UNIVERSIDADE FEDERAL FLUMINENSE

FERNANDO LOURENÇO PINHO COSTA

AMBIENTE DE SOFTWARE PARA ELABORAÇÃO E GERENCIAMENTO DE MODELOS DE PROGRAMAÇÃO LINEAR E INTEIRA

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória.

Orientadores:
Prof. Celso Carneiro Ribeiro
Prof. Leonardo Gresta Paulino Murta

Niterói
2012

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

C837 Costa, Fernando Lourenço Pinho
Ambiente de software para elaboração e gerenciamento de modelos de programação linear e inteira / Fernando Lourenço Pinho Costa. – Niterói, RJ : [s.n.], 2012.
150 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2012.
Orientadores: Celso Carneiro Ribeiro, Leonardo Gresta Paulino Murta.

1. Otimização combinatória. 2. Modelagem matemática. 3. Programação linear. 4. Programação inteira. I. Título.

CDD 519.7

AMBIENTE DE SOFTWARE PARA
ELABORAÇÃO E GERENCIAMENTO DE
MODELOS DE PROGRAMAÇÃO LINEAR E INTEIRA

FERNANDO LOURENÇO PINHO COSTA

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória.

BANCA EXAMINADORA

_____

Prof. Celso Carneiro Ribeiro, IC-UFF (Orientador)

_____

Prof. Leonardo Gresta Paulino Murta, IC-UFF (Orientador)

_____

Prof. Simone de Lima Martins, IC-UFF

_____

Prof. Márcio de Oliveira Barros, UNIRIO

Niterói, 13 de abril de 2012

Aos meus pais Fernando e Beatriz,
à minha irmã Júlia, ao meu sobrinho Pedro,
à minha avó Laidinha e à minha querida Sabrina.

AGRADECIMENTOS

A Deus, que me deu forças para enfrentar todos os desafios.

Aos meus pais Fernando e Beatriz, à minha avó Laidinha e à minha irmã Júlia pelo amor e por estarem sempre presentes em minha vida.

À minha esposa Sabrina pelo carinho e pela ajuda nas revisões desta dissertação.

Aos meus professores e orientadores Leonardo Murta e Celso Ribeiro, que me incentivaram e me mostraram o caminho a seguir na elaboração do GeMM.

Aos professores do Instituto de Computação da UFF, especialmente a professora Simone Martins e o professor Luiz Satoru Ochi, pelos ensinamentos.

Às secretárias da pós-graduação Teresa e Viviane pela prestatividade e cordialidade no atendimento das solicitações.

Aos meus amigos Arthur, Bruno Diuana, Paulo Marcos, Luiz Fernando, Samuel, Pedro Paulo, Bruno Bastos e Leonardo Gil pelo companheirismo em todos os momentos.

À equipe de Pesquisa Operacional da Petrobras (PESOP) pelo apoio no desenvolvimento do GeMM.

À Petrobras e ao meu gerente Roberto Iachan pelo apoio que me permitiu realizar as tarefas do mestrado paralelamente ao trabalho na companhia, principalmente durante as disciplinas.

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.”

Albert Einstein

RESUMO

Essa dissertação trata das características dos ambientes de software para modelagem matemática e propõe um sistema para modelagem e gerenciamento de modelos de problemas de programação linear e inteira. As principais características deste ambiente são: o controle da evolução dos modelos e dos dados utilizando controle de versão; a arquitetura cliente-servidor, que permite a interação do sistema tanto com os modeladores quanto com os tomadores de decisão; a utilização de banco de dados para armazenar as informações sobre os modelos e os cenários de dados; e a utilização de servidores de otimização remotos, que permitem executar as otimizações em máquinas diferentes das que geram e tratam os modelos e seus dados. O ambiente de modelagem proposto nesta dissertação foi validado através de modelos matemáticos que explorassem diferentes características, tais como: o tratamento das condições de geração de variáveis e de restrições, a utilização de parâmetros calculados a partir de outros parâmetros, a mistura de variáveis inteiras e contínuas em modelos mistos, entre outras. Esta validação mostrou que o ambiente é capaz de tratar modelos encontrados em diferentes áreas de aplicação de Pesquisa Operacional e resolver problemas com dezenas de milhares de variáveis e restrições.

Palavras-chave: Modelagem Matemática, Programação Linear, Programação Inteira, Controle de Versão.

ABSTRACT

This work addresses characteristics of software environments for mathematical modeling and proposes a system for modeling and management of models of linear and integer programming problems. The main features of this modeling environment are: version control of models and data; client-server architecture, which allows interaction among modelers and decision makers; the use of a database to store information about the models and data scenarios; and the use of remote servers of optimization, which allows to solve the optimization problems on different machines. The modeling environment proposed in this work was evaluated using mathematical models which exploit different characteristics, such as the treatment of conditions for generating variables and constraints, the usage of calculated parameters derived from other parameters, the mixture of integers and continuous variables in mixed models, among others. This evaluation showed that it is able to treat models found in various application areas of Operations Research and solve problems with tens of thousands of variables and constraints.

Keywords: Mathematical Modeling, Linear Programming, Integer Programming, Version Control.

Índice de figuras

Índice de tabelas

LISTA DE ABREVIATURAS E SIGLAS

API	: *Application Programming Interface*
DER	: Diagrama de Entidades e Relacionamentos
GeMM	: Gerenciador de Modelos Matemáticos
IDE	: *Integrated Development Environment*
IHC	: Interface Homem-Computador
JPA	: *Java Persistence API*
LHS	: *Left-Hand Side*
PDF	: *Portable Document Format*
PI	: Programação Inteira
PIM	: Programação Inteira Mista
PL	: Programação Linear
PPEP	: Problema de Programação de Entrega de Pedidos
PPL	: Problema de Programação Linear
PPLI	: Problema de Programação Linear e Inteira
RHS	: *Right-Hand Side*
SGBD	: Sistema Gerenciador de Banco de Dados
SQL	: *Structured Query Language*
TTPPV	: *Traveling Tournament Problem with Predefined Venues*
UML	: *Unified Modeling Language*
XML	: *eXtensible Markup Language*

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO

1.1 MOTIVAÇÃO

Em uma grande companhia, a área de Pesquisa Operacional desenvolve modelos matemáticos para resolver problemas de otimização nas diversas áreas de negócio. Os profissionais que tomam as decisões nessas áreas de negócio utilizam tais modelos através de softwares em que podem analisar os diversos cenários, a fim de apoiar as decisões que devem ser tomadas. Esses profissionais não precisam conhecer os métodos de modelagem matemática, uma vez que interagem com um software que encapsula as técnicas de Pesquisa Operacional e permite o tratamento e a análise dos dados envolvidos na tomada de decisão, que são inerentes ao domínio do negócio. Os profissionais da área de Pesquisa Operacional devem conhecer as técnicas de resolução dos problemas de otimização para fornecer ferramentas aos tomadores de decisão que gerem respostas adequadas aos problemas encontrados na companhia.

