COPPE/UFRJ

OTIMIZAÇÃO DE UM CASO REAL DE ALOCAÇÃO DE EQUIPES DA PETROBRAS

Marcos Henrique de Azevedo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Márcia Helena Costa Fampa

Rio de Janeiro

Setembro de 2010

OTIMIZAÇÃO DE UM CASO REAL DE ALOCAÇÃO DE EQUIPES DA PETROBRAS

Marcos Henrique de Azevedo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof^a. Márcia Helena Costa Fampa, D.Sc.

Prof. Nelson Maculan, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2010

Azevedo, Marcos Henrique de

Otimização de Um Caso Real de Alocação de Equipes da Petrobras/Marcos Henrique de Azevedo. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIII, 66 p.: il.; 29, 7cm.

Orientador: Márcia Helena Costa Fampa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 61 – 66.

1. Problema de Alocação. 2. GRASP. 3. VND. 4. Metaheurística. 5. Otimização. I. Fampa, Márcia Helena Costa. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico este trabalho a minha

família, meu bem mais valioso.

Agradecimentos

Agradeço...

A Deus, por tudo que tenho e acontece em minha vida.

A minha esposa Cláudia, por me incentivar e cuidar de nossa filha Letícia durante o período de realização deste trabalho.

Aos meus pais Arylton Mauro de Azevedo e Rosa Caputo de Azevedo, pelo amor e educação que me deram.

A minha família, em especial minha irmã Marcia, Marcelo e Rafael, pela amizade e companherismo.

Aos amigos, pela compreensão de minha ausência neste período.

Aos amigos de mestrado Tiago, Jurair, Jesus, Michael e Francisco.

À orientadora Marcia Fampa, por sua presteza, apoio, incentivo e paciência.

Ao professor Adilson Xavier, pela acolhida na UFRJ e incentivo.

Ao professor Satoru, membro da banca, pela participação e contribuição no início do meu mestrado na UFF.

Ao professor Nelson Maculan, membro da banca, pela participação.

Aos professores da UFRJ, pelo conhecimento compartilhado ao longo das disciplinas.

Ao gerente da Petrobras Luiz Antonio Pereira de Araujo, pelo apoio e incentivo.

Ao gerente da Petrobras Roberto Iachan, por me oferecer o apoio de sua equipe.

Ao amigo de trabalho Mayron, por sua paciência e presteza.

Aos amigos de trabalho Dalila, Daniel e Marcel, pelos ensinamentos e conhecimentos compartilhados.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

OTIMIZAÇÃO DE UM CASO REAL DE ALOCAÇÃO DE EQUIPES DA PETROBRAS

Marcos Henrique de Azevedo

Setembro/2010

Orientador: Márcia Helena Costa Fampa

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta um módulo de otimização para resolver um problema de alocação de equipes. Este módulo de otimização utiliza a metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) para construir uma solução inicial e a metaheurística *Variable Neighborhood Descent* (VND) no procedimento de busca local para melhorar a solução construída, utilizando duas estruturas de vizinhanças distintas. Quando uma solução com tarefas não completamente alocadas é gerada, utiliza-se dois procedimentos suplementares à fase de busca local para tentar alocar estas tarefas e melhorar a solução. Após a fase de busca local, se todas as tarefas forem atendidas, a solução receberá outro tratamento de refinamento. Este trabalho é ilustrado através de um caso real de alocação de equipes de especialistas em atividades de assessoramento técnico, qualificação e certificação da área de Engenharia da Petrobras. Optou-se por essa abordagem por ser um problema *NP-Difícil*.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

OPTIMIZATION OF A REAL CASE OF TEAM ALLOCATION AT THE PETROBRAS

Marcos Henrique de Azevedo

September/2010

Advisor: Márcia Helena Costa Fampa

Department: Systems Engineering and Computer Science

This work presents a optimization module to solve a allocation of staffs problem. This optimization module uses the *Greedy Randomized Adaptive Search Procedure* (GRASP) algorithm to construct an initial solution and the *Variable Neighborhood Descent* (VND) algorithm into the procedure local search to improve a constructed solution, using two distinct neighborhood structures. When a solution with incomplete tasks is generated, is used two supplement procedures to the local search phase to try allocate that tasks and to improve the solution. After the local search phase, if all tasks are allocated, the solution will receive another treatment of refinement. This work is illustrated by a real case of allocation of specialists staff in activities of technique assistance, qualification and certification of the Petrobras Engineering Group. Choose this approach to be a *NP-Hard* problem.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algorítmos

Capítulo 1

Introdução

A busca da excelência em Segurança, Meio Ambiente e Saúde (SMS), objetivo previsto em seu Plano Estratégico, levou a Petrobras a estabelecer como uma de suas metas a certificação de suas unidades de acordo com normas internacionais de gestão de SMS.

Esta certificação envolve as atividades de Implementação de Empreendimentos (compreendendo Aquisição de Bens e Serviços; Logística; Fiscalização de Projeto Executivo, Construção e Montagem, Condicionamento, Testes; Assistência para Pré-Operação e Entrega das Instalações) desenvolvidas nas unidades localizadas em várias Unidades de Implementação de Empreendimentos (UIEs) em todo o território nacional.

O problema de alocação de equipes consiste no problema em alocar resursos nas atividades citadas acima. Recursos são pessoas que pertencem a equipes de especialistas, integrantes de uma das áreas de engenharia da Petrobras que tem por fim atuar nas atividades de assessoramento técnico, qualificação e certificação especificadas no parágrafo anterior. Estas atividades são chamadas de tarefas.

A alocação de equipes exige de seus responsáveis um planejamento contínuo do seu setor e dos recursos pertencentes a ele. Tal planejamento visa: programação das equipes; alocação, escalonamento, treinamento, especialização, aprimoramento e contratação dos recursos; reprogramação em virtude da necessidade do cliente, imprevistos dos recursos e atrasos; além de outros.

Tais tomadas de decisão variam sob diversos aspectos, como por exemplo: desde um planejamento de longo prazo, como aumento no quadro de funcionários, cursos de especialização para os recursos, até um planejamento de curto prazo, como alocação dos recursos nas tarefas, aumento da graduação dos recursos; desde decisões pontuais, como treinamento de um recurso, até decisões mais amplas, como contratação de recursos especialistas; desde decisões que necessitem de um prazo maior para serem planejadas, como obtenção de qualificação para executar novas tarefas, até decisões emergenciais, como replanejamento e/ou substituição dos recursos em função da necessidade da demanda ou imprevistos dos recursos.

As grandes empresas que possuem uma visão bem definida de seus objetivos, e que dentre esses objetivos buscam se sobressair perante as demais empresas concorrentes no mercado nacional e internacional, sabem da necessidade de utilização de um sistema que as apoiem em suas tomadas de decisões, para a otimização do aproveitamento dos seus recursos em todos os processos.

Mesmo que não houvesse concorrência nem mesmo a necessidade de superar desafios, um simples planejamento de alocação dos recursos nas tarefas não é um trabalho trivial. Porém, há de se preocupar sempre com o aumento da qualidade na realização das tarefas e com a redução dos custos.

A exigência dos clientes, normalmente, tende a aumentar com o passar do tempo, uma vez que os seus resultados dependem direta ou indiretamente da realização das tarefas demandadas por eles. Esses resultados envolvem lucros, reduções de custos, qualidade, cargos de confiança, entre outros. Para todos os casos, um planejamento bem definido deve ser realizado para se conquistar melhores resultados, ou até mesmo manter os já realizados.

Tendo em vista os pontos descritos acima, observa-se facilmente o quanto que um planejamento de uma equipe é importante.

Há ainda de se considerar o fato da existência de complicadores ligados à legislação trabalhista e à CIPA (Comissão Interna de Prevenção de Acidentes). Daí, surge a necessidade de trabalhos como este, dedicados ao desenvolvimento de

soluções que tratem ou auxiliem no tratamento de todos os assuntos envolventes.

Este trabalho concentra-se exclusivamente no desenvolvimento de um módulo de otimização para o tratamento do problema de alocação de equipes. Esta abordagem se deve à necessidade de se criar uma alocação viável dos recursos nas tarefas, aproveitando ao máximo, basicamente, os recursos disponíveis e reduzindo os custos sempre que possível, e atendendo ao máximo de tarefas solicitadas, respeitando os requisitos e as restrições do problema.

A necessidade de desenvolver um módulo para otimizar o serviço de alocação dos recursos às tarefas surgiu da dificuldade de construir uma alocação com o máximo de aproveitamento dos recursos e o máximo de tarefas atendidas.

Entretanto, essa não é uma tarefa trivial, a qual deve gerar um conjunto considerável de restrições [9], além de que nenhum requisito considerado forte pode ser violado, e os requisitos considerados fracos devem ser atendidos sempre que possível em função de uma solução com maior qualidade.

A equipe de Pesquisa Operacional da Petrobras foi solicitada pela área de Engenharia da Petrobras, responsáveis pelo problema de alocação de equipes, para desenvolver um módulo de otimização para este problema, utilizando Programação Matemática.

Após o desenvolvimento de tal módulo, foram realizados vários testes considerando cenários reais e fictícios, com diversas configurações do algoritmo. Os resultados obtidos mostram que o algoritmo proposto fornece soluções ótimas ou quase ótimas em tempos aceitáveis, variando de poucos segundos a algumas horas conforme o tamanho do problema - diretamente relacionado ao número de técnicos, tarefas, locais de realização e das competências requeridas.

Atualmente a alocação dos recursos é realizada manualmente. Não existe nenhum sistema que possibilite algum tipo de otimização em qualquer aspecto. O único dispositivo utilizado para tentar alocar as equipes da melhor maneira possível é a experiência dos profissionais responsáveis pela alocação.

Além de ser um problema que trata de uma situação real, o prolema de alocação

de equipe apresenta grande complexidade devido ao grande número de restrições envolvidas.

Diante dessas circunstâncias, as formas tradicionais e exatas de resolução exigem um alto poder computacional para percorrer todo o espaço solução a fim de encontrar a solução ótima global. Atualmente, é possível utilizar pacotes modernos que utilizam programação matemática para resolução desse tipo de problema. Porém, o alto custo normalmente implica na inviabilidade de investimento dessas ferramentas, além do mais, a sua utilização não é trivial.

O Problema de Alocação de Equipes é um problema de otimzação que possui semalhanças com o Problema de Quadro de Horários, ou *timetabling*, o qual pertence à classe NP-Difícil [28]. Por sua vez, a respeito do Problema de Quadro de Horários, sabe-se que não existe um algoritmo capaz de resolvê-lo em tempo polinomial, a menos que P = NP [13]. Algumas soluções de destaque para o problema de programação de quadro de horários pode ser vista em [36, 44, 46, 48]. Assim, com o objetivo de reduzir o tempo e custo computacional gasto na resolução desse tipo de problema, tem-se estudado o uso de técnicas de inteligência computacional [27]. As metaheurísticas [35, 50], uma desssas técnicas, têm sido bastante aplicadas, se destacando nos problemas de alta complexidade, principalmente nos de natuzera combinatória.

Essas técnicas propõem uma busca inteligente em sub-espaços do espaço solução, a partir de uma solução viável contruída inicialmente. A redução do espaço de busca diminui expressivamente o custo operacional. Porém, vale ressaltar que tais técnicas não garantem a solução global para o problema. Mesmo assim, os resultados obtidos são aceitáveis e de boa qualidade, desde que se faça um bom levantamento dos requisitos exigidos e não haja necessidade de obtenção da solução global.

As técnicas de metaheurísticas se definem pela aplicação de mais de uma heurística com o objetivo de escapar dos mínimos locais e otimizar a solução. Não existe um desenvolvimento único para um tipo de problema, ou seja, cada necessidade exige um projeto específico que contemple todos os seus objetivos e restrições.

Porém, as modelagens de vários problemas podem ser adaptadas a partir de outros problemas semelhantes, pois contemplam modelagens compatíveis e essas técnicas são muito flexíveis.

Podemos citar como exemplos de metaheurísticas os Algoritmos Genéticos (AG) [19], Busca Tabu (BT) [14–16, 18], *Simullated Anealling* (SA) [23, 24], *Greedy Randomized Adaptive Search Procedure* (GRASP) [10, 11, 30, 33, 34], *Iterated Local Search* (ILS) [17], entre outras.

Na literatura, os problemas de Programação de Quadro de Horários têm recebido grande interesse por parte dos pesquisadores em computação. Pode-se destacar alguns trabalhos referentes à otimização da solução destes problemas, como por exemplo: [36] apresenta uma solução heurística híbrida baseada em Busca Tabu e uma solução utilizando métodos baseados em Programação Linear Inteira Mista; [44] desenvolve uma heurística de busca local baseada em caminhos mínimos; [48] propõe um algoritmo GRASP que usa a particularidade semi-gulosa para construir a solução inicial e tenta melhorá-la usando um algoritmo Tabu Search; [46] apresenta uma técnica de busca local que, baseada em movimentos dos horários das aulas das turmas, a partir de uma solução viável inicial gera soluções melhores através de detecção de ciclos de custo negativo nos grafos associados aos quadros de horário.

Tendo em vista os fatos supra citados, a necessidade de desenvolver soluções eficientes é, portanto, de grande importância. Neste trabalho será apresentado o Modelo Matemático para o problema de alocação de equipes, além de duas abordagens distintas para solução do problema. Ambas apresentam o uso da metaheurística GRASP. Na primeira, a busca local baseia-se em uma busca local simples, e a segunda baseia-se na busca local Variable Neighborhood Descent (VND) [20, 21]. Ao final, veremos um estudo comparativo entre os resultados dos métodos propostos neste trabalho e a Programação Matemática. Os resultados da Programação Matemática foram produzidos através do pacote de *software* CPLEX.

1.1 Organização do texto

Os demais capítulos encontram-se organizados da seguinte forma: no Capítulo 2 descreve-se o problema real de alocação de equipes. No Capítulo 3 apresenta-se o modelo matemático do problema abordado. No Capítulo 4 apresenta-se uma revisão bibliográfica das metodologias aqui aplicadas estabelecendo, assim, pilares essenciais para o desenvolvimento deste trabalho. Por meio do Capítulo 5, apresentam-se as metodologias aplicadas utilizando-se uma abordagem direcionada ao estudo de caso que aqui se faz. Através do Capítulo 6, apresentam-se todos os resultados produzidos por meio dos experimentos realizados. E por fim, no Capítulo 7, tratam-se as conclusões e considerações finais, onde também apresentam-se proposicões, as quais vislumbram-se desenvolver em trabalhos futuros.

Capítulo 2

O Problema de Alocação de Equipes

O Problema de Alocação de Equipes, enfocado neste trabalho, consiste basicamente em um problema de alocar resursos em tarefas, e surgiu da necessidade de desenvolver um módulo para otimizar este serviço, devido à dificuldade de construir um quadro de horários com o máximo de aproveitamento dos recursos.

Neste trabalho consideramos o problema que surgiu da necessidade da área de engenharia da Petrobras a qual é responsável por assegurar a confiabilidade das Unidades de Implementação de Empreendimentos (UIE's) através das atividades de Assessoramento Técnico em processos de inspeção, fabricação, construção e montagem, do desenvolvimento, implementação da qualificação e certificação de recursos utilizados em processos de inspeção e controle da qualidade. A Figura 2.1 ilustra uma parte do organograma da área de engenharia.

O objetivo do trabalho é desenvolver um módulo de otimização, com a finalidade de definir uma alocação viável dos recursos às tarefas, visando otimizar os seguintes ítens: maximizar o número de tarefas programadas, minimizar a distância total percorrida pelos recursos, minimizar o *giveaway* (diferença entre o nível de graduação do recurso alocado na competência e o nível de graduação da competência requerida), entre outros requisitos.

