: <http://www.sciencedirect.com/science/article/B6VC5-4N0PPXS-5/2/28c127f3dbcfb5d08ff6dfffea10ce2a>.

BLAZEWICZ, J.; LENSTRA, J. K.; KAN, A. H. G. Rinnooy. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, v. 5, n. 1, p. 11 – 24, 1983. ISSN 0166-218X. Disponível em: <http://www.sciencedirect.com/science/article/B6TYW-46R0R6F-4/2/a67d403ffd23a6fe6276b7ee246a28ff>.

BOCTOR, F. F. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, v. 49, n. 1, p. 3–13, 1990. ISSN 0377-2217. Project Management and Scheduling. Disponível em: <http://www.sciencedirect.com/science/article/B6VCT-48MYJ8P-KT/2/758d4a769e515cc60da9fc92c4679d68>.

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, n. 2, p. 349–361, 1996. ISSN 0377-2217. Dis-

ponível em: <http://www.sciencedirect.com/science/article/B6VCT-3VVVR6Y-15/2/5a62f092ecc2b5f7bdb22f47ba0c8764>.

BOCTOR, F. F. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, v. 34, n. 8, p. 349–361, 1996. ISSN 0020-7543. Disponível em: <http://www.informaworld.com/10.1080/00207549608905028>.

BOULEIMEN, K.; LECOCQ, H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, v. 149, n. 2, p. 268–281, 2003. ISSN 0377-2217. Sequencing and Scheduling. Disponível em: <http://www.sciencedirect.com/science/article/B6VCT-47P1V7K-12/2/3828de7d4bb17e48494484620d4a49b7>.

BROOKS, Frederick P. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition)*. Anniversary. Addison-Wesley Professional, 1995. Paperback. ISBN 0201835959. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0201835959>.

BRUCKER, Peter et al. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 107, n. 2, p. 272–288, 1998. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/B6VCT-3VGMD1N-4/2/4a3052fdd1d9541d8b94aecb007b3c8b>.

BURDETT, Greg; LI, Raymond K.-Y. A quantitative approach to the formation of workgroups. In: *SIGCPR '95: Proceedings of the 1995 ACM SIGCPR conference on Supporting teams, groups, and learning inside and outside the IS function reinventing IS*. New York, NY, USA: ACM, 1995. p. 202–212. ISBN 0-89791-712-X.

CARRUTHERS, J. A.; BATTERSBY, A. Advances in critical path methods. *Operational Research Society*, v. 17, n. 4, p. 359–380, 1966. Disponível em: <http://www.jstor.org/stable/3007441>.

CHANG, Carl K. et al. Time-line based model for software project scheduling with genetic algorithms. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 50, p. 1142–1154, October 2008. ISSN 0950-5849. Disponível em: <http://portal.acm.org/citation.cfm?id=1405197.1405315>.

CHO, J. H.; KIM, Y. D. A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society*, v. 48, n. 7, p. 736–744, 1997. Disponível em: <http://www.jstor.org/stable/3010062>.

COFFMAN JR., E. G.; GAREY, M. R.; JOHNSON, D. S. Approximation algorithms for bin packing: a survey. In: ____. Boston, MA, USA: PWS Publishing Co., 1997. p. 46–93. ISBN 0-534-94968-1. Disponível em: <http://portal.acm.org/citation.cfm?id=241938.241940>.

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

COOPER, Dale F. Heuristics for scheduling resource-constrained projects: An experimental investigation. *MANAGEMENT SCIENCE*, v. 22, n. 11, p. 1186–1194, 1976. Disponível em: <http://mansci.journal.informs.org/cgi/content/abstract/22/11/1186>.

COOPER, D. F. A note on serial and parallel heuristics for resource-constrained project scheduling. *Foundations of Control Engineering*, v. 2 (24), p. 131–133, 1977.

DAVIS, Edward W.; PATTERSON, James H. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *MANAGEMENT SCIENCE*, v. 21, n. 8, p. 944–955, 1975. Disponível em: <http://mansci.journal.informs.org/cgi/content/abstract/21/8/944>.

DEB, Kalyanmoy D. et al. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, abr. 2002. ISSN 1089778X. Disponível em: <http://dx.doi.org/10.1109/4235.996017>.

DEMEULEMEESTER, Erik; HERROELEN, Willy. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Manage. Sci.*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 38, p. 1803–1818, December 1992. ISSN 0025-1909. Disponível em: <http://portal.acm.org/citation.cfm?id=161376.161411>.