A qualidade de um sistema de apoio à decisão é medida não apenas pelos resultados obtidos, mas também pela facilidade de interação entre os usuários, a aplicação e os dados necessários, e pela agilidade do seu desenvolvimento, uma vez que o tempo é uma variável importante nas tomadas de decisão. Os tomadores de decisão esperam que os sistemas de Pesquisa Operacional forneçam resultados coerentes, no tempo adequado, que tragam ganho às áreas de negócio e que tratem adequadamente as informações envolvidas. Assim, alguns desafios do desenvolvimento de sistemas de Pesquisa Operacional podem ser identificados, tais como: a obtenção e o gerenciamento dos dados necessários para a modelagem do problema e a construção dos diferentes cenários; o controle da evolução dos modelos matemáticos, dos dados associados e dos resultados obtidos; e a elaboração da interface homem-computador (IHC), para os profissionais interagirem com o sistema de apoio à decisão.

Com isso, pode-se associar o processo de desenvolvimento de um modelo matemático, que é utilizado dentro da cadeia de tomada de decisão de uma organização, com o processo de desenvolvimento de software. A área da Engenharia de Software possui conceitos que podem ser aplicados também na construção e na manutenção de modelos matemáticos e de sistemas de apoio à decisão. Segundo PRESSMAN (2006), existem cinco fases em um modelo de desenvolvimento de software, com diferentes tarefas em cada uma delas, que devem ser executadas para a construção de um software que atenda às necessidades definidas pelos seus

futuros usuários: a comunicação, quando é feita a iniciação do projeto e o levantamento dos requisitos; o planejamento, fase para estimar o custo e o prazo, e elaborar o cronograma do projeto; a modelagem, quando o software e sua arquitetura são projetados; a construção, quando o software é codificado e testado; e a implantação, fase para entregar o software aos seus usuários e planejar a manutenção.

Para apoiar algumas dessas atividades é possível usar técnicas de gerência de configuração, área da Engenharia de Software que busca controlar a evolução de sistemas complexos. Os sistemas evoluem ao longo do tempo, uma vez que seus requisitos mudam e defeitos são encontrados. Contribuições importantes da gerência de configuração são o controle de versão, gerenciando repositórios de componentes de software, e o controle de mudança, apoiando o processo de desenvolvimento de software (ESTUBLIER, 2000).

Comparando com as atividades necessárias para o desenvolvimento de um modelo matemático, que resolve um problema de otimização e é utilizado no processo decisório de uma organização, algumas semelhanças com as tarefas descritas anteriormente podem ser observadas. Atividades como levantamento de requisitos junto aos usuários (neste caso, os tomadores de decisão), o projeto e a implementação do sistema, o teste, a implantação e a manutenção também podem ser encontradas na elaboração de ferramentas de apoio à decisão. Desta forma, a motivação deste trabalho está no apoio ao desenvolvimento de sistemas de Pesquisa Operacional, principalmente envolvendo modelagem matemática, que tratam as informações das áreas de negócio e geram resultados para auxiliar nas tomadas de decisão.

1.2 OBJETIVOS

O objetivo maior dessa dissertação é desenvolver um ambiente de software para a modelagem de problemas de programação linear e inteira, que trate não apenas a modelagem, mas também o ciclo de vida dos modelos matemáticos e dos dados associados. Além disso, o ambiente deve apoiar o compartilhamento de informações entre seus usuários. Esse ambiente, denominado GeMM, Gerenciador de Modelos Matemáticos, deve atender às necessidades tanto dos modeladores, quanto dos tomadores de decisão. As principais características do GeMM são: a modelagem de problemas de Programação Linear (PL), Programação Inteira (PI) e Programação Inteira Mista (PIM); o controle de versão dos modelos matemáticos e dos dados associados; a geração de uma aplicação a partir de um modelo matemático para a entrada de dados e análise dos resultados; o uso de banco de dados para armazenar os modelos e os dados; e a geração de documentação dos modelos matemáticos.

Como objetivos específicos deste trabalho pode-se citar: a elaboração de um estudo baseado em revisão sistemática da literatura para a identificação das características dos ambientes de modelagem e das pesquisas feitas sobre softwares para apoio à tomada de decisão; a aplicação de um questionário junto a profissionais e estudantes da área de Pesquisa Operacional, a fim de identificar características do processo de desenvolvimento de modelos matemáticos que podem ser apoiadas por ambientes de modelagem; a implementação do GeMM, atendendo as características identificadas; e a validação do GeMM em termos de sua capacidade para modelar diferentes problemas de otimização de PL, PI e PIM.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em seis capítulos, incluindo esta introdução. O Capítulo 2 apresenta a revisão da literatura elaborada e descreve o estudo feito para a identificação das características dos ambientes de modelagem. O Capítulo 3 apresenta os conceitos utilizados na elaboração do GeMM. Ele descreve a arquitetura do sistema, o tratamento das informações dos modelos matemáticos, o conceito da metamodelagem para tratamento dos dados e as características do controle de versão utilizado.

O Capítulo 4 apresenta o detalhamento das características de implementação do GeMM. Ele mostra também como os usuários podem utilizar o ambiente para modelar os Problemas de Programação Linear e Inteira (PPLI), ou seja, os problemas de PL, PI e PIM. No Capítulo 5 é feita a validação do GeMM como ambiente de modelagem de problemas de otimização. Por fim, o Capítulo 6 resume as contribuições do trabalho, suas limitações e sugestões para desenvolvimentos futuros.

CAPÍTULO 2 - AMBIENTES DE MODELAGEM MATEMÁTICA

2.1 INTRODUÇÃO

A primeira atividade desenvolvida com o objetivo de elaborar um novo ambiente de software para criar e gerenciar modelos matemáticos foi a pesquisa na literatura por trabalhos que indicassem as principais características desejadas nesse tipo de ambiente. Os softwares já existentes e como eles podem ser integrados para atender às necessidades de modelagem e da utilização de modelos matemáticos no processo de tomada de decisão também foram pesquisados.