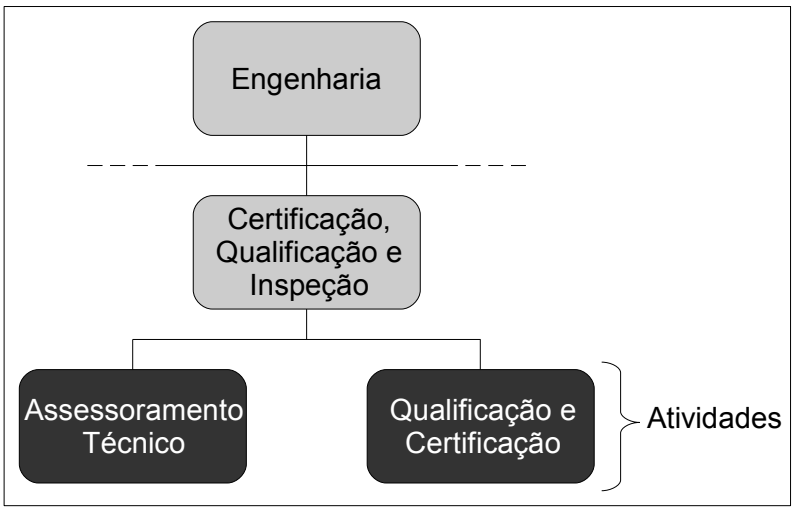

Figura 2.1: Parte do organograma da área de engenharia

As atividades de assessoramento técnico e de qualificação e certificação mostradas na fig 2.1 são também chamadas de tarefas. As atividades ou tarefas correspondem a um conjunto de competências.

As competências requerem uma especialização por parte do profissional que irá executá-la. Além disto, cada competência possui vários níveis de especialização na qual deverá ser classificada. Este nível de especialização deverá ser informado no momento em que se solicitar a competência.

As competências das tarefas, chamadas Competências Requeridas, devem ser executadas por recursos que possuem tal especialização. Cada competência requerida exige um nível de graduação e um quantitativo de recursos necessários. Cada tarefa será realizada em local e período pré-determinados e possui um peso de acordo com a sua importância. Cada tarefa poderá ser atendida por mais de um recurso, de acordo com a necessidade requerida.

A Figura 2.2 ilustra um exemplo sobre os principais dados referentes às tarefas, conforme descrito: (a) identificação da tarefa; (b) identificação da competência; (c) nível da competência da tarefa. A tarefa poderá requerer mais de uma competência com níveis distintos ou não. Competência sem valor de nível informado significa que a tarefa não requisitou tal competência. A tarefa que não tiver todas as suas competências alocadas, não será executada; (d) localidade em que será executada

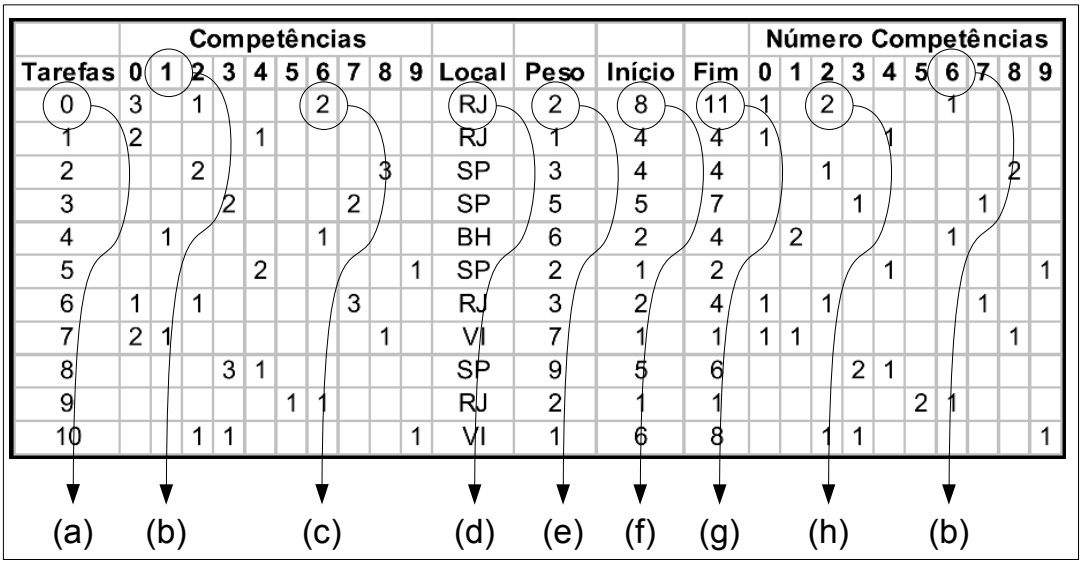

Figura 2.2: Exemplo de informações sobre tarefas: (a) identificação da tarefa; (b) identificação da competência; (c) nível da competência da tarefa; (d) localidade da tarefa; (e) importância da tarefa; (f) dia de início da tarefa; (g) dia de término da tarefa; (h) número de recursos necessários a serem alocados na competência da tarefa

a tarefa; (e) peso ou importância da tarefa; (f) dia de início da tarefa; (g) dia de término da tarefa; (h) número de recursos necessários a serem alocados na competência da tarefa. Caso seja alocado um número inferior ao necessário ou nenhum, a tarefa não será executada. Todos os dados referentes às tarefas podem ser vistos na seção 3.2, Base de Dados, no Capítulo 3, Modelo Matemático.

As datas definidas para que as tarefas sejam executadas são fixas, ou seja, não poderão ser alteradas. A unidade de tempo utilizada é o dia. Ao final do processo de alocação, caso a tarefa não seja alocada, a data de execução da tarefa poderá ser antecipada ou postergada em algumas unidades de tempo, com anuência do requisitante. Com isto, em um segundo momento, por meio de um segundo módulo o qual não é abordado neste trabalho, a tarefa poderá ser alocada.

Os recursos são profissionais especialistas que pertencem a equipes integrantes da área de engenharia da Petrobras que tem por finalidade atuar nas atividades citadas anteriormente. Os recursos possuem especializações ou competências, chamadas Competências Disponíveis, a fim de atender às competências requeridas das tarefas. As competências dos recursos são graduadas de acordo com o nível de graduação

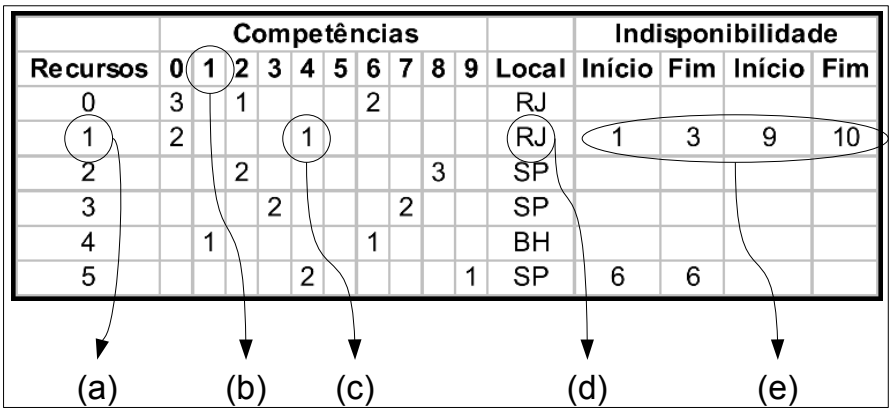

Figura 2.3: Exemplo de informações sobre recursos: (a) identificação do recurso; (b) identificação da competência; (c) nível do recurso na competência; (d) localidade do recurso; (e) períodos de indisponibilidade do recurso

de cada especialista. Cada recurso pode possuir mais de uma competência com graduações distintas entre si ou não.

Os recursos, inicialmente, estão alocados em uma localidade e podem possuir períodos de indisponibilidade. Estes períodos de indisponibilidade dos recursos são em função de que o recurso pode estar em curso, em treinamento, de férias, ou de algum outro motivo que o impossibilte de executar tarefas em tal período.

A Figura 2.3 ilustra um exemplo sobre os principais dados referentes aos recursos, conforme descrito: (a) identificação do recurso; (b) identificação da competência; (c) nível do recurso na competência. O recurso poderá atender mais de uma competência na tarefa. Recurso sem valor de nível informado significa que o recurso não possui tal competência; (d) localidade onde estará o recurso no início da execução das tarefas - normalmente na localidade de sua residência ou de seu trabalho; (e) períodos de indisponibilidade do recurso, ou seja, nesse período o recurso não executará nenhuma tarefa. Todos os dados referentes aos recursos podem ser vistos na seção 3.2, Base de Dados, no Capítulo 3, Modelo Matemático.

Um recurso poderá ser alocado em uma tarefa se houver compatibilidade entre eles. A compatibilidade entre recursos e tarefas contempla as seguintes situações: o recurso possui nível de competência igual ou superior ao nível da competência requerida pela tarefa; o recurso está disponível no período de execução da tarefa.

2.1 Identificação das etapas e das restrições

Esta seção baseia-se em entrevistas realizadas por mim a integrantes da Equipe de Pesquisa Operacional da Petrobras.

O objetivo da alocação de equipes visa atender todas ou o máximo possível de tarefas, considerando as competências requeridas nas tarefas, as competências disponíveis nos recursos e respeitando as restrições do problema. Caso alguma tarefa não seja alocada, esta poderá ser alocada em uma segunda etapa do problema não tratada neste trabalho.

No planejamento da alocação de equipes, a fase de alocação de equipes é realizada após as fases de identificação de recursos e tarefas. Por sua vez, normalmente, a fase de identificação de competências se inicia anteriormente a todas, podendo se estender até as fases de identificação dos recursos e das tarefas.

A seguir é descrito sucintamente cada umas das etapas do planejamento para a alocação de equipes, para melhor entendimento.

Etapa de identificação de competências É a identificação das competências que serão oferecidas, por meio dos recursos. Cada competência pode possuir vários níveis ou graduações. Estas competências são divididas em atividades de assessoramento técnico, qualificação e certificação.

Etapa de identificação de recursos É a identificação dos recursos que estarão disponíveis durante todo o período, ou parte, em que as atividades deverão ser realizadas. Cada recurso identificado possui, pelo menos, uma competência. Cada competência do recurso receberá um nível de acordo com o grau de especialização do recurso. Os recursos pertencem ao orgão responsável pela execução das tarefas ou podem ser contratados temporariamente.

Etapa de identificação de tarefas É a identificação das tarefas que foram requisitadas e que deverão ser executadas durante o período de realização das tarefas, conforme solicitado. Cada tarefa identificada possui, pelo menos, uma competência.

Cada competência requerida da tarefa receberá um nível de acordo com o grau de especialização exigida. As tarefas são requisitadas por orgãos da Petrobras.

Etapa de alocação É a definição do relacionamento entre os recursos disponíveis e as tarefas requeridas em função das competências e dos seus níveis durante um determinado período de tempo.

Uma vez conhecido o objetivo e a identificação dos elementos essenciais para a alocação de equipes, da-se início ao processo de definição de todas as informações referentes a esses elementos. Estas informações são importantes para o processo de modo a estruturar a solução do problema. Normalmente, estas informações são previamente definidas, porém, há casos em que algumas informações necessitam ser calculadas, como por exemplo, tarefas que têm interseções entre si, distâncias entre os locais onde as tarefas serão realizadas, entre outras.

Por fim, para iniciar o processo de alocação de recursos, faz-se necessário a identificação das restrições do problema. Há dois tipos de restrições: as restrições que não podem ser desrespeitadas, chamadas de Fortes; e as que desejavelmente devem ser atendidas, chamadas de Fracas. Uma vez elencadas as restrições, o problema passa a ser a seleção ótima entre os elementos identificados de cada conjunto, definindo uma jornada de trabalho para cada recurso, de tal forma que todas as tarefas sejam atendidas, ou o número máximo possível. A seção a seguir descreve as restrições associadas ao nosso problema.

2.2 Restrições do problema de alocação de equipes

Neste trabalho, as restrições foram classificadas em dois grupos: fortes e fracas. As restrições fortes são aquelas que não podem ser violadas pela solução final do método, ou seja, devem necessariamente ser satisfeitas para que uma solução viável seja produzida. Por outro lado, as restrições fracas são aquelas que desejavelmente

não devem ser violadas pela solução final do método. Caso somente as restrições fracas sejam violadas, ainda assim o método produzirá uma solução viável, porém a sua violação deve ser evitada para que seja possível obter uma solução final com melhor qualidade. As restrições fracas são tratadas na própria função objetivo, conforme descrito na seção 3.6.

2.2.1 Restrições fortes

As seguintes restrições fortes são consideradas neste trabalho:

1. O número de recursos alocados na competência c requerida pela tarefa t terá que ser igual ao número de recursos solicitados para realizar tal competência c na tarefa t;

2. Um recurso r não será alocado em uma tarefa t se estes não forem compatíveis, isto é: se o recurso r não possuir a competência c; ou se o recurso r possuir nível na competência c inferior ao nível da competência c requerida pela tarefa t; ou se o recurso r possuir período de indisponibilidade coincidente com o período de realização da tarefa t;

3. Um recurso r não será alocado simultaneamente na tarefa t e na tarefa j caso elas tenham interseção entre si - duas tarefas tem interseção quando a execução de ambas coincide no tempo em pelo menos uma unidade de tempo;

4. Um recurso r somente poderá ser alocado para tarefas em paralelo que somem no máximo o valor da jornada de trabalho diária;

2.2.2 Função Objetivo

Os seguintes itens são otimizados, compondo a função objetivo juntamente com a penalização das restrições fracas.

1. Maximizar a soma dos pesos das tarefas atendidas;

2. Maximizar o número de recursos titulares alocados - titular é um quesito que diferencia o recurso em relação a uma determinada competência de tarefa de acordo com a sua experiência sobre tal competência;

3. Maximizar o número de recursos suplentes alocados - suplente é um quesito que diferencia o recurso em relação a uma determinada competência de tarefa de acordo com a sua experiência sobre tal competência;

4. Maximizar o número de recursos alocados em tarefas nas quais foram os últimos a executarem;

5. Maximizar o número de recursos alocados em tarefas nas quais já executaram;

6. Minimizar o *giveaway* (*giveaway* é a diferença entre o nível da competência do recurso alocado e o nível da competência da tarefa);

7. Minimizar o custo total de alocação dos recursos nas tarefas - cada recurso possui um custo associado;

8. Minimizar a distância percorrida pelos recursos. Consideramos como "distância percorrida"por um recurso como sendo a soma das distâncias entre as localidades das tarefas nas quais o recurso está alocado e das distâncias entre a localidade original do recurso e as localidades das tarefas nas quais o recurso está alocado. Veja a fórmula 3.16 definida no Modelo Matemático, Capítulo 3.

9. Minimizar o número de recursos alocados nas tarefas - um recurso r poderá estar alocado em mais de uma competência na mesma tarefa t;

Nota-se, pela descrição realizada ao longo deste Capítulo, que o problema de alocação de equipes possui vários objetivos. Esta característica multiobjetivo [6] é intrínseca ao problema. Sabe-se que em uma função $f : s \rightarrow \mathbb{R}$ possui-se uma relação de ordem, ou seja, dadas duas soluções quaisquer sabemos qual dentre as duas é de fato a melhor. Neste trabalho, optou-se por considerar uma única função objetivo que incorpora todos os critérios a serem analisados.

A área de Pesquisa Operacional da Petrobras desenvolveu um módulo de otimização utilizando Programação Matemática. Foram realizados vários testes considerando cenários reais e fictícios. Os resultados obtidos mostram que a solução do modelo proposto fornece soluções ótimas ou aproximadas em tempos aceitáveis, variando de poucos segundos a algumas horas conforme o tamanho do problema (diretamente relacionado ao número de técnicos, tarefas, locais de realização e das competências requeridas).

Uma função objetivo foi construída para avaliar a solução do problema, a qual deve ser maximizada. As expressões analíticas desta função objetivo e das restrições do problema, são devidamente apresentadas no próximo capítulo, no modelo matemático do problema.

Capítulo 3

Modelo Matemático

Esta seção apresenta o modelo matemático para o problema de alocação de equipes.