DREXL, A.; GRUENEWALD, J. Nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, v. 25 (5), p. 74–81, 1993.

DURILLO, Juan J. et al. *jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics*. E.T.S.I. Informática, Campus de Teatinos, December 2006.

EPSTEIN, Sheldon; WILAMOWSKY, Yonah; DICKMAN, Bernard. Deterministic multiprocessor scheduling with multiple objectives. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 19, p. 743–749, November 1992. ISSN 0305-0548. Disponível em: <http://portal.acm.org/citation.cfm?id=147921.147931>.

FALKENAUER, E.; DELCHAMBRE, A.; DELCHAMBRE, E. Falkenauer A. A genetic algorithm for bin packing and line balancing. In: *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 1992. p. 1186–1192.

FREITAS, Fabrício et al. Aplicação de metaheurísticas em problemas da engenharia de software: Revisão de literatura. *II Congresso Tecnológico INfobrasil*, 2009.

GONçALVES, J.; MENDES, J. Um algoritmo genético para o problema do sequenciamento de projetos com recursos limitados. In: *Actas do II Congresso Luso-Moçambicano de Engenharia*. [s.n.], 2001. Disponível em: <http://10.255.0.115/pub/2001/GM01>.

HARMAN, Mark. The current state and future of search based software engineering. In: *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007. p. 342–357. ISBN 0-7695-2829-5.

HARMAN, Mark; JONES, Bryan F. Search-based software engineering. *Information & Software Technology*, v. 43, n. 14, p. 833–839, 2001.

HARTMANN, S. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, v. 45, p. 733–750, 1997.

HARTMANN, Sönke. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*, Wiley Subscription Services, Inc., A Wiley Company, v. 49, n. 5, p. 433–448, 2002. ISSN 1520-6750. Disponível em: <http://dx.doi.org/10.1002/nav.10029>.

ICHIHARA, A. Solução do resource-constrained project scheduling problem mediante heurísticas de programação baseadas em regras de prioridade. XXI Encontro Nacional de Engenharia de Produção, 2001.

ICHIHARA, A. O problema de programação de projetos com restrição de recursos. XXII Encontro Nacional de Engenharia de Produção, 2002.

KOLISCH, Rainer. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, v. 90, n. 2, p. 320–333, April 1996. Disponível em: <http://ideas.repec.org/a/eee/ejores/v90y1996i2p320-333.html>.

KOLISCH, R.; HARTMANN, S. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In: ____. Kluwer, Dordrecht: [s.n.], 1999. p. 147–178.

KOLISCH, Rainer; SPRECHER, Arno; DREXL, Andreas. Characterization and generation of a general class of resource-constrained project scheduling problems. *Manage. Sci.*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 41, p. 1693–1703, October 1995. ISSN 0025-1909. Disponível em: <http://portal.acm.org/citation.cfm?id=222088.222105>.

KURTULUS, I.; DAVIS, E. W. Multi-project scheduling: Categorization of heuristic rules performance. *MANAGEMENT SCIENCE*, v. 28, n. 2, p. 161–172, 1982. Disponível em: <http://mansci.journal.informs.org/cgi/content/abstract/28/2/161>.

LAWRENCE, S. R. *Resource constrained project scheduling - A computational comparison of heuristic scheduling techniques*. [S.l.], 1985.

LEE, J. K.; KIM, Y. D. Search heuristics for resource constrained project scheduling. *Journal of the Operational Research Society*, v. 47, n. 5, p. 678–689, 1996. Disponível em: <http://www.jstor.org/stable/3010018>.

LI, S. New approach for optimization of overall construction schedule. v. 122, p. 7–13, 1996.

MARINHO, D. *Uma Aplicação do Algoritmo Genético Multiobjetivo NSGA II Para Seleção de Imagens de Satélite de Trechos de Mata Atlântica*. Dissertação (Graduação) — Monografia em Bacharel em Engenharia da Computação da Universidade de Pernambuco, 2010.

MERKLE, D.; MIDDENDORF, M.; SCHMECK, H. Ant colony optimization for resource-constrained project scheduling. In: *IEEE Transactions on Evolutionary Computation*. [S.l.]: Morgan Kaufmann, 2000. p. 893–900.

MILLER, W.; SPOONER, D.L. Automatic generation of floating-point test data. *IEEE Transactions on Software Engineering*, IEEE Computer Society, Los Alamitos, CA, USA, v. 2, p. 223–226, 1976. ISSN 0098-5589.

MINGOZZI, Aristide et al. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Manage. Sci.*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 44, p. 714–729, May 1998. ISSN 0025-1909. Disponível em: <http://portal.acm.org/citation.cfm?id=293898.293907>.