Um estudo baseado em revisão sistemática da literatura foi feito para identificar trabalhos relacionados com a modelagem de problemas de otimização. Para complementar esse estudo, também foi elaborada uma pesquisa de opinião com profissionais e estudantes de Pesquisa Operacional. Esses estudos buscaram, principalmente, responder à seguinte questão: quais os requisitos mais importantes que devem ser atendidos por um ambiente de modelagem matemática, tanto para os desenvolvedores dos modelos quanto para os seus usuários, os tomadores de decisão? Nas seções a seguir são descritos os estudos e os seus resultados.

2.2 METODOLOGIA USADA NA REVISÃO DA LITERATURA

Um estudo baseado em revisão sistemática da literatura é um meio de identificar, avaliar e interpretar uma questão particular a partir da pesquisa disponível para uma área de interesse (KITCHENHAM, 2004). Ela tem por objetivo gerar uma síntese da literatura existente sobre um determinado assunto, fazer uma pesquisa abrangente, rigorosa e não tendenciosa, deixando explícitos os critérios de seleção de publicações, de forma que possa ser avaliada e reproduzida por outros pesquisadores, permitindo que seu resultado tenha maior valor científico.

A população identificada para responder à questão desse estudo, ou seja, quais os requisitos desejados para um ambiente de modelagem, são as pesquisas que envolvem os desenvolvedores de modelos matemáticos, os tomadores de decisão e os projetos de desenvolvimento de ambientes e linguagens de modelagem existentes. A intervenção são os elementos da Engenharia de Software utilizados em sistemas e ambientes para modelagem matemática de PPLI. Os resultados esperados desse estudo são as características e requisitos dos ambientes de modelagem para PPLI.

A estratégia para fazer essa pesquisa foi identificar inicialmente quais os termos mais apropriados que deveriam estar na consulta e em quais fontes de informação essa consulta

deveria ser feita para que fossem encontrados os resultados esperados. Em seguida, foram definidos os critérios para inclusão e exclusão de trabalhos a serem analisados para responder a questão definida. A estratégia, a definição da consulta e das fontes, bem como os trabalhos que foram analisados e os que foram excluídos desse estudo estão detalhados no Apêndice A.

Seguem as análises das características dos principais trabalhos obtidos através da revisão sistemática da literatura e de outros trabalhos que também orientaram a identificação dos requisitos dos ambientes de modelagem para PPLI.

2.3 TRABALHOS RELACIONADOS

O artigo de FOURER (2011) ajudou a definir a nomenclatura e os elementos de software envolvidos na resolução de PPLI. Ele fez uma pesquisa no mercado e na academia a fim de identificar e classificar as principais ferramentas envolvidas na resolução desse tipo de problema. Essas ferramentas podem ser separadas em dois grupos: resolvedores e modeladores. Os resolvedores são softwares que implementam os algoritmos de solução, recebem uma instância de um PPLI e retornam seus resultados. Os softwares de modelagem fazem a interface entre os profissionais modeladores e os softwares resolvedores, fornecendo formas gerais e intuitivas para expressar modelos simbólicos de representação da realidade. Eles também oferecem, normalmente, funcionalidades de importação e processamento dos dados, geração das instâncias dos problemas para os resolvedores, análise dos resultados e interface com outras aplicações. Entre essas aplicações são destacados os Sistemas Gerenciadores de Banco de Dados (SGBD) e os softwares de planilhas eletrônicas. Esse estudo também classificou os softwares quanto às plataformas, que se referem tanto ao hardware quanto ao sistema operacional, os tipos de licenças, os preços praticados pelos fornecedores, os tipos de problemas que são resolvidos e os principais algoritmos de solução.

GEOFFRION (1989) descreve cinco características desejáveis para ambientes de modelagem por meio de computador para apoiar o trabalho de Pesquisa Operacional: tratar todo o ciclo de vida da modelagem, não apenas uma parte; ser aderente às necessidades dos tomadores de decisão e de outras pessoas envolvidas, não apenas dos profissionais de Pesquisa Operacional; facilitar a evolução do modelo e dos sistemas ao seu redor ao longo da sua existência; possuir uma linguagem de definição do modelo independente do paradigma usado para a resolução do problema; e permitir o fácil gerenciamento de recursos usados no processo de modelagem, tais como os dados, os modelos e os resolvedores, por exemplo. Essas características ajudaram a direcionar a definição dos requisitos para um novo ambiente de modelagem e a pesquisa por trabalhos relacionados na literatura.

O objetivo principal do GeMM, ambiente apresentado neste trabalho, é a modelagem de PPLI. MURPHY *et al.* (1992) apresentam uma série de esquemas para representação de problemas desse tipo. Entre essas representações é possível destacar os geradores de matrizes, as representações algébricas, a modelagem estruturada e o uso de esquemas de bancos de dados. Os geradores de matrizes, como o OMNI (HAVERLY, 2001), foram os primeiros modeladores. Esses geradores fornecem uma linguagem procedural para permitir a criação de arquivos no formato MPS (IBM CORPORATION, 1975), que são interpretados pela maioria dos resolvedores e representam uma instância de um PPLI. Apesar do formato MPS permitir que matrizes esparsas sejam representadas de maneira bastante eficiente, este formato dificulta a depuração dos resultados e dos modelos (BROOK *et al.*, 1988). A representação algébrica, na qual os modelos são descritos na forma de expressões matemáticas, usada em sistemas como GAMS (BISSCHOP e MEERAUS, 1982) e AMPL (FOURER *et al.*, 1990), é bastante geral e concisa. A modelagem estruturada proposta por GEOFFRION (1987) tem como objetivo desenvolver uma especificação geral para representar de forma inequívoca todos os elementos essenciais de uma variedade de modelos de representação da realidade. Outra forma de representação é a utilização de esquemas de bancos de dados. Existem dois requisitos distintos nesta representação, que devem estar relacionados: a necessidade de registrar informações sobre a estrutura do modelo matemático e de armazenar os dados do problema, bem como os resultados que são obtidos. Além dessas representações, ainda existem as formas gráficas que usam grafos ou esquemas de blocos para representar os problemas de otimização.