3.1 Índices

- r: Recurso;
- t: Tarefa;
- c: Competência;
- k: Localidade;
- i: Janela de indisponibilidade do recurso;
- u: União de tarefas.

3.2 Base de Dados

- $Custo_r$: custo do recurso r;
- $CompDisp_{r,c}$: Valor do nível da competência c do recurso r disponível;
- $CompReq_{t,c}$: Valor do nível da competência c da tarefa t requerida;
- $NRNC_{t,c}$: Número de recursos requeridos para a competência c da tarefa t;

- $Inicio_t$: Dia de início da tarefa t;
- Fim_t: Dia de fim da tarefa t;
- $LocalTarefa_t$: Identificador do local de realização da tarefa t;
- $LocalRecurso_r$: Identificador do local atual do recurso r;
- $Titular_{r,t,c}$: indica que o recurso r é titular da tarefa t na competência c;
- $Suplente_{r,t,c}$: indica que o recurso r é suplente da tarefa t na competência c;
- $Ultimo_{r,t}$: indica que o recurso r foi o último a executar a tarefa t;
- $Executou_{r,t}$: indica que o recurso r já executou a tarefa t;
- $PesoTarefa_t$: Valor da importância da tarefa t - quanto maior o valor, mais importante;
- $MatrizCompatibilidade_{r,t}$: indica que o recurso r é compatível com a tarefa t;
- $IntersecaoUniaoTarefas_{u,t}$: Conjunto ($u$) de tarefas ($t$) localizadas no mesmo local que podem ser executadas em paralelo pelo mesmo recurso. O total de tarefas que podem ser executadas em paralelo é controlado pela restrição 3.8;
- $JornadaTrabalho$: Jornada máxima de trabalho;
- $TempoExecucao_t$: Total de horas diárias necessárias para execução da tarefa t;
- $InicioIndisp_{r,i}$: Início da indisponibilidade do recurso r na janela i;
- $FimIndisp_{r,i}$: Fim da indisponibilidade do recurso r na janela i;
- $NumeroRecursos$: Total de recursos disponíveis;
- $NumeroTarefas$: Total de tarefas requeridas;
- $NumeroCompetencias$: Total de competências;

- $Intersecao_{t,j}$: Se a condição a seguir for satisfeita, retorna 1 indicando que a tarefa t tem interseção com a tarefa j, e 0 caso contrário;

$$\begin{aligned} ((Inicio_t \geq Inicio_j)\&(Inicio_t \leq Fim_j)) \quad & | \\ ((Fim_t + TempoTarefa_{t,j} \geq Inicio_j)\&(Fim_t \leq Fim_j)) \quad & | \\ ((Inicio_t \leq Inicio_j)\&(Fim_t \geq Fim_j)) \quad & \forall t, j \end{aligned}$$

- $DistanciaTarefa_{t,j}$: Distância normalizada entre o local da tarefa t e o local da tarefa j;

- $DistanciaRecursoTarefa_{r,t}$: Distância normalizada entre o local do recurso r e o local da tarefa j;

- NR_t: Número de recursos requeridos pela tarefa t;

$$NR_t = \max(NRNCr_{t,c}) \qquad \forall t$$

- $IndisponibilidadeAux_{r,i,t}$: indica que o recurso r está indisponível para executar a tarefa t na janela de indisponibilidade i;

$$\begin{aligned} ((InicioIndisp_{r,i} \geq Inicio_t)\&(InicioIndisp_{r,i} \leq Fim_t)) \quad & | \\ ((FimIndisp_{r,i} \geq Inicio_t)\&(FimIndisp_{r,i} \leq Fim_t)) \quad & | \\ ((InicioIndisp_{r,i} \leq Inicio_t)\&(FimIndisp_{r,i} \geq Fim_t)) \quad & \forall r, i, t \end{aligned}$$

- $Indisponibilidade_{r,t}$: Indisponibilidade do recurso r para executar a tarefa t;

$$Indisponibilidade_{r,t} = \sum_i IndisponibilidadeAux_{r,i,t} \qquad \forall r, t$$

3.3 Variáveis

- $x_{r,t,c}$: 1 indica que o recurso r foi alocado na tarefa t na competência c, e 0 caso contrário;

  Para todo índice tal que:

$$\begin{aligned} CompDisp_{r,c} \neq 0 \quad & \& \\ CompReq_{t,c} \neq 0 \quad & \& \\ CompDisp_{r,c} \geq CompReq_{t,c} \quad & \& \\ MatrizCompatibilidade_{r,t} = 1 \quad & \& \\ NRNCr_{t,c} \neq 0 & \end{aligned}$$

- $y_{r,t}$: 1 indica que o recurso r foi alocado na tarefa t, e 0 caso contrário;

  Para todo índice tal que:

$$MatrizCompatibilidade_{r,t} = 1$$

- w_t: 1 indica que a tarefa t foi atendida, e 0 caso contrário.

  Para todo índice tal que:

$$\sum_{r} MatrizCompatibilidade_{r,t} = 1$$

3.4 Restrições

A restrição 3.1 garante que para cada tarefa t haverá pelo menos um recurso r compatível.

$$\sum_{r} MatrizCompatibilidade_{r,t} \geq 1 \quad , \forall t \tag{3.1}$$

A restrição 3.2 garante que serão alocados $NRNCr$ recursos à competência c da tarefa t ou nenhum.

$$NRNCr_{t,c} \times w_t = \sum_{r} x_{r,t,c} \quad , \forall t, c \tag{3.2}$$

A restrição 3.3 garante que o recurso r será alocado na tarefa t somente se o recurso r for alocado a pelo menos uma competência c da tarefa t.

$$y_{r,t} \leq \sum_{c} x_{r,t,c} \quad , \forall r, t \tag{3.3}$$

A restrição 3.4 garante que o recurso r não será alocado simultaneamente na tarefa t e na tarefa j, caso elas tenham interseção entre si.

$$(y_{r,t} + y_{r,j}) \times Intersecao_{t,j} \leq 1 \quad , \forall r, t, j \tag{3.4}$$

A restrição 3.5 garante que serão alocados todos os $NRNCr$ recursos a todas as competências da tarefa t ou nenhum.

$$\sum_{c} NRNCr_{t,c} \times w_t = \sum_{r,c} x_{r,t,c} \quad , \forall t \tag{3.5}$$

A restrição 3.6 garante que serão alocados NR_t recursos à tarefa t ou nenhum.

$$NR_t \times w_t = \sum_{r} y_{r,t} \quad , \forall t \tag{3.6}$$

A restrição 3.7 garante que o recurso r será alocado na competência c da tarefa

t somente se o recurso r for alocado na tarefa t.

$$x_{r,t,c} \leq y_{r,t} \tag{3.7}$$

A restrição 3.8 garante que o recurso r será programado para tarefas em paralelo que somem no máximo o valor do parâmetro $JornadaTrabalho$.

$$\sum_{t} IntersecaoUniaoTarefas_{u,t} \times TempoExecucao_{t} \times y_{r,t} \leq JornadaTrabalho \qquad \forall r, u \tag{3.8}$$

3.5 Pesos da função objetivo

Os pesos da função objetivo correspondem aos valores associados às parcelas da função objetivo. Os valores dos pesos são números inteiros não negativos e podem variar de uma instância para outra, de acordo com a preferência ou necessidade do responsável pela alocação das equipes.

- $PesoGiveAWay$: Peso na composição da função objetivo do $GiveAway$ (diferença entre o nível de graduação do recurso alocado na competência e o nível de graduação da competência requerida);
- $PesoNumeroTarefasProg$: Peso na composição da função objetivo do número de tarefas atendidas;
- $PesoCustoTotal$: Peso na composição da função objetivo do custo total de alocação dos recursos nas tarefas;
- $PesoDistanciaPercorrida$: Peso na composição da função objetivo da distância percorrida pelos recursos;
- $PesoBonusTitular$: Peso na composição da função objetivo do bônus atribuído ao recurso quando este for alocado em uma competência de uma tarefa na qual é titular;

- *PesoBonusSuplente*: Peso na composição da função objetivo do bônus atribuído ao recurso quando este for alocado em uma competência de uma tarefa na é suplente;
- *PesoBonusUltimo*: Peso na composição da função objetivo do bônus atribuído ao recurso quando este for alocado em uma tarefa na qual foi o último a executá-la;
- *PesoBonusExecutou*: Peso na composição da função objetivo do bônus atribuído ao recurso quando este for alocado em uma tarefa na qual já a executou;
- *PesoNumeroAlocacoes*: Peso na composição da função objetivo do número total de recursos alocados às tarefas.

3.6 Função objetivo

$$
\begin{aligned}
maximizar \quad FO \quad = & \qquad (3.9)\\
PesoNumeroTarefasProgramadas \quad & \times \quad NumeroTarefasProgramadas\\
+ \quad PesoBonusTitular \quad & \times \quad BonusTitular\\
+ \quad PesoBonusSuplente \quad & \times \quad BonusSuplente\\
+ \quad PesoBonusUltimo \quad & \times \quad BonusUltimo\\
+ \quad PesoBonusExecutou \quad & \times \quad BonusExecutou\\
- \quad PesoGiveAWay \quad & \times \quad GiveAWay\\
- \quad PesoDistanciaPercorrida \quad & \times \quad DistanciaPercorrida\\
- \quad PesoCustoTotal \quad & \times \quad CustoTotal\\
- \quad PesoNumeroAlocacoes \quad & \times \quad NumeroAlocacoes
\end{aligned}
$$

onde:

$NumeroTarefasProgramadas$, definida em 3.10, quantifica os pesos das tarefas atendidas;

$$NumeroTarefasProgramadas = \sum_{t} PesoTarefa_t \times w_t \tag{3.10}$$

$BonusTitular$, definida em 3.11, quantifica os recursos alocados às competências das tarefas as quais são titulares;

$$BonusTitular = \sum_{r,t,c} Titular_{r,t,c} \tag{3.11}$$

$BonusSuplente$, definida em 3.12, quantifica os recursos alocados às competências das tarefas as quais são suplentes;

$$BonusSuplente = \sum_{r,t,c} Suplente_{r,t,c} \tag{3.12}$$

$BonusUltimo$, definida em 3.13, quantifica os recursos alocados às tarefas as quais foram os últimos a executá-las;

$$BonusUltimo = \sum_{r,t} Ultimo_{r,t} \tag{3.13}$$

$BonusExecutou$, definida em 3.14, quantifica os recursos alocados às tarefas as quais já executaram;

$$BonusExecutou = \sum_{r,t} Executou_{r,t} \tag{3.14}$$

$GiveAWay$, definida em 3.15, quantifca a diferença entre os níveis das competências dos recursos alocados e das tarefas atendidas;

$$GiveAWay = \sum_{r,t,c} (CompDisp_{r,c} - CompReq_{t,c}) \times x_{r,t,c} \tag{3.15}$$

$DistanciaPercorrida$, definida em 3.16, quantifica as distâncias entre as

localidades das tarefas atendidadas pelo recurso r e as distâncias entre a localidade original do recurso r e as localidades das tarefas atendidas pelo recurso r;

$$DistanciaPercorrida = \Big(\sum_{r,t,j} DistanciaTarefa_{t,j} \times y_{r,t} \times y_{r,j} + \sum_{r,t} DistanciaRecursoTarefa_{r,t} \times y_{r,t} \Big) \quad (3.16)$$

$CustoTotal$, definida em 3.17, quantifica o custo total dos recursos alocados nas tarefas;

$$CustoTotal = \sum_{r,t} Custo_r \times y_{r,t} \quad (3.17)$$

$NumeroAlocacoes$, definida em 3.18, quantifca a quantidade de recursos alocados nas tarefas.

$$NumeroAlocacoes = \sum_{r,t} y_{r,t} \quad (3.18)$$

Capítulo 4

Abordagem de Solução

Conforme introduzido ao longo do Capítulo 1, o problema de alocação de equipes, enfocado neste trabalho, apresenta um planejamento de grande complexidade. Esse problema surgiu a partir de uma demanda da área de engenharia da Petrobras, onde os seus responsáveis tinham a necessidade de gerar soluções com qualidade, em um tempo considerado rápido de acordo com a estrutura do problema e que despendesse pouco esforço humano para tal finalidade. A solução encontrada foi otimizar esta necessidade.

Inicialmente, houve a tentativa de desenvolver uma solução baseada apenas em análise de sistemas. Os resultados obtidos não foram satisfatórios. Perceberam, então, que para este tipo de desenvolvimento haveria necessidade de um desenvolvimento especializado. Com isso, solicitaram à Equipe de Pesquisa Operacional da Petrobras que desenvolvessem um módulo que contemplasse otimização para o problema de alocação de equipes em tarefas. Um módulo de otimização foi desenvolvido utilizando Otimização Combinatória.

Este trabalho baseia-se no desenvolvimento de um módulo de otimização utilizando Metaheurísticas.

O desenvolvimento de uma solução otimizada não é trivial. Ela requer o conhecimento e entendimento minucioso dos objetivos, requisitos e restrições, além de outros. Quanto mais profundo for este estudo inicial, melhores poderão ser os resultados obtidos. Além disso, faz-se necessário também entender o que se deseja

como solução, saber o tempo de resposta aceitável, qual o objetivo da solução e qual o grau de qualidade exigido e aceitável para a mesma.

Analisando o problema como um todo, vimos que ele é um problema singular, porém, para seu estudo e melhor compreensão, ele pode ser sub-dividido sob a forma de problemas menores os quais são bastante abordados na literatura, como por exemplo a programação de quadro de horários em escolas (*timetabling*).

O problema de programação de quadro de horários em escolas, também conhecido como programação de quadro de horário, ou *timetabling*, trata da alocação de professores em turmas de uma instituição educacional satisfazendo certas condições, dentre elas a quantidade de locais para aula, capacidade de alunos por sala, os horários disponíveis dos professores, entre outros.

Esta abordagem permitiu a possibilidade de estudar vários tipos de soluções e estruturas implementadas, possibilitando a aquisição de conhecimentos generalizados e específicos de cada tipo de problema e que serviu como diretriz para desenvolver uma solução para o problema como um todo. Outros trabalhos que apresentam procedimentos heurísticos para resolver o problema de *Timetabling* estão descritos em [1, 3–5, 7, 8, 29, 36–45, 47, 49, 52]

A solução para o problema de alocação de equipes se inicia com o planejamento da formação de um conjunto de recursos ou especialistas disponíveis, um conjunto de tarefas requeridas e um conjunto de competências. As competências são devidamente especificadas e elas correspondem às especializações requeridas nas tarefas e às especializações disponíveis nos recursos.

Em outras palavras, cada tarefa possui um conjunto de competências as quais necessitam ser realizadas, em contrapartida, cada recurso é especializado em uma ou mais competências a fim de realizar tais atividades demandadas. Todos os recursos, tarefas e competências são devidamente identificados.

4.1 Metaheurísticas baseadas em busca local

O procedimento de busca local, de modo geral, é uma busca que começa a partir de uma solução inicial s^0 e navega pelo espaço de soluções viáveis, através de um movimento m passando de uma solução à outra, de modo que esta seja sua vizinha, procurando assim melhorar o valor de uma função objetivo $f : s \in S \rightarrow \mathbb{R}$ a ser otimizada, e ao final da busca obtenha-se uma solução ótima local s. Define-se o conjunto de soluções vizinhas de s como:

$$\mathcal{N}(s) = \{s' : s' \leftarrow s \oplus m\}$$

Estruturas de vizinhança Geralmente é possível definir vários tipos distintos de estrutura de vizinhança, ou diversas famílias de movimentos, em um único problema. Vale ressaltar que quanto maior for a complexidade da transformação para gerar uma vizinhança, maior será o custo computacional para tratá-la.

O conjunto que define todas as vizinhanças de um dado problema, ou apenas aquelas com as quais se deseja trabalhar, pode ser representado sob a forma:

$$\mathcal{N}(s) = \left\{ \bigcup_{k=1}^{r} \mathcal{N}_{(s)}^{(k)} \right\},$$

onde r representa o número de vizinhanças distintas.