MODER, Joseph J.; PHILLIPS, Cecil R.; DAVIS, Edward W. *Project Management with CPM, Pert & Precedence Diagramming*. [S.l.]: Blitz, 1995.

PENTA, Massimiliano Di et al. *The Effect of Communication Overhead on Software Maintenance Project Staffing: a Search-Based Approach*. 2007.

PMI. *A Guide To The Project Management Body Of Knowledge (PMBOK Guides)*. [S.l.]: Project Management Institute, 2008.

QURESHI, F.; HARMAN, M. *Search-based Approaches to Understanding Project Management Choices*. [S.l.], 2006.

SCHRAGE, L. *Solving Resource-Constrained Network Problems by Implicit Enumeration Preemptive Case*. [S.l.]: Operations Research, 1972. 263–278 p.

SIMõES, J. *Algoritmos Baseados em Busca Tabu e Busca Tabu Reativa para Problemas Generalizados de Programação de Projetos*. Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e de Computação, UNICAMP, 2004.

SIMPSON, Wendell P.; PATTERSON, James H. A multiple-tree search procedure for the resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 89, n. 3, p. 525–542, 1996. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/B6VCT-3VW1D20-9/2/3138851ad3941771ac8f477022a511b2>.

SLOWINSKI, R. Two approaches to problems of resource allocation among project activities: A comparative study. *Journal of the Operational Research Society*, v. 31 (8), p. 711–723, 1980.

SLOWINSKI, R. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, v. 7, n. 3, p. 265 – 273, 1981. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/B6VCT-48NBJFD-2MR/2/87cfc0a3f64b27bda7580138dbcaa5af>.

SPRECHER, Arno. Scheduling resource-constrained projects competitively at modest memory requirements. *MANAGEMENT SCIENCE*, v. 46, n. 5, p. 710–723, 2000. Disponível em: <http://mansci.journal.informs.org/cgi/content/abstract/46/5/710>.

SRINIVAS, N.; DEB, Kalyanmoy. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, v. 2, p. 221–248, 1994.

TCHAO, C. *Heurísticas para o Problema de Escalonamento de Projetos com Restrição de Recursos*. Dissertação (Mestrado) — Dissertação de Mestrado em Ciência da Computação da Universidade Federal Fluminense, 2007.

THOMAS, Paul R.; SALHI, Said. A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics*, Springer Netherlands, v. 4, p. 123–139, 1998. ISSN 1381-1231. 10.1023/A:1009673512884. Disponível em: <http://dx.doi.org/10.1023/A:1009673512884>.

VACA, O. C Lopez. *Um Algoritmo Evolutivo para a Programação de Projetos Multi-Modos com Nivelamento de Recursos Limitados*. Dissertação (Doutorado) — Universidade Federal de Santa Catarina, UFSC, 1995.

VALLS, Vicente; QUINTANILLA, Sacramento; BALLESTIN, Francisco. Resource-constrained project scheduling: A critical activity reordering heuristic. *European Journal of Operational Research*, v. 149, n. 2, p. 282–301, 2003. Disponível em: <http://econpapers.repec.org/RePEc:eee:ejores:v:149:y:2003:i:2:p:282-301>.

VIANA, V. *Meta-Heurísticas e Programação Paralela em Otimização Combinatória*. [S.l.]: UFC Edições, 1998.

VILLELA, A. *Novos Algoritmos Heurísticos e Híbridos para o Problema de Escalonamento de Projetos com Restricao de Recursos Dinâmicos*. Tese (Doutorado) — Tese de Doutorado em Computação da Universidade Federal Fluminense, 2010.

WEGLARZ, Jan et al. Algorithm 520: An automatic revised simplex method for constrained resource network scheduling [h]. *ACM Trans. Math. Softw.*, ACM, New York, NY, USA, v. 3, p. 295–300, September 1977. ISSN 0098-3500. Disponível em: <http://doi.acm.org/10.1145/355744.355755>.

WEN, Feng; LIN, Chi-Ming. Multistage human resource allocation for software development by multiobjective genetic algorithm. *The Open Applied Mathematics Journal*, v. 2, p. 95–103, 2008.

XANTHAKIS, S. et al. Application of genetic algorithms to software testing. *Proceedings of the 5th International Conference on Software Engineering and Applications*, p. 625–636, 1992.

ÖZDAMAR, L. A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Aplications and Reviews*, v. 29, p. 268–281, 1999. ISSN 1094-6977.