Linguagens textuais são amplamente utilizadas para descrever modelos matemáticos, como as representações algébricas, por exemplo. MATURANA (1994) analisa algumas das principais questões que surgem no projeto de linguagens de modelagem e que podem ser aplicadas em qualquer linguagem de modelagem algébrica, para ajudar a avaliar seus pontos fortes e fracos. Esse trabalho ainda aborda itens como a verificação de erros de modelagem e a integração de softwares para obtenção dos dados dos problemas. GREENBERG (1987) desenvolveu uma linguagem natural, textual, mas não algébrica, para explicar um modelo de programação linear e encontrar a sua solução. A principal ideia é criar regras de formação em uma linguagem natural, no caso o inglês, para elaborar um modelo matemático de programação linear e permitir a sua passagem para um resolvedor encontrar uma solução computacionalmente. Esse tipo de linguagem torna a definição do modelo autoexplicativa e

de fácil entendimento para os usuários, porém não consegue ser tão concisa quanto às linguagens algébricas para modelos mais complexos.

Uma forma de modelar problemas é através do uso da lógica de programação e de elementos de programação procedural. Por mais funcionalidades que um ambiente de modelagem possua, em algum nível uma linguagem algébrica ou de programação deve ser utilizada na criação dos modelos. MCALOON e TRETKOFF (1997) discutem a integração da lógica, da modelagem e da programação a fim de resolver problemas em Pesquisa Operacional. O objetivo desse estudo é integrar a modelagem em uma estrutura de programação e utilizar elementos da programação procedural para resolver problemas de otimização. MAROS e KHALIQ (2002) destacam a complexidade da concepção e execução de softwares de otimização e discutem as melhores práticas para a elaboração desse tipo de software. Definem também critérios de um bom software de otimização e destacam que, embora problemas de otimização e algoritmos de solução sejam bastante heterogêneos, ainda assim possuem características em comum, o que torna razoável a combinação dessas características em sistemas genéricos. Os principais atributos analisados em um sistema de otimização são: robustez, eficiência, capacidade, portabilidade, escalabilidade do uso da memória, modularidade, extensibilidade, manutenibilidade e interface com o usuário.

Uma outra forma de elaborar e manter modelos matemáticos é através de representações gráficas, usando grafos ou diagramas, por exemplo. COLLAUD e PASQUIER-BOLTUCK (1994) propõem uma nova linguagem de modelagem para PPLI baseada em grafos. Os autores definiram uma ferramenta de software com a capacidade de elaborar e armazenar o modelo matemático, chamar bibliotecas externas, traduzir a representação gráfica para a representação textual e vice-versa, obter a solução e gerar relatórios, obter dados de um banco de dados e reutilizar estruturas pré-definidas.

Em uma abordagem seguindo esta mesma linha, FORSTER e MEVERT (1994) também propõem a modelagem matemática através da representação gráfica. A principal ideia deste trabalho é facilitar o entendimento da modelagem pelos não especialistas em Pesquisa Operacional, facilitando a comunicação entre os modeladores e os tomadores de decisão. A modelagem gráfica é feita através da representação do problema como diagramas de fluxos em rede.

CHARI e SEN (1998) também tratam de um sistema que utiliza a modelagem estruturada para permitir a formulação, a manutenção e a resolução de modelos matemáticos através da representação gráfica. A utilização da representação gráfica facilita a compreensão

do usuário e os modelos podem ser facilmente avaliados e analisados. As principais características do sistema proposto são: interface com usuário para criar, manter e resolver os modelos matemáticos, apoio à modelagem de problemas de diversos domínios, construção gráfica do modelo usando pouca ou nenhuma notação algébrica, verificação da sintaxe no momento da edição para prevenir inconsistências, visões em diferentes níveis de detalhe dos modelos para facilitar a compreensão, comunicação com diversos resolvedores e automatização da aquisição de dados, fornecendo a integração com softwares de planilhas eletrônicas e banco de dados.

A representação gráfica, como feita por COLLAUD e PASQUIER-BOLTUCK (1994) e FORSTER e MEVERT (1994), torna o desenvolvimento de modelos bastante intuitivo e visual. Porém, modelos grandes e complexos se mostram difíceis de serem desenvolvidos e mantidos nessas representações, pela quantidade de nós e ligações em um mesmo grafo ou diagrama.

Uma característica destacada em muitos trabalhos é a separação entre os dados e o modelo, principalmente através do uso de aplicações de bancos de dados e planilhas eletrônicas. O tratamento dos dados se mostrou de grande relevância e os SGBD's se mostraram úteis nesta tarefa, tanto para armazenamento dos dados, quanto na representação dos modelos. LEE (1991) aborda o armazenamento de modelos matemáticos separado dos dados de diferentes instâncias do problema. Modelos devem ser gerais para tratar uma gama de problemas comuns e não devem ser armazenados de forma a resolver uma única instância do problema. Nesse trabalho, o autor apresenta uma linguagem que foi desenvolvida para armazenar modelos de programação matemática e mostra maneiras eficientes para guardar esses modelos em bancos de dados.

A utilização de bancos de dados para tratar modelos matemáticos é explorada por FOURER (1997), que apresenta estruturas de bancos de dados para modelos de programação matemática. Apesar de ser uma estrutura para um problema de programação matemática específico, mostra que é uma abordagem interessante para problemas de grande escala. Atualmente, os SGBD's são amplamente utilizados nas organizações, gerindo e centralizando as principais informações. Logo, os sistemas de apoio à decisão de alguma forma precisam estar integrados com estes repositórios de informações. SRINIVASAN e SUNDARAM (2000) também propõem a utilização de tecnologias de gerenciamento de dados para sistemas de modelagem. A proposta consiste em desenvolver um sistema baseado na modelagem estruturada e em sistemas de banco de dados relacional. A opção por essa arquitetura é a

flexibilidade de escolha da interface com os usuários, a habilidade de lidar com implementações escaláveis e a possibilidade de tratar aspectos chave no ciclo de vida da modelagem, através de componentes modulares.

Uma característica importante da atividade de modelagem é a independência das ferramentas utilizadas na resolução dos problemas de otimização. Por exemplo, os ambientes de modelagem devem ser capazes de usar diferentes resolvedores, permitindo a integração com as melhores opções do mercado e da academia. Nesta linha, RAMIREZ *et al.* (1993) apresentam uma estrutura de software cujo objetivo é tornar independente o resolvedor dos dados associados ao modelo matemático. A ideia dessa separação é reutilizar componentes e possibilitar a utilização de diferentes softwares de banco de dados e resolvedores na tomada de decisão. Esse trabalho apresenta um sistema, chamado DAMS, que implementa a independência entre modelo/dados e modelo/resolvedor, permitindo o mapeamento entre diversos resolvedores, modelos e fonte de dados. Ele foi construído sobre um sistema gerenciador de banco de dados relacional e provê tanto o gerenciamento dos dados quanto dos modelos, estendendo a linguagem SQL.