Estratégias de exploração de uma vizinhança Dentre um conjunto de opções, a heurística mais conhecida e mais sistemática é a chamada *BestImprovement* (BI) ou *MelhorVizinho*. Aplicar a heurística BI significa que, a partir de uma solução, analisa-se toda a vizinhança em busca do melhor vizinho. Para exemplificar, supondo um caso de maximização, teremos:

$$s'' = \arg \max_{s' \in \mathcal{N}(s)} \{f(s')\}$$

Porém, dependendo do problema a ser tratado, a busca pelo melhor vizinho pode tornar-se computacionalmente muito onerosa. Diante deste complicador, ob-

jetivando um processo menos custoso, pode-se empregar outra heurística, tal como:

First Improvement - realiza a busca pela primeira solução de melhora, ou seja, interrompe a busca local tão logo encontre uma solução melhor que a solução na atual iteração. Porém, esta heurística torna-se equivalente a BI quando a solução de melhora for a última solução buscada na vizinhança, ou se simplesmente não houver um vizinho de melhora;

Não há comprovação de qual das estratégias supracitadas comporta-se de forma mais adequada, pois a qualidade de tal escolha pode estar associada à estrutura do problema ou à qualidade da solução. Tendo isto em vista, recomenda-se experimentos iniciais.

Condição de parada A condição de parada geralmente é definida sob as seguintes formas:

- tempo de computação;
- número de iterações;
- alcance de uma solução tão boa quanto ou próxima de uma solução já conhecida, ou seja, um limitante (*bound*).

A maioria das técnicas metaheurísticas são técnicas relativamente recentes e merecem atenção, pois vem apresentando bons resultados ao serem aplicadas a problemas de otimização de grande complexidade.

Atualmente, estas técnicas são mais aplicadas em conjunto com outras metaheurísticas. Elas também podem auxiliar o método de programação matemática, trabalhando em conjunto. Nos casos em que a solução de um problema não necessite da otimalidade, as metaheurísticas podem agilizar o processo.

4.1.1 Greedy Randomized Adaptive Search Procedure (GRASP)

O GRASP é um processo multi-start e iterativo, no qual cada iteração consiste de duas fases, uma fase de construção, em que uma solução viável é produzida e uma fase de busca local, em que um ótimo local na vizinhança da solução construída é obtido. A melhor solução global é mantida como resultado. O Algoritmo 1 ilustra um procedimento GRASP de maximização no qual *maxitr* iterações GRASP são realizadas, onde: $f(.)$ é a função que desejamos maximizar; $g(.)$ é a função que determina o benefício de inserir cada elemento na fase de construção; *maxitr* é número máximo de iterações; x^* é a melhor solução retornada pelo algoritmo; α é o parâmetro que define o tamanho da Lista de Candidatos Restrita (LCR).

Algoritmo 1 Procedimento GRASP

requer $f(.)$, $g(.)$, $maxitr$, x^*
assegura solução viável x^*
  $x^* \leftarrow \emptyset$
  para $k = 1$ to $maxitr$ **faça**
    fase de construção($g(.)$, α, x)
    busca local($f(.)$, x)
    se $f(x) > f(x^*)$ **então**
      $x^* \leftarrow x$
    fim se
  fim para

Na fase de construção, uma solução é iterativamente construída, um elemento de cada vez. A fase de construção do GRASP básico é semelhante à heurística semi-gulosa proposta independentemente em [22]. Em cada iteração da Fase de Construção, a escolha do próximo elemento a ser adicionado à solução é determinada ordenando todos os elementos candidatos, isto é, aqueles que podem ser adicionados, em uma lista de candidatos C, em relação a uma função gulosa $g : C \rightarrow R$. Esta função determina o benefício de selecionar cada elemento. A heurística é adaptativa porque os benefícios associados a todos os elementos são atualizados na memória a cada repetição da fase de construção para refletir as mudanças ocorridas pela seleção do elemento anterior. O GRASP é randômico porque é escolhido um dos melhores

candidatos aleatoriamente da lista C, mas não necessariamente o melhor candidato. A lista de melhores candidatos é chamada de Lista de Candidatos Restrita (LCR). Esta técnica de escolha permite que soluções diferentes sejam obtidas a cada execução da Fase de Construção do GRASP. Considere $\alpha \in [0; 1]$ o parâmetro utilizado para definir a LCR. O Algoritmo 2 descreve a Fase de Construção do GRASP básico.

Algoritmo 2 Procedimento Fase de Construção

requer $g(.)$, α, x
assegura solução inicial viável
  $x \leftarrow \emptyset$
  inicializar a lista de candidatos C
  enquanto $C \neq \emptyset$ **faça**
    $s^- \leftarrow min\{g(t)|t \in C\}$
    $s^+ \leftarrow max\{g(t)|t \in C\}$
    $LCR \leftarrow \{s \in C|g(s) \leq s^- + \alpha(s^+ - s^-)\}$
    selecionar um $s \in LCR$ aleatoriamente
    $x \leftarrow x \cup \{s\}$
    atualizar a lista de candidatos C
  fim enquanto

A definição de LCR no Algoritmo 2 mostra que o parâmetro α controla o algoritmo quanto a sua característica semi-gulosa e randômica. Um valor $\alpha = 0$ corresponde a um procedimento guloso, enquanto $\alpha = 1$ corresponde a um procedimento com uma escolha totalmente aleatória.

Não são garantidas que as soluções geradas por uma construção do GRASP serão localmente ótimas com respeito a definições de vizinhanças. Conseqüentemente, quase sempre é benéfico aplicar uma busca local para tentar melhorar cada solução construída.

Um algoritmo de busca local trabalha de uma forma iterativa sucessivamente substituindo a solução atual por uma solução melhor na vizinhança da solução atual. Termina quando nenhuma solução melhor é achada na vizinhança.

A estrutura de vizinhança N para um problema P relaciona uma solução s do problema a um subconjunto de soluções $N(s)$. Uma solução s é dita localmente ótima se não há nenhuma solução melhor em $N(s)$. Uma escolha satisfatória de

uma estrutura de vizinhança proporcionará melhores resultados para um algoritmo de busca local.

Enquanto tais procedimentos de otimização local podem requerer tempo exponencial de um ponto de partida arbitrário, empiricamente a eficiência deles melhora de acordo com a solução inicial. De acordo com a estrutura de dados definida a implementação desenvolvida, uma eficiente fase de construção pode ser criada de forma a produzir soluções iniciais boas para uma eficiente busca local. O resultado é que frequentemente muitas soluções GRASP são geradas na mesma quantia de tempo requerida para o procedimento de busca local. Além disso, a melhor destas soluções GRASP geralmente é significativamente melhor que a única solução obtida de um ponto de partida aleatório. O Algoritmo 3 descreve um procedimento da Busca Local Básico.

Algoritmo 3 Procedimento Busca Local

requer $f(.)$, $N(.)$, x
assegura solução viável localmente ótima
  $H \leftarrow \{y \in N(x) | f(y) > f(x)\ \}$
  enquanto $|H| > 0$ **faça**
    selecionar $x \in H$
    $H \leftarrow \{y \in N(x) | f(y) > f(x)\ \}$
  fim enquanto

É difícil analisar a qualidade de valores das soluções usando somente a metodologia GRASP. Porém, há uma justificativa que vê o GRASP como uma técnica de amostragem repetitiva. Cada iteração do GRASP produz uma solução de amostra de uma distribuição desconhecida de todos os resultados alcançáveis. O valor da pior solução e a variância da distribuição são funções de natureza restritiva da lista de candidatos. Por exemplo, se a lista de candidatos restrita é limitada a um elemento, então só uma solução será produzida e a variância da distribuição será zero. Se um limite menos restritivo é imposto, muitas soluções diferentes serão produzidas implicando uma variância maior. Desde que a função gulosa é mais comprometida neste caso, o valor da pior solução pode degradar.

Uma característica especialmente atraente do GRASP é a facilidade com que

pode ser implementado. Poucos parâmetros precisam ser fixados e ajustados, e então o desenvolvimento pode focar em implementar estruturas de dados eficientes para assegurar iterações do GRASP rápidas.

Finalmente, o GRASP pode ser implementado trivialmente em paralelo. Cada processador pode ser inicializado com sua própria cópia do código, os dados da instância, e uma seqüência de números aleatórios independentes. As iterações do GRASP são então executadas em paralelo com somente uma única variável global exigida que servirá para armazenar a melhor solução achada sobre todos os processadores.

4.1.2 Variable Neighborhood Descend (VND)

O VND [21], uma variação do *Variable Neighborhood Search* (VNS) [20], é uma metaheurística que consiste basicamente na troca sistemática de vizinhanças associada a um algoritmo de busca local. O VND explora vizinhanças distantes da solução atual incrementalmente, e atualiza esta solução se e somente se uma nova solução melhor for produzida. Toda vez que há uma melhora da solução corrente, o método retorna à primeira estrutura de vizinhança. Desta maneira, características favoráveis para uma boa solução poderão ser mantidas e utilizadas para buscar uma boa solução nas vizinhanças.

As estruturas de vizinhança do VND devem ser previamente selecionadas e em número finito, diferentemente da maioria das heurísticas de busca local que utiliza apenas uma estrutura de vizinhança.

As definições das estruturas de vizinhança do VND podem propiciar uma maior qualidade da solução obtida, ou seja, vizinhanças bem definidas poderão promover bons resultados na fase de busca local.

Normalmente, a ordem de exploração das vizinhanças é das que possuem menor "distância"para as que possuem maior "distância"em relação ao ponto corrente.

Considere N_k, ($k = 1, 2, ..., k_{max}$) um conjunto finito de estruturas de vizinhanças previamente definidos e $N_k(x)$ o conjunto de vizinhanças definidas a partir de uma

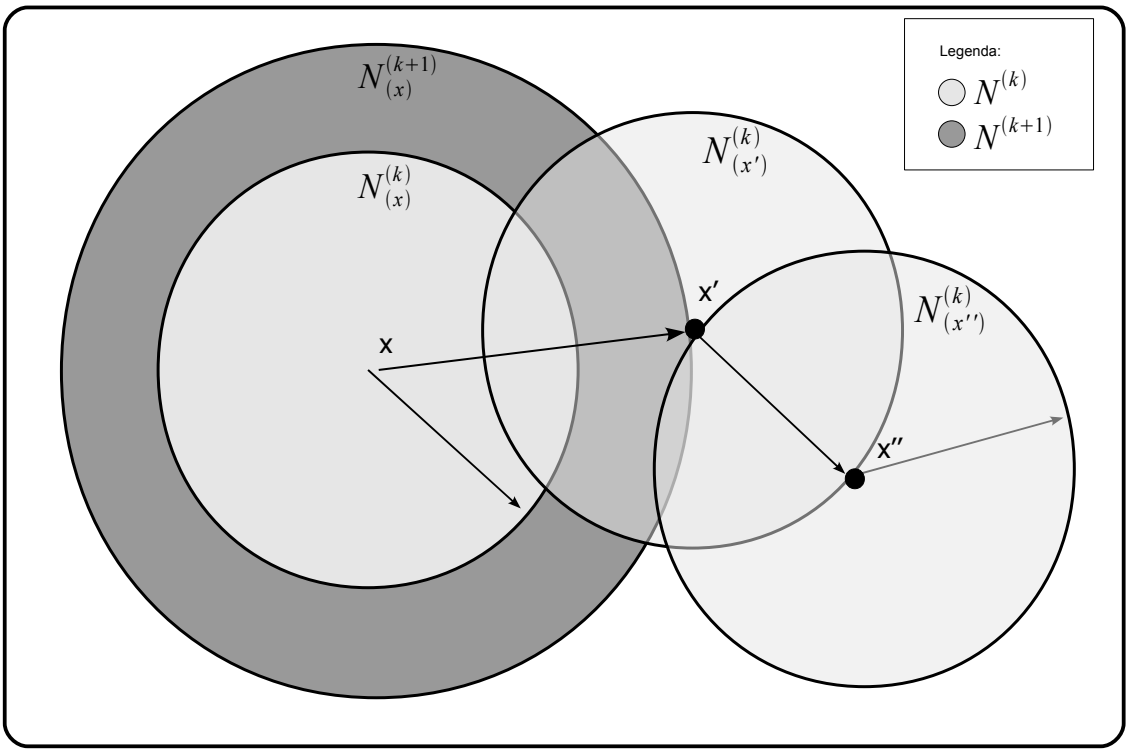

Figura 4.1: Comportamento do VND

solução inicial x. O Algoritmo 4 ilustra o procedimento VND básico.

O critério de parada é acionado quando, após percorrer todas as vizinhanças, não houver obtido nenhuma melhoria de solução.

A Figura 4.1 ilustra o comportamento do VND.

Algoritmo 4 Procedimento VND

requer $f(.)$, $N_k(x)$, x^0
assegura solução viável localmente ótima $x^* : f(x^*) \geq f(x^0)$
  $x^* \leftarrow x^0$
  $k \leftarrow 1$
  enquanto $k \leq k_{max}$ **faça**
    $x \leftarrow \{y \in N_k(x) | f(y) > f(x)\}$
    se $f(x) > f(x^*)$ **então**
      $x^* \leftarrow x$
      $k \leftarrow 1$
    senão
      $k \leftarrow k + 1$
    fim se
  fim enquanto

Capítulo 5

Métodos Propostos

Neste capítulo são apresentadas as metodologias empregadas para a criação do módulo de otimização do problema de alocação de equipes. Para que se tenha uma solução bem sucedida, é necessário que os objetivos e as restrições do problema tenham sido bem elucidadas.

Para este trabalho, por tratar de um problema semelhante ao problema de programação de quadro de horários, e pelo fato do problema de programação de quadro de horários ser um problema NP-Difícil [28] [13], optou-se por desenvolver um módulo de otimização utilizando uma metaheurística. A metaheurística escolhida foi Greedy Randomized Adaptive Search Procedure (GRASP) por ser de fácil programação e por apresentar bons resultados na literatura. Dentre os trabalhos que também apresentam a metaheurística GRASP para resolver outros tipos de problemas, podemos citar [2, 12, 25, 26, 31, 32, 51].

Para a fase de busca local optou-se por desenvolver um procedimento Variable Neighborhood Descent (VND) [20, 21].

Inicialmente, é gerada uma solução inicial, onde os recursos são alocados nas tarefas, obedecendo a um critério guloso, de acordo com as competências e níveis requeridos. A alocação é feita até que não seja mais possível alocar nenhum recurso em nenhuma outra tarefa, seja por motivo de incompatibilidade ou pelo fato de todas as tarefas já estarem alocadas.

A partir da solução inicial, é realizada uma busca local sobre as vizinhanças.

Após este procedimento de busca local, define-se a melhor solução local e um novo procedimento de refinamento é aplicado a esta solução. Esse melhor local é comparado com a melhor solução atual, e a melhor entre as duas soluções é mantida. Assim, uma nova iteração é reiniciada e todo o procedimento acima é executado, até que se atinja um número máximo de iterações pré-definido.

Durante a fase de busca local, no procedimento de definição de um novo vizinho, sempre que um recurso é desalocado de uma tarefa, é chamado um procedimento de realocação de recurso descrito na seção 5.7.1 na tentativa de aproveitar melhor o recurso em prol de uma melhor solução.

Um procedimento de alocação complementar descrito na seção 5.7.2 também é chamado na fase de busca local, após a definição de um novo vizinho, na tentativa de alocar recursos em tarefas incompletas, ou seja, tarefas com competências sem recurso alocado ou com número de recursos insuficiente.

A seguir, são definidas as fases do processo para o desenvolvimento da solução do módulo de otimização do problema de alocação de equipes.