Os conceitos de orientação a objetos, a divisão dos modelos em blocos ou módulos menores, que permitem o reuso de componentes e a construção de modelos através da composição de submodelos são características também abordadas pelos trabalhos pesquisados. LENARD (1993) e MUHANNA (1993) tratam da representação do modelo por meio da orientação a objetos. A principal ideia consiste em definir classes que representam um problema ou um grupo de problemas de otimização, juntamente com a utilização de um banco de dados relacional para gerenciar esses modelos. Com base em uma estrutura orientada a objetos para a representação de modelos, MA (1997) propõe a regulamentação da tipagem e do uso da teoria de herança para a concepção de sistemas de modelagem. Exemplos apresentados mostram que o uso dos conceitos da orientação a objetos para o desenvolvimento de modelos permite reduzir a redundância e evitar inconsistências na modelagem.

A ideia de dividir modelos em blocos para facilitar e automatizar a construção de novos modelos, promovendo a reutilização de funcionalidades, não é recente. MURPHY e STOHR (1986) propuseram um sistema que utiliza técnicas de inteligência artificial e bancos de dados para auxiliar na formulação de grandes PPLI's. As principais características do sistema proposto são: dividir grandes modelos em subunidades compreensíveis, simplificar a identificação de variáveis e restrições na modelagem, prover verificação lógica da

formulação, gerar a matriz de entrada ou se comunicar diretamente com um resolvedor e facilitar a identificação de falhas nos modelos e a geração de modelos alternativos para o mesmo problema. KWON e PARK (1996) apresentam um sistema para prover a reutilização de modelos, utilizando a ideia de modelagem reversa. Ao contrário das tarefas executadas para modelagem convencional de problemas reais, a modelagem reversa foca na identificação, extração e reorganização de um modelo em construção a partir de modelos existentes, o que permite reutilizar experiências anteriores.

PILLUTLA e NAG (1996) fizeram um estudo para a construção automática de modelos que resolvem problemas recorrentes da indústria. Por exemplo, o problema de planejamento de produção é um domínio típico, em que os modelos estruturados são encontrados e reaproveitados. Uma vez que o modelo tenha sido estabelecido, a questão não é mais resolver a modelagem do problema dentro desse domínio, mas sim a sua resolução com os dados específicos do problema. A proposta desse trabalho é automatizar a resolução de problemas do mesmo domínio a partir de padrões identificados. Nesta linha, MATURANA *et al.* (2004) apresentam um projeto de um gerador de modelo que tem por objetivo automatizar o processo de desenvolvimento. O objetivo é integrar uma especificação do modelo com um banco de dados, um resolvedor e com a interface de usuário. O gerador de modelo cria automaticamente o banco de dados para armazenar os dados, as interfaces com os usuários e com o resolvedor a partir do modelo matemático.

O uso de modelos matemáticos para a tomada de decisão em ambientes distribuídos, com pessoas envolvidas no processo em diferentes locais e com diferentes habilidades, é abordado por HUH e KIM (2004) e GREENBERG e MURPHY (1995). HUH e KIM (2004) propõem um modelo de dados orientado a objetos para o desenvolvimento de um sistema de gestão de modelos de forma colaborativa, que facilita não só o compartilhamento de modelos matemáticos entre os vários usuários do sistema, mas também a coordenação e a propagação de mudanças nos modelos em tempo real. Como principais características desta proposta pode-se citar o uso de bancos de dados orientados a objetos para compartilhamento de informações entre os envolvidos e a notificação automática de mudança nas interfaces com os usuários. GREENBERG e MURPHY (1995) abordam a questão das múltiplas visões necessárias para um sistema de tomada de decisão, uma vez que modelos matemáticos de larga escala são construídos e gerenciados por profissionais com diferentes habilidades. Visões específicas de um modelo podem conter a modelagem algébrica, esquema de blocos, gráficos e textos. Forma e conteúdo são importantes para auxiliar a compreensão dos problemas e dos modelos por usuários de diferentes perfis.

MAKOWSKI (2005) discute os requisitos necessários para a construção de ambientes de modelagem. O autor afirma que os softwares de modelagem disponíveis atendem a apenas uma ou duas fases do ciclo de desenvolvimento de modelos. De acordo com esse trabalho, as etapas do ciclo de modelagem são: levantamento dos requisitos, projeto, construção, testes, uso, revisão, manutenção, documentação, análise do modelo, análise sobre os resultados e a evolução do modelo. Questões como o processamento de dados, a documentação das fontes de dados, o controle de modificações, a utilização de resultados de um modelo como entrada de outro e as permissões de acesso são especialmente críticas para modelos complexos ou de larga escala. As atividades identificadas como mal apoiadas pelas ferramentas de modelagem incluem o controle de versão da especificação do modelo, o tratamento dos dados necessários para a definição dos parâmetros, a criação de uma instância do modelo, definida por uma especificação e um correspondente conjunto de dados e, por fim, a análise dos resultados, com diferentes visões sobre os dados. Algumas dessas atividades são tratadas por GREENBERG (1996), que aborda um sistema para apoiar análises que podem ser feitas com modelos de programação linear. Ele foca em três funcionalidades: análise da sensibilidade, análise de defeitos, com foco na identificação de inviabilidades, e gerenciamento do modelo, com tratamento da documentação, validação e verificação dos modelos de programação linear. Porém, outras atividades levantadas por MAKOWSKI (2005), como o controle de versão e de modificações, por exemplo, não foram tratadas por nenhum trabalho pesquisado.

Muitas tecnologias atuais também podem auxiliar na construção de sistemas de apoio à decisão, além dos SGBD's. FOURER (1998) trata do uso de tecnologia *web* em sistemas de otimização e FOURER *et al.* (2010) propõem um servidor de otimização que utiliza serviços *web* e o formato eX*tensible Markup Language* (XML). Esse tipo de serviço é bastante útil, pois permite a separação de hardware e software entre linguagens de modelagem, resolvedores e sistemas de informação. Esses trabalhos mostram como a evolução da computação e da Engenharia de Software tem ajudado no desenvolvimento de sistemas de apoio à decisão e na resolução de problemas de otimização.

2.4 PESQUISA DE OPINIÃO

Complementarmente ao estudo baseado em revisão sistemática da literatura, foi executada uma pesquisa de opinião com o objetivo de identificar, junto a desenvolvedores de modelos matemáticos, as características importantes de um ambiente de modelagem e de que forma a Engenharia de Software pode auxiliar neste desenvolvimento. Esta pesquisa focou principalmente na modelagem de PPLI. O questionário utilizado na pesquisa de opinião e suas

respostas podem ser vistos no Apêndice B. Os participantes dessa pesquisa de opinião foram os profissionais da equipe de Pesquisa Operacional da Petrobras e os estudantes de mestrado e doutorado da UFF na área de otimização. No total, foram 19 os participantes da pesquisa.

As primeiras perguntas dessa pesquisa de opinião buscaram identificar o perfil do participante. Na sequência, o objetivo foi identificar que tipos de problemas são resolvidos, quais as ferramentas são utilizadas para modelagem e resolução dos problemas de otimização, assim como perfil de utilização dos modelos matemáticos para tomada de decisão. Por fim, foram identificados os processos de desenvolvimento e manutenção utilizados, como é feito o controle de versão e a documentação dos modelos implementados.

Apesar do universo da pesquisa não ter sido grande, com apenas 19 participantes, o perfil deles se mostrou bastante qualificado, com 74% de mestres e 32% de doutores ou doutorandos, e com experiência, uma vez que cerca de metade dos participantes já desenvolveu modelos matemáticos com aplicações diretamente na indústria. Os problemas de otimização resolvidos são, principalmente, PPLI's e a principal linguagem e ferramenta de modelagem é o AIMMS (PARAGON DECISION TECHNOLOGY, 2011). Pelas características do AIMMS, é possível concluir que a facilidade de utilização e o tratamento dos dados são pontos importantes na escolha da ferramenta de modelagem.

Uma questão relevante que foi levantada é o compartilhamento do trabalho. Em 63% das respostas, o desenvolvimento dos modelos é feito com equipes de três ou mais desenvolvedores. Além disso, em 83% das respostas, os modelos desenvolvidos são usados por outros profissionais que conhecem o problema, mas não conhecem sua modelagem matemática. Logo, é necessário o uso de ferramentas que integrem o trabalho desses profissionais e que os usuários dos modelos possam informar os dados e obter os resultados sem o conhecimento das técnicas de Pesquisa Operacional. Em 89% das respostas os problemas de programação matemática não são otimizados nos computadores em que são implementados. Assim, uma arquitetura que permita a execução das otimizações em um ambiente remoto é desejável.

Cinco perguntas tiveram o objetivo de identificar o uso de controle de versão e da gerência de configuração no desenvolvimento dos modelos matemáticos. Pelas respostas, a maioria utiliza algum tipo de controle de versão, porém o processo de gerência de configuração se mostrou falho na identificação de quem solicitou as modificações e as razões delas ocorrerem. Foi identificado também que os dados das instâncias estão associados a uma versão específica do modelo e que, na grande maioria das vezes, os dados também são

afetados e devem ser tratados para se adequarem às modificações feitas. Com essas respostas, pode-se concluir que os dados e os modelos estão intimamente ligados e que a manutenção do modelo normalmente impacta no tratamento dos dados. Pelas respostas obtidas na pesquisa de opinião, as ferramentas de modelagem usadas não são integradas com sistemas de controle de versão.

As últimas perguntas tiveram por finalidade identificar objetivamente as características mais importantes dos ambientes de modelagem. Na Tabela 2.1 podem ser vistas as características que foram identificadas como importante ou muito importante e o percentual de participantes da pesquisa que as identificaram desta forma.

Tabela 2.1: Importância das características de um ambiente de modelagem

Características	**Importante ou Muito Importante**
Facilidade de integração dos dados com o modelo	94%
Performance na geração da instância do problema para passagem ao resolvedor	84%
Facilidade no tratamento dos dados do modelo	83%
Linguagem de modelagem intuitiva e de fácil aprendizado	83%
Facilidade na entrada de dados e na visualização dos resultados	78%
Facilidade de separação das informações do modelo e dos dados associados	77%
Integração com a maior quantidade de resolvedores possíveis	61%
Integração com sistemas externos	50%
Geração de documentação automática	34%
Controle de versão no modelo matemático	28%

2.5 CONSIDERAÇÕES FINAIS

No estudo baseado em revisão sistemática da literatura foram analisados diversos trabalhos e identificadas algumas características desejadas em ambientes de modelagem matemática e nos sistemas de apoio à decisão. O questionário montado para a pesquisa de opinião teve foco na obtenção de informações do uso prático de ambientes de modelagem e permitiu identificar características importantes do dia a dia do desenvolvimento de modelos matemáticos.

A partir dos resultados dos dois estudos e procurando responder à pergunta inicial, ou seja, “quais os requisitos mais importantes que devem ser atendidos em um ambiente de modelagem matemática, tanto para os desenvolvedores dos modelos quanto para os seus usuários, os tomadores de decisão?”, é possível concluir que características como o tratamento dos dados separados do modelo, a utilização de bancos de dados, integração com outros softwares, o uso de interfaces gráficas intuitivas para a entrada de dados e análise dos

resultados pelos tomadores de decisão e a comunicação eficiente com os resolvedores são críticas no desenvolvimento deste tipo de aplicação. Entre os trabalhos analisados e os softwares existentes não foi identificado um ambiente de modelagem matemática para PPLI que atendesse às necessidades tanto dos modeladores, quanto dos tomadores de decisão, que permitisse a interação entre esses profissionais, que tratasse o ciclo de vida dos modelos matemáticos e dos dados associados de forma conjunta, e que fornecesse apoio para controle de versão. Assim surgiu a proposta de elaboração do GeMM para cobrir estes requisitos. Na Seção 4.8 os principais trabalhos relacionados são comparados com a abordagem do GeMM.

O próximo capítulo apresenta as principais características e conceitos utilizados no desenvolvimento do GeMM, que procuraram abranger os requisitos identificados. O Capítulo 3 descreve a arquitetura do sistema para atender tanto a modeladores, quanto a tomadores de decisão, e se comunicar com os resolvedores para otimizar as instâncias dos PPLI's. Ele ainda apresenta as estruturas de dados para o armazenamento dos modelos matemáticos e seus cenários em um banco de dados e mostra os conceitos do controle de versões e do controle de concorrência que são implementados pelo GeMM.

CAPÍTULO 3 - CONCEITOS DO GEMM

3.1 INTRODUÇÃO

Como o Capítulo 2 expôs, um ambiente de modelagem matemática é um software que permite descrever matematicamente um determinado problema real, a fim de resolvê-lo por meio de técnicas de programação linear e inteira, por exemplo. A principal função dos ambientes de modelagem é fornecer aos usuários uma ferramenta para modelar o problema da forma mais simples e flexível possível. Além disso, este tipo de sistema deve ser capaz de converter o modelo e os dados do problema para formatos que permitam aos resolvedores tratar e gerar soluções (FOURER, 2011).