Inicialmente, na seção 5.1, define-se a estrutura para a representação da solução do problema; na seção 5.2, verifica-se como os dados de entrada são apresentados e como são tratados; na seção 5.3, define-se como será gerada a solução inicial na fase de construção; e na seção 5.4, define-se as estruturas de vizinhança. Estas fases são comuns a todos as metodologias propostas neste trabalho. Em seguida, na seção 5.6 são descritos os procedimentos de busca local. Ao final deste capítulo, são apresentados dois procedimentos que são chamados na fase de busca local os quais tentam realocar recursos em tarefas. A condição de parada é a mesma para todos os métodos e está descrita na seção 5.5.

5.1 Representação do problema

A solução para o módulo de otimização do problema de alocação de equipes é representado por meio de uma matriz $X_{r \times t \times c}$ onde os seus elementos podem conter os valores -1, 0 ou 1. Cada elemento $x_{r,t,c} \in \{-1, 0, 1\}$ indica o status do recurso r

tarefas →	1				2				...				t			
competências →	1	2	..	c	1	2	..	c					1	2	..	c
recurso 1																
recurso 2																
...																
recurso r																

Figura 5.1: Representação da solução do problema

em relação à tarefa t e à competência c. Valor igual a 1 significa que o recurso r está alocado na competência c da tarefa t; valor igual a zero significa que o recurso r não está alocado na competência c da tarefa t; e valor igual a -1 significa que o recurso r não é compatível com a tarefa t, ou seja, o recurso r não possui a competência c ou possui a competência c com nível inferior ao requisitado pela competência c da tarefa t, ou o recurso r não está disponível no período de realização da tarefa t.

A Figura 5.1 representa a matriz solução sem valores.

5.2 Entrada de dados

Os dados de entrada do problema são informados pelos responsáveis da alocação de equipes através de um sistema de coleta de dados. Estes dados correspondem a informações sobre recursos, tarefas, competência requeridas e disponíveis, pesos utilizados na função objetivo, entre outras informações necessárias para definir a alocação das equipes.

Diante disto, realiza-se um pré-tratamento destas informações básicas a fim de gerar novas informações que serão repassadas ao módulo de otimização. Estas informações auxiliarão no preenchimento da matriz solução. Um exemplo deste processo é a geração dos valores das distâncias entre as localidades a partir das informações de latitude e longitude das localidades das tarefas e dos recursos. No Modelo Matemático, descrito no capítulo 3, na seção 3.2, Base de Dados, pode ser visto que dados são informados pelos usuários e a que tratamento eles são submetidos.

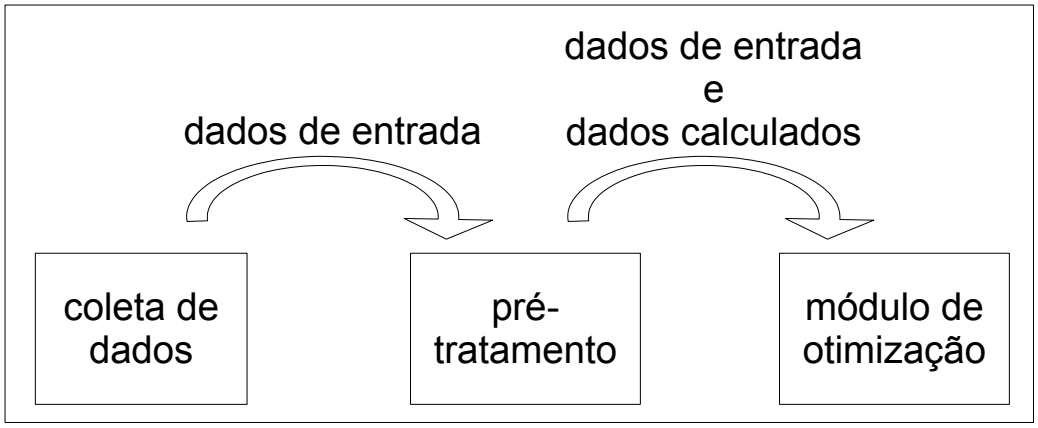

Figura 5.2: Percurso dos dados de entrada

A Figura 5.2 ilustra o percurso dos dados.

5.3 Fase de Construção

A solução inicial é gerada de forma construtiva na Fase de Construção da meta-heurística GRASP, descrito nas sub-seções a seguir.

5.3.1 Criando uma lista de tarefas e competências

Inicialmente, a partir das tarefas e competências requeridas, cria-se uma lista composta pelo par tarefa e competência candidatas que necessitam de recursos. Esta lista é ordenada, decrescentemente, de acordo com o valor calculado a partir de uma função de avaliação para cada elemento da lista. Esta função foi definida de forma empírica e é constituída de três parcelas.

A primeira parcela equivale ao nível da competência requerida e ainda não atendida; a segunda parcela calcula o total de recursos que podem atender à competência da tarefa; e a terceira parcela calcula o total de interseções que a tarefa requerida tem com as demais tarefas no atual momento.

Para se constituir uma ordem de importância entre as parcelas desta função de avaliação, cada parcela foi multiplicada por pesos distintos. A equação 5.1 ilustra a Função de Avaliação das Tarefas (F.A.T.).

$$
\begin{aligned}
F.A.T. &= Parcela_1 * Peso_1 \\
&+ Parcela_2 * Peso_2 \\
&+ Parcela_3
\end{aligned}
\tag{5.1}
$$

$$onde Peso_1 >> Peso_2 >> 1$$

As tarefas e competências que obtiverem os maiores valores na função de avaliação serão consideradas as mais críticas e terão prioridade na alocação dos recursos. Assim, seleciona-se o primeiro elemento da lista de tarefas e competências. Caso não seja possível alocar nenhum recurso nesta tarefa e competência selecionadas, seleciona-se o segundo elemento da lista de tarefas e competências, e assim por diante.

5.3.2 Criando uma lista de recursos

A partir dos recursos disponíveis, cria-se uma lista de recursos candidatos em função da tarefa e competência selecionadas. Esta lista é ordenada, decrescentemente, de acordo com outra função de avaliação. Esta função de avaliação, também definida de forma empírica, é constituída de 4 parcelas.

A primeira parcela equivale ao nível de competência que o recurso possui em relação ao par tarefa e competência selecionadas; a segunda parcela calcula o total de competências na tarefa a qual o recurso pode atender; a terceira parcela calcula o inverso do *giveaway* entre o recurso e as demais competências da tarefa; e a quarta parcela calcula o número de alocações do recurso em tarefas.

Para se constituir uma ordem de importância entre as parcelas desta função de avaliação, cada parcela foi multiplicada por pesos distintos. A equação 5.2 ilustra a Função de Avaliação dos Recursos (F.A.R.).

$$
\begin{aligned}
F.A.R. &= Parcela_1 * Peso_1 \\
&+ Parcela_2 * Peso_2 \\
&+ Parcela_3 * Peso_3 \\
&+ Parcela_4
\end{aligned}
\tag{5.2}
$$

$$onde Peso_1 >> Peso_2 >> Peso_3 >> 1$$

Para que um recurso faça parte desta lista de recursos candidatos, é necessário que o recurso seja compatível com a tarefa e competência selecionadas e não esteja alocado em alguma outra tarefa que tenha interseção com a tarefa selecionada. Caso não exista recurso para compor a lista de recursos candidatos, seleciona-se o próximo elemento da lista de tarefas e competências e tenta-se novamente criar uma lista de recursos candidatos em função da tarefa e competência selecionadas.

5.3.3 Criando uma lista de candidatos restrita

A partir da lista de recursos candidatos e de um parâmetro α, cria-se uma Lista de Candidatos Restrita (LCR), de tamanho $\mid LCR \mid$. A partir daí, inicia-se a tentativa de alocar um recurso na tarefa selecionada.

O valor de α escolhido foi 0,5. Por meio de experimentos, definiu-se este valor por apresentar melhores resultados. Os valores testados nos experimentos variaram de 0 a 1 com intervalo de 0,1 entre eles.

5.3.4 Alocando um recurso em uma tarefa e competência

Nesta parte, seleciona-se aleatoriamente um recurso da LCR. Aloca-se este recurso na tarefa e competência selecionadas e tenta-se alocar este recurso em outras competências da tarefa selecionada que necessitam de recurso e que sejam compatíveis.

Alocando-se o recurso na tarefa e competência, reinicia-se o processo conforme descrito a partir da seção 5.3.1. Senão, seleciona-se o próximo elemento da lista de

tarefas e competências e continua-se o procedimento a partir da seção 5.3.2.

O Algoritmo 5 ilustra esta Fase de Construção.

Algoritmo 5 Procedimento Fase de Construção da solução

requer lista de recursos R
requer lista de tarefas e competências TC
requer função de avaliação das tarefas e competências $g(.)$
requer função de avaliação dos recursos $h(.)$
requer parâmetro utilizado para definição da LCR α
requer x
assegura solução inicial viável
  $x \leftarrow \emptyset$
  $alocou \leftarrow$ **verdadeiro**
  enquanto $alocou$ **faça**
    $t, c \leftarrow min\{g(t,c)|t,c \in TC\}$
    $r^- \leftarrow min\{h(r)|r \in R\}$
    $r^+ \leftarrow max\{h(t)|r \in R\}$
    $LCR \leftarrow \{r \in R|h(r) \leq r^- + \alpha(r^+ - r^-)$
    $LCS \leftarrow \{r \in R|h(r) > r^- + \alpha(r^+ - r^-)$
    $alocou \leftarrow$ **falso**
    enquanto não $alocou$ **e** $LCR \neq \emptyset$ **faça**
      selecionar um $r \in LCR$ aleatoriamente
      $LCR \leftarrow LCR - \{r\}$
      se $x_{r,t,c} = 0$ **então**
        $x_{r,t,c} \leftarrow 1$
        $alocou \leftarrow$ **verdadeiro**
      fim se
    fim enquanto
    enquanto não $alocou$ **e** $LCS \neq \emptyset$ **faça**
      selecionar um $r \in LCS$
      $LCS \leftarrow LCS - \{r\}$
      se $x_{r,t,c} = 0$ **então**
        $x_{r,t,c} \leftarrow 1$
        $alocou \leftarrow$ **verdadeiro**
      fim se
    fim enquanto
  fim enquanto

Vale ressaltar que não é garantido que todas as tarefas serão atendidas.

Ao final desta fase, calcula-se o valor da função objetivo e o valor referente ao número de tarefas não atendidas para esta solução inicial, a fim de que sejam comparados com os respectivos valores dos seus vizinhos na fase de busca local. Neste momento, a solução inicial também é definida como sendo a melhor solução local.

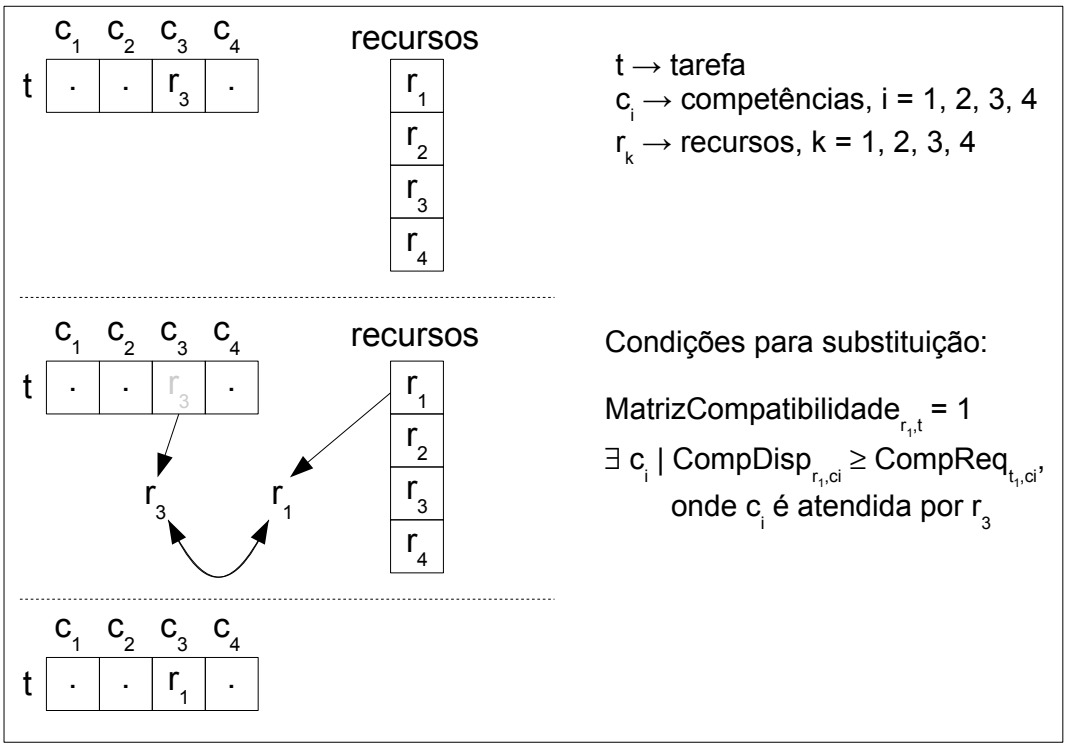

Figura 5.3: Movimento de substituição

5.4 Estruturas de vizinhança

Para este trabalho, definimos duas vizinhanças - uma contemplando um movimento de substituição e outra contemplando um movimento de troca.

A primeira vizinhança é definida pelo movimento de sustituição. Este movimento consiste na substituição de um recurso que está alocado a uma tarefa por outro recurso, desde que o recurso substituto possa atender a pelo menos uma das competências da tarefa na qual o recurso a ser substituído esteja alocado. Todos os movimentos de substituição possíveis são verificados.

A Figura 5.3 ilustra o movimento de substituição utilizado nesta vizinhança.

A segunda vizinhança é definda pelo movimento de troca. Este movimento consiste na troca entre dois recursos que estão alocados em duas tarefas distintas, desde que um recurso possa atender a pelo menos uma das competências da tarefa na qual o outro recurso esteja alocado, e vice-versa. Todos os movimentos de troca possíveis são verificados.

A Figura 5.4 ilustra o movimento de troca utilizado nesta vizinhança.

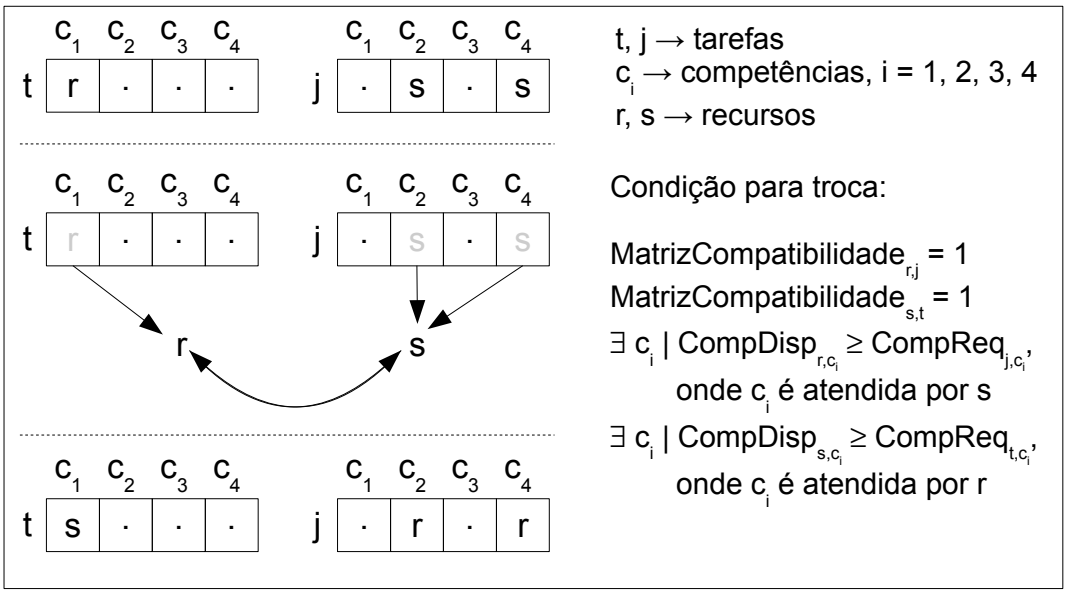

Figura 5.4: Movimento de troca

Vale enfatizar que, para ambas as vizinhanças, o recurso é desalocado de todas as competências da tarefa em que está alocado, até mesmo nos casos em que o recurso substituto não puder ser alocado em todas estas competências.