O GeMM se propõe não apenas a fornecer uma ferramenta para a escrita de equações matemáticas, mas também a controlar o ciclo de vida de um modelo, desde a sua elaboração até a sua utilização pelos tomadores de decisão. Para isso, o GeMM utiliza uma arquitetura cliente-servidor, armazena suas informações em um banco de dados, oferece controle de versão tanto dos modelos matemáticos quanto dos dados associados e permite o compartilhamento de informações entre os usuários do sistema, observando o controle de concorrência.

As seções a seguir apresentam a arquitetura e as principais características do GeMM, como ele trata a modelagem matemática e os dados dos modelos, além de mostrar como é feito o controle das versões dos modelos e dos dados associados.

3.2 ARQUITETURA

Para a definição da arquitetura do sistema, foram identificados dois papéis distintos e essenciais no desenvolvimento de modelos matemáticos que são utilizados no apoio à tomada de decisão. O primeiro é o modelador, profissional de Pesquisa Operacional que conhece no detalhe as técnicas de modelagem para resolver problemas na área de otimização. O outro papel é dos tomadores de decisão, que não precisam ter conhecimentos específicos de Pesquisa Operacional, mas devem conhecer as informações necessárias e utilizar os modelos de otimização para apoiar os processos de tomada de decisão dentro de uma organização.

Desta forma, o sistema proposto neste trabalho foi projetado com uma arquitetura cliente-servidor. Nessa arquitetura, é possível ter módulos para atender diferentes funções que se interligam através da aplicação central e da base de dados única. Além disso, essa arquitetura permite que o sistema seja multiusuário, com dois tipos de perfis principais: modeladores e tomadores de decisão. A Figura 3.1 mostra a arquitetura geral do sistema.

Basicamente, os modeladores interagem com a aplicação criando e mantendo os modelos matemáticos com o intuito de resolver os problemas de otimização. Os tomadores de decisão informam os dados de entrada, com a possibilidade de criação de diversos cenários para um mesmo problema de otimização, e visualizam os resultados obtidos pelos resolvedores. Com essa estrutura, cada usuário do sistema tem um ambiente adequado à realização do seu trabalho, seja ele de modelagem ou de uso do modelo, mas esses ambientes podem ser compartilhados e interligados pela base de dados única.

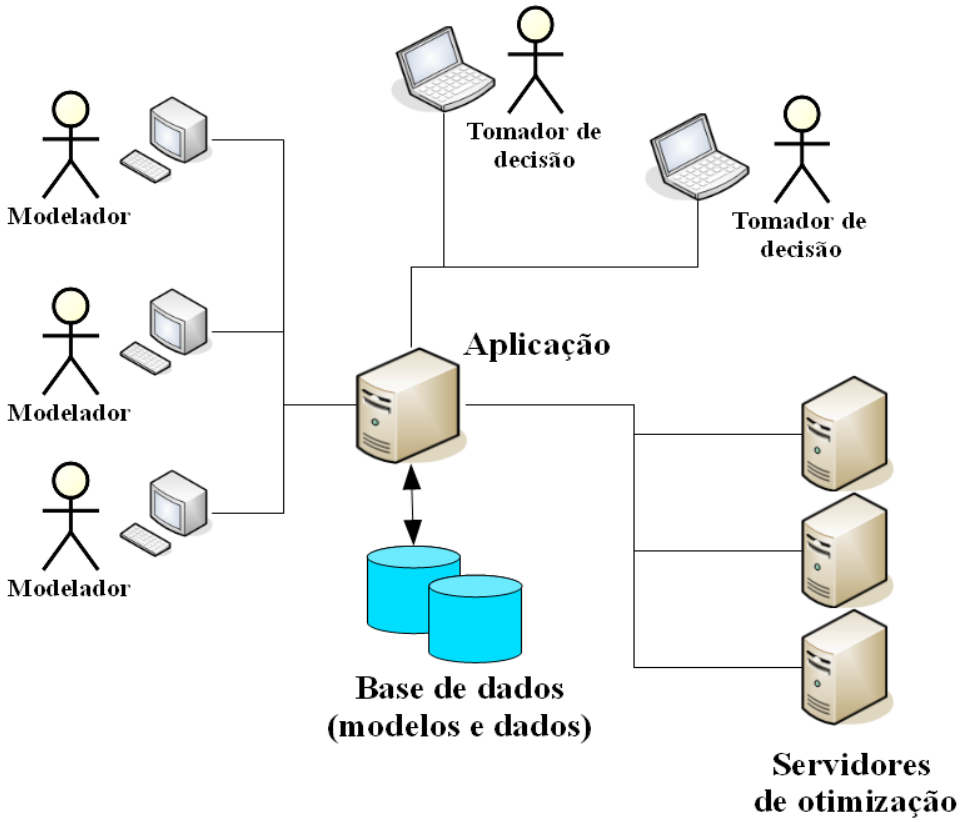


Figura 3.1: Arquitetura básica do sistema

Outro elemento importante na arquitetura do sistema são os servidores de otimização. Para problemas de grande porte, que precisam de grande poder de processamento a fim de encontrar soluções ótimas em um tempo adequado, normalmente utilizam-se máquinas com hardware e software dedicados à execução dos processos de otimização. Além disso, é bastante desejável que essas máquinas sejam compartilhadas para a resolução de diferentes problemas em momentos distintos. Muitas vezes, o computador de mesa não tem a capacidade adequada de executar o processamento necessário. Desta forma, aparece na arquitetura do GeMM a figura dos servidores de otimização. O servidor da aplicação também é responsável

por gerenciar as execuções das otimizações, na medida em que controla a distribuição entre os servidores específicos para otimização, a concorrência das otimizações e até a prioridade de um determinado usuário ou projeto sobre outros. Os resolvedores, que solucionam os problemas de otimização, são executados nos servidores de otimização.

3.3 MODELAGEM MATEMÁTICA

A principal função do GeMM é realizar a modelagem de problemas de programação linear e inteira. Na abordagem utilizada, a modelagem dos problemas é feita diretamente no ambiente, através de formulários, e é armazenada em um banco de dados, no qual também são guardados os dados dos problemas. Assim, nesta arquitetura, o modelo e os dados podem ser compartilhados entre os usuários, o que permite uma maior integração no trabalho de modelagem. A interação entre os usuários do GeMM é discutida na Seção 3.5.