5.5 Condição de parada

A condição de parada utilizada neste trabalho é a mesma para todas as metodologias propostas. Cada execução dos métodos é limitada a 100 iterações, ou seja, serão geradas no máximo 100 (cem) soluções iniciais na fase de construção, e para cada solução inicial gerada será executada uma busca local e um outro refinamento.

Em cada iteração, na fase de busca local, o algoritmo finaliza tal iteração quando não ocorrer redução do números de tarefas não atendidas ou o número de tarefas não atendidas for zero. Assim, enquanto forem descobertas novas soluções no espaço delimitado pelas vizinhanças que consigam atender um número maior de tarefas em relação à solução anterior, o algoritmo continua sua execução na iteração.

5.6 Busca Local

Neste trabalho, utilizamos dois métodos de busca local: um método de busca local simples com uma única vizinhança, e o método *Variable Neighborhood Descent* (VND) com duas vizinhanças.

Empiricamente, por meio de experimentos realizados nas instâncias deste trabalho, decidiu-se utilizar a estratégia de exploração de vizinhança *First Improvement* por ser menos custosa e apresentar resultados tão bons quanto a estratégia *Best Improvement*.

Esta fase de Busca Local se inicia após concluída a fase Construção. A solução inicial gerada na fase de construção é repassada como parâmetro de entrada para os métodos de busca local, descritos nas sub-seções seguintes.

5.6.1 Busca Local Simples

Neste método é utilizada a vizinhança com movimento de substituição.

A partir da solução construída na Fase de Construção, que neste momento é a solução atual, realiza-se uma busca no espaço definido pela primeira vizinhança 5.4, por meio de movimento de substituição, a procura de um vizinho melhor do que a solução atual.

Por consequência do movimento de substituição, há uma preocupação em verificar se o vizinho poderá ser melhorado. Ou seja, nesta fase foram inseridos dois procedimentos suplementares para se verificar a possibilidade de alocar recursos em tarefas incompletas, melhorando assim o vizinho.

Em vista disto, após cada substituição de recursos, executa-se o Procedimento Suplementar Pós-Troca, conforme descrito na seção 5.7.1.

Além disso, após cada definição de um novo vizinho, algumas tarefas ainda podem permanecer incompletas. Então, para estes casos, executa-se o Procedimento Suplementar Para Tarefas Incompletas, conforme descritos na seção 5.7.2.

Após as tentativas de alocação dos procedimetos suplementares nas tarefas, calcula-se o valor referente ao número de tarefas não atendidas por este vizinho.

Agora, compara-se os valores referentes ao número de tarefas não atendidas entre o vizinho e a solução atual. Caso o vizinho possua um número menor de tarefas não atendidas em relação à solução atual, este vizinho passa a ser a nova solução atual e uma nova busca local é reiniciada. Caso contrário, continua-se a busca local corrente a partir da solução atual a fim de encontrar um novo vizinho.

Assim, este procedimento de busca local repete-se até todos os vizinhos da solução corrente terem sido visitados sem apresentar melhora com relação ao número de tarefas atendidas ou até que o número de tarefas não atendidas pela solução atual seja igual a zero.

5.6.2 Variable Neighborhood Descent

No método VND, além da primeira vizinhança utilizada na Busca Local Simples, usa-se também a segunda vizinhança definida pelo Movimento de Troca.

Inicialmente, executa-se toda o procedimento de Busca Local Simples descrito na seção anterior sobre a primeira vizinhança, definida pelo Movimento de Substituição. Após finalizado este procedimento pela ocorrência de não melhoria, reinicia-se o mesmo procedimento de Busca Local Simples utilizando a segunda vizinhança, definida pelo Movimento de Troca. Porém, nesta segunda vizinhança, caso o vizinho supere a solução atual, ou seja, possua um número menor de tarefas não atendidas em relação à solução atual, a solução atual passa a ser este vizinho e o VND é reiniciado, retornando a Busca Local Simple a partir da primeira vizinhança, definida pelo Movimento de Substituição. Caso o vizinho não supere a solução atual, o método continua a fim de encontrar um novo vizinho.

Este procedimento de busca local repete-se até todos os vizinhos terem sido visitados ou o número de tarefas não atendidas pela solução atual seja igual a zero.

O Algoritmo 6 ilustra esta fase de Busca Local baseada no método VND, onde $r(.)$ é uma função que calcula o número de tarefas não atendidas na solução.

Algoritmo 6 Procedimento Busca Local da solução

requer $r(.)$, $N_k(x)$, x^0
assegura solução viável localmente ótima $x^* : r(x^*) \leq r(x^0)$
  $x^* \leftarrow x^0$
  $k \leftarrow 1$
  enquanto $k \leq k_{max}$ **ou** $r(x^*) > 0$ **faça**
    $x \leftarrow \{y \in N_k(x) | r(y) < r(x)\}$
    procedimento pós troca(x)
    procedimento para tarefas incompletas(x)
    se $r(x) < r(x^*)$ **então**
      $x^* \leftarrow x$
      $k \leftarrow 1$
    senão
      $k \leftarrow k + 1$
    fim se
  fim enquanto

5.7 Procedimentos Suplementares

Estes procedimentos suplementares são uma extensão dos procedimentos de busca local descritos na seção 5.6.

Como visto na seção 5.6, alguns procedimento suplementares são executados visando completar as tarefas incompletas, ou seja, tarefas com competência sem recurso ou com número insuficiente de recursos alocados, e assim gerar soluções melhores. Estes procedimentos, assim como na Fase de Construção, não garantem que todas as tarefas serão completadas.

As sub-seções seguintes explicam os procedimentos suplementares adotados.

5.7.1 Procedimento Pós-Troca

Na fase de busca local, após as ações de substituição e troca de um recurso r em uma tarefa t e competência c, pode ser que seja possível alocar o recurso r em uma tarefa j na qual tem interseção com a tarefa t. Por este motivo, verifica se a tarefa j tem esta característica e em caso positivo tenta-se alocar o recurso r nas competências da tarefa j.

5.7.2 Procedimento Para Tarefas Incompletas

Na fase de busca local, após o Procedimento de Pós-Troca, é verificada a possibilidade de alocar recursos nestas tarefas incompletas, utilizando os mesmos critérios da fase de contrução. Cria-se uma lista formada por pares tarefas e competências incompletas e a partir da tarefa e competência selecionadas, cria-se uma lista de recursos e repete-se o mecanismo de alocação da Fase de Construção, conforme descrito na seção 5.3.4.

5.8 Outro Refinamento

Após terminada a execução da fase de busca local de cada iteração, se o número de tarefas não atendidas da solução atual for igual a zero, a solução atual recebe outro tratamento de refinamento. Nesta fase, a solução atual passará novamente pelo método VND. Porém, a comparação entre a solução atual e seus vizinhos será por meio dos valores das funções objetivo referentes a tais soluções, e não por meio dos valores referentes ao número de tarefas não atendidas como acontece na Fase de Busca Local. Assim, ao final deste procedimento, a solução atual será a melhor solução obtida nesta execução.

O Algoritmo 7 ilustra a chamada do outro refinamento, onde $r(.)$ é uma função que calcula o número de tarefas não atendidas na solução.

Algoritmo 7 Algoritmo GRASP + VND da solução

requer $maxitr$, x^*
assegura solução viável x^*
  $x^* \leftarrow \emptyset$
  $k \leftarrow 1$
  enquanto $k \leq maxitr$ **faça**
    fase de construção(x^*)
    busca local(x^*)
    se $r(x^*) = 0$ **então**
      refinamento(x^*)
    fim se
    $x^* \leftarrow x^*$
    $k \leftarrow k + 1$
  fim enquanto

Capítulo 6

Resultados e Discussão

Os dois métodos propostos neste trabalho foram desenvolvidos em linguagem Java, através do Ambiente de Desenvolvimento Integrado Netbeans 6.8. A versão do compilador Java utilizado foi a jdk1.6.0_18. Para a resolução do problema por meio de Programação Matemática foi utilizado o pacote comercial CPLEX 11.2. A maioria dos experimentos mencionados nesse trabalho foram realizados em um microcomputador com processador AMD Athlon(tm) 64 X2 Dual Core 4800, 2,50 GHz, 2 GB de memória RAM e sistema operacional Windows XP. As exceções foram os experimentos realizados nas instâncias 16 a 20, resolvidos por Programação Matemática, que foram processados em um microcomputador com processador Intel Core 2 Duo 7400, 2,8 GHz, 3,23 GB de memória RAM e sistema operacional Windows XP, porque o microcomputador teve que ser trocado.

Vale ressaltar que as metodologias implementadas neste trabalho utilizam a estratégia de exploração de vizinhança First Improvement (FI), pois esta estratégia apresentou melhores resultados em experimentos preliminares. Estes resultados foram comparados com outros resultados obtidos utilizando a estratégia Best Improvement (BI), aplicados nas mesmas instâncias, e observou-se uma convergência das soluções mais rápida.

Em relação ao parâmetro α utilizado na Fase de Construção do GRASP, foi utilizado o valor de 0,5. Verificou-se, empiricamente, por meio de experimentos preliminares que este valor fornecia melhores resultados.

6.1 Instâncias teste

Considera-se um conjunto de 20 (vinte) instâncias de teste, as quais foram geradas com a finalidade de serem utilzadas somente para testes. Pelo fato de o método de Programação Matemática não estar em produção na Petrobras, os dados utilizados para testes não são reais, porém, pode-se considerar que tais dados possuem valores bem próximos da realidade, uma vez que foram gerados e compilados pelo pessoal da área de engenharia da Petrobras, o qual solicitou a solução do problema.

As principais propriedades de tais instâncias encontram-se dispostas através da Tabela 6.1. A primeira coluna é uma identificação numérica sequencial para as instâncias; a segunda coluna corresponde ao nome dado à cada instância; e as demais colunas informam a quantidade de recursos disponíveis, a quantidade de tarefas requisitadas e a quantidade de competências, nesta ordem, para cada instância.

A Tabela 6.2 apresenta o resumo de desempenho da Programação Matemática, por meio dos valores das funções objetivo e tempos de processamentos da solução ótima para cada instância teste. Além disso, informa o número de variáveis contidas em cada uma das instâncias teste. Estes valores servirão de parâmetro de comparação para os resultados dos experimentos dos métodos propostos, a fim de se obter uma melhor análise e referência em relação à qualidade das soluções produzidas pelos métodos propostos.

Vale ressaltar que as instâncias teste mostradas na Figura 6.4 as quais obtiveram soluções de metaheurísticas com gap superior a 5%, são instâncias que foram criadas visando reduzir ao máximo o espaço solução, ou seja, representam problema de mais difícil solução.

6.2 Desempenho das metodologias

Para cada par instância/metodologia, foram realizados 10 (dez) experimentos.

A Tabela 6.3 apresenta os detalhes dos resultados dos experimentos e desempenho dos métodos propostos realizados nas instâncias teste. As colunas desta

	instância	recursos	tarefas	competências
1	R30T50C10a	30	50	10
2	R30T50C10b	30	50	10
3	R30T50C10c	30	50	10
4	R30T50C10d	30	50	10
5	R30T50C10e	30	50	10
6	R30T50C10f	30	50	10
7	R30T50C10g	30	50	10
8	R30T50C10h	30	50	10
9	R30T50C10i	30	50	10
10	R30T50C10j	30	50	10
11	R50T100C10a	50	100	10
12	R50T100C10b	50	100	10
13	R50T100C10c	50	100	10
14	R50T100C10d	50	100	10
15	R50T100C10e	50	100	10
16	R50T100C10f	50	100	10
17	R50T100C10g	50	100	10
18	R50T100C10h	50	100	10
19	R50T100C10i	50	100	10
20	R50T100C10j	50	100	10
21	R100T200C10a	100	200	10
22	R100T200C10b	100	200	10
23	R100T200C10c	100	200	10
24	R100T200C10d	100	200	10
25	R100T200C10e	100	200	10
26	R100T200C10f	100	200	10
27	R100T200C10g	100	200	10
28	R100T200C10h	100	200	10
29	R100T200C10i	100	200	10
30	R100T200C10j	100	200	10

Tabela 6.1: Propriedades das intâncias teste

instância	z^*	tempo (s)	variáveis	gap (%)
1	24.883,860	4,75	4.147	0,00
2	24.921,034	4,28	5.139	0,00
3	24.912,341	9,81	5.162	0,00
4	24.926,699	16,22	6.700	0,00
5	24.938,926	11,88	6.518	0,00
6	24.998,641	0,81	4.884	0,05
7	24.940,112	1,14	5.088	0,05
8	24.996,940	3,95	4.884	0,05
9	24.990,901	3,91	4.840	0,03
10	24.916,399	0,88	6.061	0,04
11	48.601,729	1.825,81	37.599	0,02
12	50.597,212	1.834,14	62.436	0,06
13	50.597,212	1.896,19	62.436	0,06
14	50.599,097	1.556,34	62.473	0,05
15	50.614,983	2.031,86	62.165	0,06
16	50.606,069	570,89	62.566	0,05
17	50.606,243	735,88	60.153	0,05
18	50.642,246	494,47	57.890	0,05
19	51.165,184	161,80	53.643	0,05
20	51.205,426	157,92	53.431	0,05
21	84.017,990	1.806,84	162.621	9,37
22	92.649,100	1.809,00	387.862	0,28
23	92.880,382	1.826,02	387.862	0,26
24	93.562,709	1.806,02	392.050	0,26
25	92.893,277	1.810,91	392.919	0,24
26	93.617,012	1.804,06	387.128	0,13
27	92.956,968	1.810,80	392.968	0,32
28	92.884,641	1.804,13	392.680	0,25
29	92.889,199	1.809,61	387.623	0,23
30	93.018,308	1.807,70	387.906	0,27

Tabela 6.2: Propriedades das soluções ótimas para as intâncias teste, obtidas por Programação Matemática

tabela, da esquerda para a direita, representam as seguintes informações, nesta ordem: identificação da instância; sigla da metodologia; média aritmética dos valores das funções objetivo de todas as execuções; valor da função objetivo da melhor solução; desvio entre a média aritmética dos valores das funções objetivo de todas as execuções e o valor da função objetivo da melhor solução; desvio entre a média aritmética dos valores das funções objetivo de todas as execuções e o valor da função objetivo da solução ótima; média aritmética dos tempos de processamento de todas as execuções, em segundos; média aritmética dos tempos de processamento para se alcançar um valor sub-ótimo da função objetivo que se encontre em uma distância (desvio) de 5% em relação a solução ótima, em segundos.

Os valores de z^* informados na primeira coluna, entre parênteses, correspondem aos valores das funções objetivos das soluções ótimas informados Tabela 6.2, novamente dispostos apenas para facilitar a comparação.

Os valores percentuais de desvio informados na Tabela 6.3, $desvio(\%)melhor$ e $desvio(\%)z^*$, tem o objetivo de mensurar o quanto de desvio existe entre valor médio e valor da melhor solução e valor médio e valor da solução ótima, respectivamente. Para se obter estes valores, emprega-se a fórmula 6.1.

$$\sigma(z, z') = \frac{z - z'}{z'} \times 100 \tag{6.1}$$

As informações contidas na última coluna da Tabela 6.3 é relevante para se analisar o comportamento das metodologias propostas quando se deseja alcançar uma solução de mínima qualidade.

Observa-se que, para a maioria das instâncias teste, as metodologias propostas apresentam tempos bastante curtos e soluções a uma distância inferior a 5% em relação a solução ótima.

Nas instâncias 1, 11 e 21 temos os maiores gaps em relação a solução ótima pelo fato das metaheurísticas atenderem um número menor de tarefas em relação a Programação Matemática. Inclusive, este fato implica o gap das instâncias 1, 11 e 21 ficarem com valores superiores a 5% em relação à solução ótima. Porém, como

mencionado anteriormente, este problema é resolvido na Petrobras em um segundo módulo ou uma segunda aplicação da metaheurística, mas que não é tratado neste trabalho.

A Tabela 6.4 apresenta dados estatísticos extraídos dos experimentos realizados, com exceção das instâncias 1, 11 e 21. As colunas desta tabela, da esquerda para a direita, representam as seguintes informações, nesta ordem: distância média (gap médio) e a maior distância (gap máximo) em relação à solução ótima; desvio médio (desvio médio) e maior desvio (desvio máximo) em relação à melhor solução produzida por cada método proposto.

A metaheurística GRASP+VND se destaca em relação à GRASP Básica, pois apresentou melhores resultados, produziu, em média, soluções a uma distância inferior, alcançou melhores soluções na maioria dos experimentos e, finalmente, por sua robustez observada através dos baixos desvios.

6.3 Gráfico das Evoluções

As Figuras 6.1 e 6.2 apresentam, graficamente, a evolução das melhores soluções produzidas para cada par instância/metodologia das instâncias de tamanho reduzido e normal, respectivamente, durante a execução do experimento. Na legenda de cada um dos gráficos encontram-se a sigla das metodologias propostas.

A análise das figuras citadas acima mostra claramente a proximidade das soluções propostas em relação à solução ótima. Outro quesito que pode ser verificado na análise destas figuras é a rapidez com que as metodologias propostas convergem em direção a valores muito próximos aos das soluções ótimas.

A Figura 6.4 apresenta as instâncias que tiveram gaps maiores que 5% em relação a solução ótima. Isto ocorreu pelo fato das metaheurísticas atenderem um número menor de tarefas em relação a Programação Matemética, conforme mencionado anteriormente. Porém, como mencionado anteriormente, este problema é resolvido na Petrobras em um segundo módulo ou uma segunda aplicação da metaheurística, mas que não é tratado neste trabalho.

instância (z^*)	método	z		desvio (%)		tempo médio (s)	
		média	melhor	melhor	z^*	parada	sub-ótimo
1	GRASP+VND	23.398,998	23.493,926	0,404	5,967	23,500	-
(24.883,860)	GRASP	23.293,592	23.423,639	0,555	6,391	11,600	-
2	GRASP+VND	24.743,913	24.775,473	0,127	0,711	21,300	2,300
(24.921,034)	GRASP	24.715,677	24.744,031	0,115	0,824	14,000	1,100
3	GRASP+VND	24.749,233	24.766,266	0,069	0,655	16,400	2,300
(24.912,341)	GRASP	24.709,551	24.722,104	0,051	0,814	8,000	0,900
4	GRASP+VND	24.814,388	24.860,250	0,184	0,451	49,400	7,600
(24.926,699)	GRASP	24.769,281	24.782,658	0,054	0,632	23,400	3,500
5	GRASP+VND	24.820,484	24.846,816	0,106	0,475	45,300	8,400
(24.938,926)	GRASP	24.787,103	24.813,603	0,107	0,609	19,100	2,600
6	GRASP+VND	24.749,361	24.905,772	0,628	0,997	48,400	2,200
(24.998,641)	GRASP	24.675,781	24.853,035	0,713	1,292	17,400	1,800
7	GRASP+VND	24.772,789	24.840,488	0,273	0,671	34,900	1,500
(24.940,112)	GRASP	24.793,192	24.853,850	0,244	0,589	15,700	1,300
8	GRASP+VND	24.772,789	24.840,488	0,273	0,897	34,900	2,400
(24.996,940)	GRASP	24.823,484	24.850,880	0,110	0,694	23,900	0,700
9	GRASP+VND	24.622,198	24.847,681	0,907	1,475	40,700	2,200
(24.990,901)	GRASP	24.533,565	24.829,261	1,191	1,830	19,700	1,700
10	GRASP+VND	24.808,513	24.834,048	0,103	0,433	33,900	2,600
(24.916,399)	GRASP	24.772,370	24.796,414	0,097	0,578	10,200	2,400
11	GRASP+VND	29.891,117	30.694,945	2,619	38,498	1.207,700	-
(48.601,729)	GRASP	29.979,777	30.721,150	2,413	38,315	413,100	-
12	GRASP+VND	49.694,507	49.776,141	0,164	1,784	337,500	56,100
(50.597,212)	GRASP	49.669,437	49.737,839	0,138	1,834	53,080	23,700
13	GRASP+VND	49.675,004	49.767,425	0,186	1,823	162,100	46,800
(50.597,212)	GRASP	49.627,002	49.665,851	0,078	1,918	95,300	15,100
14	GRASP+VND	49.710,829	49.790,122	0,159	1,756	203,600	44,700
(50.599,097)	GRASP	49.640,840	49.775,366	0,270	1,894	56,600	17,600
15	GRASP+VND	49.660,755	49.708,158	0,095	1,885	192,800	52,500
(50.614,983)	GRASP	49.647,694	49.729,082	0,164	1,911	91,900	20,700
16	GRASP+VND	49.683,552	49.729,843	0,093	1,823	213,500	60,200
(50.606,069)	GRASP	49.628,591	49.705,375	0,154	1,932	57,400	8,600
17	GRASP+VND	49.638,063	49.672,947	0,070	1,913	218,900	36,900
(50.606,243)	GRASP	49.502,004	49.627,501	0,253	2,182	40,900	10,400
18	GRASP+VND	49.643,201	49.723,405	0,161	1,973	229,100	21,100
(50.642,246)	GRASP	49.389,248	49.692,930	0,611	2,474	26,600	13,700
19	GRASP+VND	50.208,418	50.275,902	0,134	1,870	229,900	50,400
(51.165,184)	GRASP	50.186,193	50.289,678	0,206	1,913	37,200	14,900
20	GRASP+VND	50.372,822	50.455,769	0,164	1,626	960,300	44,600
(51.205,426)	GRASP	49.792,092	50.368,927	1,145	2,760	32,400	18,600
21	GRASP+VND	26.400,725	28.337,500	6,835	68,577	2.064,100	-
(84.017,990)	GRASP	23.624,675	24.761,500	4,591	71,881	259,800	-
22	GRASP+VND	91.271,939	91.456,549	0,202	1,486	1.504,500	418,800
(92.649,100)	GRASP	90.908,723	91.029,755	0,133	1,878	684,800	186,500
23	GRASP+VND	91.271,939	91.456,549	0,202	1,732	1.504,500	418,800
(92.880,382)	GRASP	91.250,954	91.336,546	0,094	1,754	451,000	182,000
24	GRASP+VND	91.962,922	92.190,614	0,247	1,710	2.990,900	732,300
(93.562,709)	GRASP	91.938,398	92.082,237	0,156	1,736	829,900	293,100
25	GRASP+VND	91.277,203	91.409,514	0,145	1,740	1.418,600	529,200
(92.893,277)	GRASP	91.244,903	91.331,950	0,095	1,774	706,800	200,800
26	GRASP+VND	92.162,524	92.321,844	0,173	1,554	2.045,400	797,800
(93.617,012)	GRASP	92.119,330	92.203,851	0,092	1,600	672,600	287,400
27	GRASP+VND	91.342,591	91.466,981	0,136	1,737	1.237,600	468,800
(92.956,968)	GRASP	91.333,854	91.392,605	0,064	1,746	499,800	200,200
28	GRASP+VND	91.483,251	91.624,224	0,154	1,509	1.049,000	416,000
(92.884,641)	GRASP	91.472,719	91.571,890	0,108	1,520	517,600	149,800
29	GRASP+VND	91.084,826	91.213,933	0,142	1,943	781,200	337,400
(92.889,199)	GRASP	91.125,146	91.368,732	0,267	1,899	347,200	145,000
30	GRASP+VND	91.276,082	91.343,009	0,073	1,873	1.644,800	497,600
(93.018,308)	GRASP	91.333,423	91.414,043	0,088	1,811	612,000	226,400

Tabela 6.3: Resultados e desempenho das metodologias x instâncias

método	gap médio	desvio médio
GRASP+VND	1,43	0,20
GRASP	1,57	0,25

Tabela 6.4: Resumo de desempenho sem as instâncias 1, 11 e 21

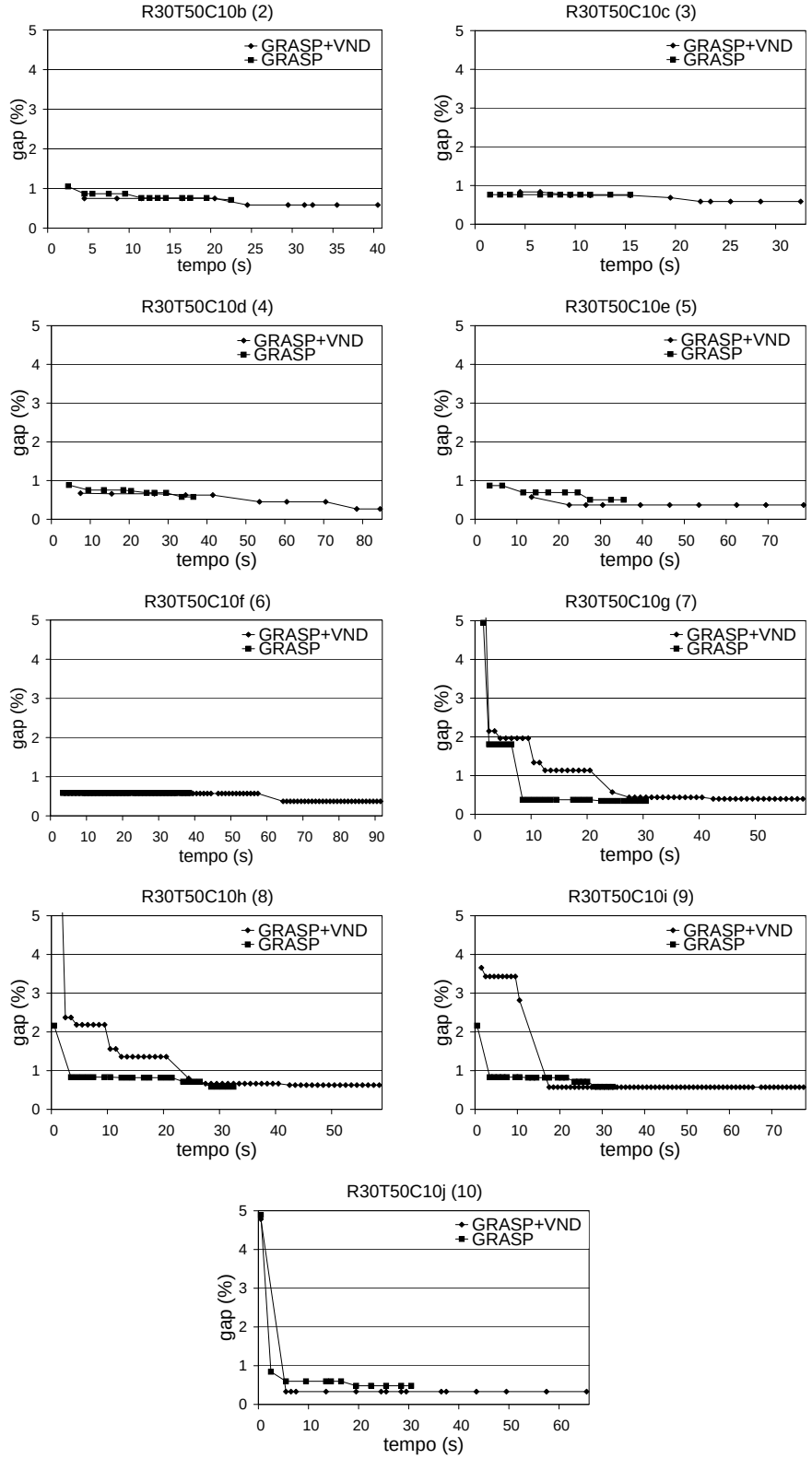

Figura 6.1: Evoluções dos experimentos de melhor solução em cada instância/metodologia das instâncias de tamanho reduzido com gap inferior a 5%

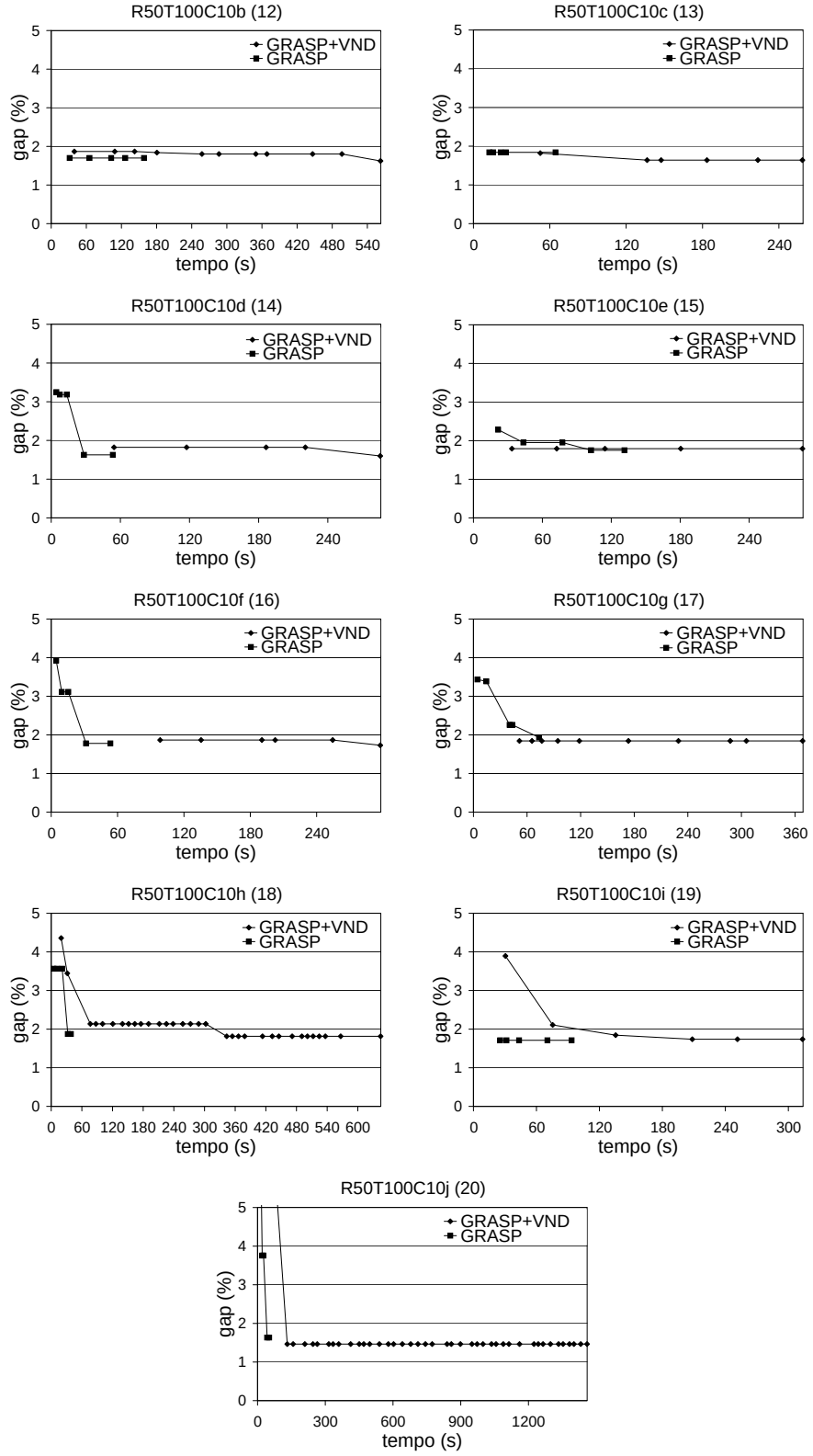

Figura 6.2: Evoluções dos experimentos de melhor solução em cada instância/metodologia das instâncias de tamanho normal com gap inferior a 5%

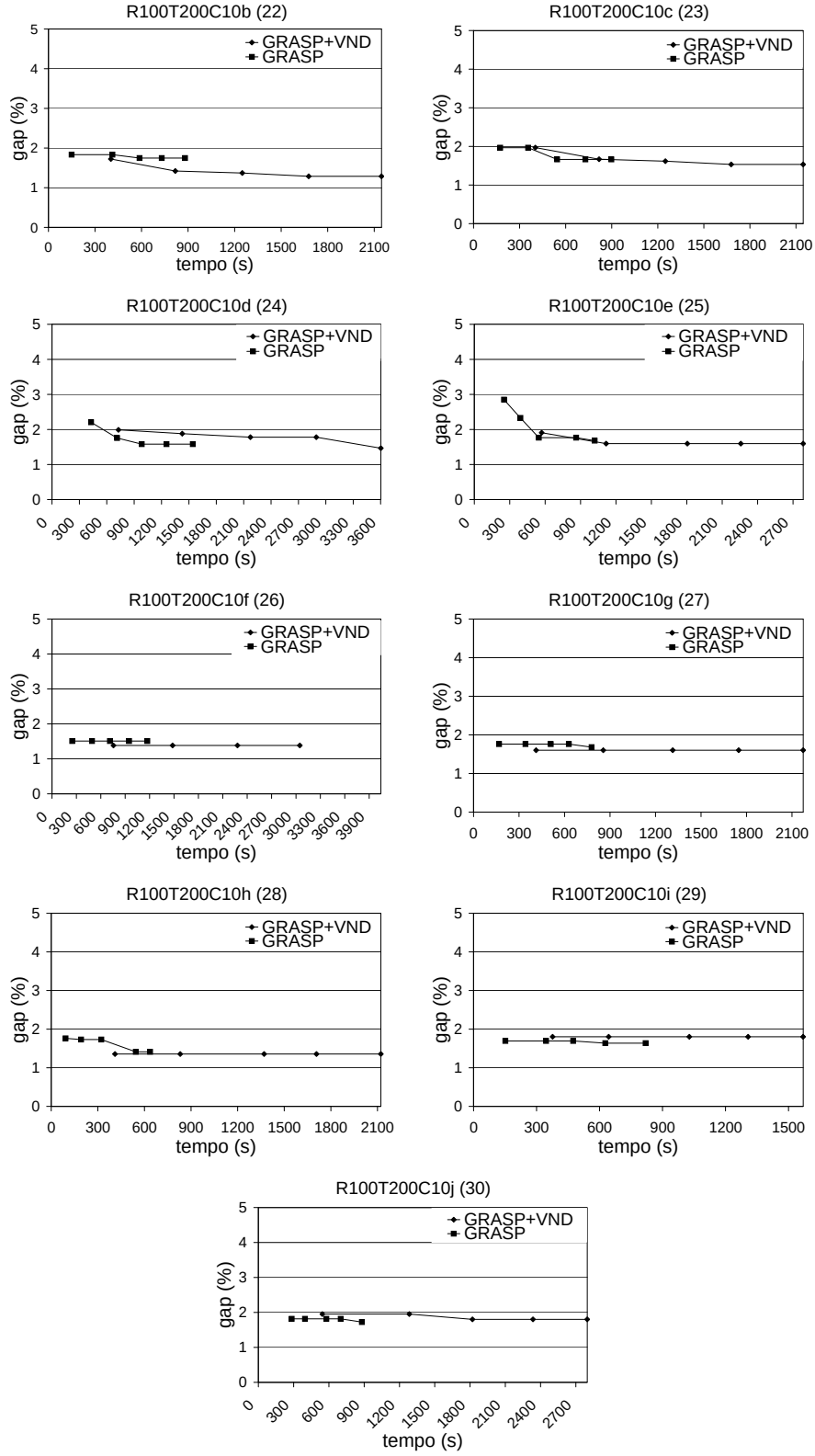

Figura 6.3: Evoluções dos experimentos de melhor solução em cada instância/metodologia das instâncias grandes com gap inferior a 5%

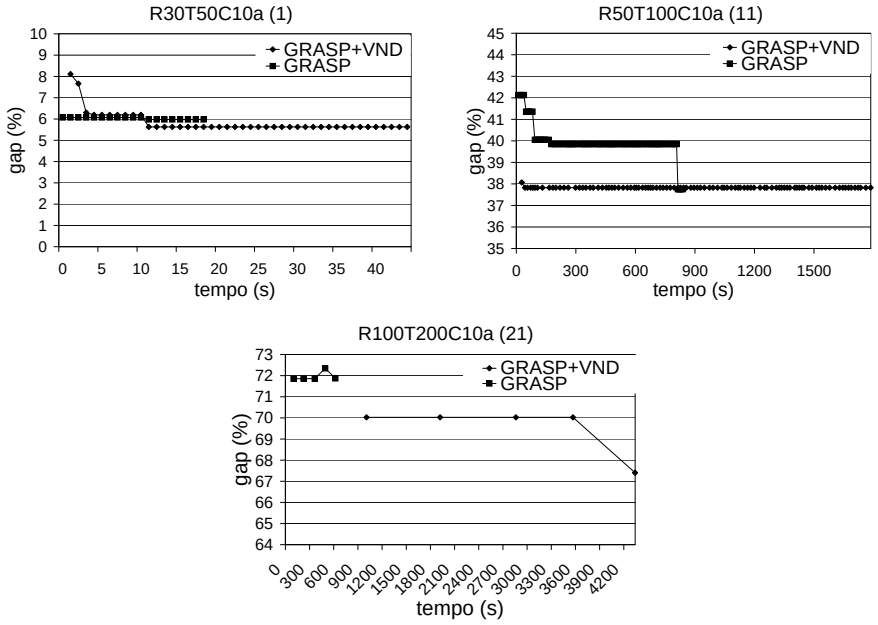

Figura 6.4: Evoluções dos experimentos de melhor solução em cada instância/metodologia das instâncias com gap superior a 5%

Capítulo 7

Conclusões e Trabalhos Futuros

A solução de problemas semelhantes ao de alocação de equipes, e de um modo geral aos de programação de quadros de horários, é muito importante para que as instituições não desperdicem tempo realizando estes tipos de programação manualmente. Porém, há uma grande necessidade de trabalhos como este que auxiliem na difusão de técnicas como as que foram propostas e desenvolvidas neste trabalho com o propósito de resolução para este tipo de problema.

Neste trabalho, para a resolução do Problema de Alocação de Equipes, foram propostas e implementadas duas metaheurística: um algoritmo GRASP em conjunto com o algoritmo VND e um algoritmo GRASP simples, para efeito de comparação. A escolha destas técnicas se justifica pelo fato de haver um histórico de destaque e sucesso em abordagens de problemas de natureza combinatória e de difícil resolução, pelo baixo número de parâmetros e pela sua boa robustez quando comparadas com outras metaheurísticas presentes na literatura. Outro fator relevante, é a simplicidade para implementar os algoritmos.

Com o propósito de se mensurar de forma mais apurada a qualidade das soluções produzidas pelas metodologias propostas, aproveitou-se a implementação da Programação Matemática desenvolvida pelo pessoal de Pesquisa Operacional da Petrobras para o Problema de Alocação de Equipes. Assim, soluções ótimas foram geradas para as instâncias teste utilizando a Programação Matemática.

Concluiu-se, de fato, por meio de uma comparação entre os resultados prove-

nientes das implementações das metaheurísticas propostas e os resultados da solução ótima, desvios em média relativamente baixos para a maioria das instâncias teste.

Este trabalho destaca-se por validar seus resultados comparando-os com resultados de uma metodologia exata, com a qual provou-se uma otimalidade em torno de um gap de 5% em relação à solução ótima para 90% dos casos de teste aqui considerados, dentre os quais alguns destes, possuem dimensões elevadas com mais de 155.000 variáveis.

Em suma, os resultados mostram de forma clara e veemente o potencial da abordagem aqui proposta, onde soluções de alta qualidade são obtidas, em tempos de processamento expressivamente baixos, para instâncias consideradas reais, de dimensões relativamente grandes, mostrando assim, a grande eficácia das metaheurísticas, em tratar tais problemas, onde encontram-se repletos de objetivos e regras operacionais.

Pode-se propor, como extensão deste trabalho, o emprego de um método híbrido o qual, por exemplo, poderia combinar a metaheurística GRASP com um método exato. Uma possível proposta poderia considerar o GRASP gerando uma solução inicial utilizando sua particularidade semi-gulosa, e esta solução inicial seria submetida a um procedimento de Programação Matemática.

Referências Bibliográficas

[1] P. Avella and I. Vasil'ev. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8:497–514, 2005.

[2] S. Binato, W. J. Hery, D. Loewenstern, and M. G. C. Resende. Approximate solution of the job shop scheduling problem using grasp. *Technical Report, AT&T Labs Research*, 1999.

[3] E. Burke. Practice and theory of automated timetabling. In P. Ross, editor, *Lecture Notes in Computer Science*. Springer, 1996.

[4] E. Burke. Practice and theory of automated timetabling ii. In M. Carter, editor, *Lecture Notes in Computer Science*. Springer, 1997.

[5] E. Burke. Practice and theory of automated timetabling iii. In W. Erben, editor, *Lecture Notes in Computer Science*. Springer, 2000.

[6] J. N. Clímaco, C. H. Antunes, and M. J. G. Alves. *Programação Linear Multi-objetivo*. Imprensa da Universidade de Coimbra, Portugal, 2003.

[7] V. Cung and et al. Strategies for the parallel implementation of metaheuristics. In *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, 2002.

[8] S. Daskalaki. An integer progamming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153:117–135, 2004.

[9] A. T. Ernst, H. Jiang, and M. Krishnamoorthy. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research 153*, pages 3–27, 2004.

[10] T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally dificult set covering problem. *Operations Research Letters*, (8):67–71, 1989.

[11] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–134, 1995.

[12] T. A. Feo, M. G. C. Resende, and S. H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, (42):860–878, 1994.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman, San Francisco, January 1979.

[14] F. Glover. Future paths for integer programming and artificial intelligence. *Computers and Operations Research, Amsterdam*, 13:533–549, 1986.

[15] F. Glover. Tabu search - part i. *ORSA Journal on Computing*, (1):190–206, 1989.

[16] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, (2):4–32, 1990.

[17] F. W. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. International Series in Operations Research e Management Science. Kluwer Academic Publishers, January 2003.

[18] F. Glover and M. Laguna. Tabu search. *Kluwer Academic Publishers*, 1997.

[19] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[20] P. Hansen and N. Mladenovic. Variable neighbourhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[21] P. Hansen, N. Mladenovic, and D. Urosevic. Variable neighborhood search and local branching. *Computers and Operations Research*, 33:3034–3045, 2006.

[22] J. P. Hart and A. W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.

[23] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, (34):975–986, 1984.

[24] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[25] R. L. Milidiú, A. A. Pessoa, V. Braconi, E. S. Laber, and P. A. Rey. Um algoritmo GRASP para o problema de transporte de derivados de petróleo em oleodutos. *XXXIII Simpósio Brasileiro de Pesquisa Operacional*, pages 237–246, 2001.

[26] T. F. Noronha and D. J. Aloise. Algoritmos e estratégias de solução para o problema do gerenciamento de sondas de produção terrestre na bacia petrolífera potiguar. *REIC - Revista Eletrônica de Iniciação Científica (http://www.sbc.org.br/reic/edi)*, I(II):1–11, 2001.

[27] L. S. Ochi. Técnicas de inteligência computacional. Notas de aula, Universidade Federal Fluminense, Niterói, RJ, 2002.

[28] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization - Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[29] K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *Journal of the Operational Research Society*, 54:230–238, 2003.

[30] M. G. C. Resende. Greedy randomized adaptive search procedures (grasp). *AT&T Labs Research Technical Report*, 1998.

[31] M. G. C. Resende. Computing approximate solutions of the maximum covering problem using grasp. *J. of Heuristics*, (4):161–171, 1998.

[32] M. G. C. Resende, T. A. Feo, and S. H. Smith. Fortran subroutines for approximate solution of maximum independent set problems using GRASP. *ACM Transactions on Mathematical Software*, pages 386–394, 1998.

[33] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. *AT&T Labs Research Technical Report*, 2002.

[34] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, page 219–249. Kluwer, 2003.

[35] M. G. C. Resende and C. C. Ribeiro. GRASP with path-relinking: Recent advances and applications. In *Metaheuristics: Progress as Real Problem Solvers*, pages 29–63. Kluwer Academic Publishers, 2005.

[36] H. G. Santos. *Formulações e Algoritmos para o Problema de Programação de Horários em Escolas*. PhD thesis, Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, Niterói, RJ, Brasil, 2007.

[37] H. G. Santos, L. S. Ochi, and M. J. F. Souza. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *ACM Journal of Experimental Algorithmics*, 10(2.9):1–16, 2005.

[38] H. G. Santos, L. S. Ochi, and M. J. F. Souza. An efficient tabu search heuristic for the school timetabling problem. *Lecture Notes in Computer Science*, 3059:468–481, 2004.

[39] H. G. Santos and M. J. F. Souza. Programação de horários em instituições educacionais: Formulações e algoritmos. *XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO*, pages 2827-2882, 2007.

[40] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, (13):87–127, 1999.

[41] A. Schaerf. Tabu search techniques for large high-school timetabling problems. *Computer Science/Department of Interactive Systems*, 1996, Technical Report.

[42] A. Schaerf. Local search techniques for large high-school timetabling problems. *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS*, pages 368–377, 1999.

[43] G. Schmidt, and T. Strohlen Timetabling Construction - an annotated bibliography. *The Computer Journal*, (23):307–316, 1979.

[44] M. Souza. *Programação de Horários em Escolas: Uma Aproximação por Metaheurísticas*. PhD thesis, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro - COPPE/UFRJ, Rio de janeiro, Brasil, 2000.

[45] M. J. F. Souza, F. P. Costa, and I. F. G. Guimarães. Um algoritmo evolutivo híbrido para o problema de programação de horários em escolas. *XXII Encontro Nacional de Engenharia de Produção*, 2002.

[46] M. J. F. Souza, N. Maculan, and L. S. Ochi. Melhorando quadros de horário de escolas através de caminhos mínimos. *TEMA - Tendências em Matemática Aplicada e Computacional*, 2:515–524, 2000.

[47] M. J. F. Souza, A. X. Martins, C. R. Araújo, and F. W. A. Costa. Método de pesquisa em vizinhança variável aplicado ao problema de alocação de salas. *XXII Encontro Nacional de Engenharia de Produção*, 2002.

[48] M. J. F. Souza, L. S. Ochi, and N. Maculan. *A Grasp-Tabu Search Algorithm for Solving School Timetabling Problems*, chapter In Metaheuristics: Computer Decision Making. Eds. Kluwer Academic Publ., Boston, MA, pages 659–672, 2003.

[49] C. D. Stefano and A. Tettamanzi. An evolutionary algorithm for solving the school time-tabling problem. *Proceedings of the EvoWorkshops on Applications of Evolutionary Computing*, 2037:452–462, 2001.

[50] I. R. Sucupira and F. S. C. Silva. Métodos heurísticos genéricos - metaheurísticas e hiper-heurísticas. *Universidade de São Paulo - Instituto de Matemática e Estatística - Departamento de Ciência da Computação*, 2004, Technical Report.

[51] A. Tortelly Jr. and L. S. Ochi. Um GRASP eficiente para problemas de roteamento de uma frota de veículos. *TEMA - Tendências em Matemática Aplicada e Computacional*, (1):149–158, 2006.

[52] A. Tripathy. School timetabling - a case in large binary integer linear programming. *Management Science*, 30:1473–1489, 1984.