Os elementos de modelagem foram projetados observando as representações algébricas, principalmente da linguagem AMPL (FOURER *et al.*, 1990), apesar do GeMM utilizar um banco de dados para persistir os modelos de PPLI. As informações dos modelos e dos dados ficam na mesma estrutura de banco de dados, porém são armazenados de forma independente. Essa separação é importante, pois a partir de um modelo matemático é possível criar diferentes instâncias do mesmo problema de otimização, apenas com a alteração dos dados de entrada. Para a modelagem de um PPLI são utilizadas cinco entidades principais: conjuntos, parâmetros, variáveis, restrições e função objetivo.

Os conjuntos são coleções de objetos distintos e bem definidos pertinentes ao problema. Cada conjunto possui índices a ele associados, que permitem referenciar no modelo um elemento genérico pertencente ao conjunto e são essenciais na descrição dos PPLI's. Um tipo bastante comum de conjunto, por exemplo, é uma sequência de números inteiros. No GeMM é possível também a definição de subconjuntos dos conjuntos identificados.

Os parâmetros no modelo matemático são invariantes utilizados na modelagem. Eles podem ser valores que devem ser informados diretamente pelos usuários (dados de entrada) ou calculados a partir de uma expressão. Esta diferença existe apenas do ponto de vista do ambiente de modelagem, que deve determinar o valor dos parâmetros antes do processo de otimização. Do ponto de vista do resolvedor, todos os parâmetros são dados de entrada. Eles podem representar valores escalares, quando não são indexados, ou ser indexados pelos elementos de um ou mais conjuntos. Neste caso, os parâmetros funcionam como vetores ou matrizes, cujos índices são os próprios elementos dos conjuntos.

As variáveis são os elementos de modelagem que devem ser calculados pelo processo de otimização. Assim como os parâmetros, as variáveis também podem ser indexadas pelos índices dos conjuntos. Uma variável pode ser do tipo real, inteiro ou binário, uma vez que o sistema permite a modelagem de PPLI. As variáveis devem ter os valores limites, superior e inferior, definidos. As restrições são representadas por equações ou inequações lineares. Para permitir a modelagem de problemas complexos, o GeMM permite a definição de condições de geração de variáveis e restrições no modelo. E, por fim, a função objetivo é uma expressão linear que o problema de otimização deve procurar maximizar ou minimizar.

Uma característica importante do GeMM é a separação entre os dados e o modelo matemático. Um conjunto de dados de entrada define uma instância do problema de otimização. A estrutura do banco de dados do GeMM deve ser capaz de armazenar o modelo, os dados de entrada e os resultados para qualquer PPLI. Para gerenciar as informações dos modelos, projetou-se a estrutura de dados apresentada no diagrama de classes da Figura 3.2.

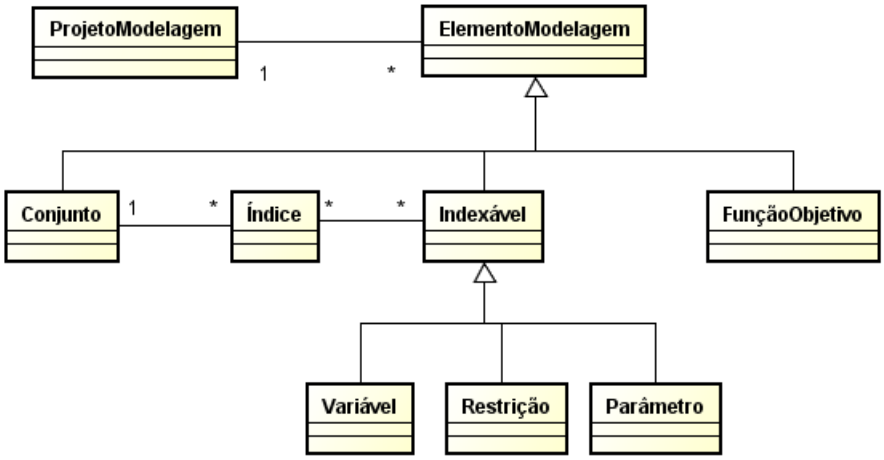


Figura 3.2: Diagrama de classes de representação do modelo matemático

Neste diagrama, a classe *ProjetoModelagem* representa um PPLI a ser resolvido. Um projeto de modelagem agrupa todos os elementos de modelagem que devem ser criados para a correta representação do problema. Os elementos de modelagem são representados pela classe *ElementoModelagem* e por suas subclasses no diagrama da Figura 3.2: *Conjunto*, *Variável*, *Restrição*, *Parâmetro* e *FunçãoObjetivo*. Pode-se associar um ou mais índices a cada conjunto, cuja representação é feita pela classe *Índice*. A classe *Indexável* representa genericamente os elementos que podem ser indexados, ou seja, ter índices.

Essa estrutura de dados é usada para tratar e armazenar os modelos matemáticos no GeMM. É importante observar que esta estrutura compreende apenas a definição do modelo, ou seja, não contempla os dados de entrada e os resultados. Por exemplo, nela é possível identificar quais conjuntos estão definidos, mas não é possível saber quais elementos compõem tais conjuntos e quais valores podem ser atribuídos aos índices. Assim como pode-se saber quais os parâmetros estão definidos, embora não se conheça os valores que são atribuídos a eles. A estrutura apresentada na Figura 3.2 também não contempla o versionamento do modelo, que é discutido posteriormente.

3.4 TRATAMENTO DOS DADOS

Os elementos contidos nos conjuntos e os valores dos parâmetros são os dados de entrada para um problema de otimização. Essas informações definem um cenário de dados de um PPLI e cada cenário de dados associado a um modelo gera uma instância diferente do problema. Um software resolvedor espera receber uma instância do problema como entrada, no qual ele aplica um ou mais métodos de solução e retorna os resultados (FOURER, 2011), que são os valores das variáveis e o valor da função objetivo.

Como foi descrito na Seção 3.2, os usuários do GeMM estão divididos em dois grupos principais: modeladores e tomadores de decisão. Observando a função de cada um desses profissionais, suas habilidades e o uso que devem fazer do GeMM, pode-se afirmar que os modeladores criam e manejam principalmente a definição dos modelos matemáticos, que são utilizados pelos tomadores de decisão. Os tomadores de decisão trabalham com os dados de entrada, executam os problemas de otimização e analisam os resultados obtidos a fim de utilizá-los no processo decisório. Esse último grupo de usuários não deve alterar a definição do modelo, ou seja, a definição das variáveis, das equações ou inequações das restrições ou da expressão da função objetivo, mas deve trabalhar com os valores dos elementos dos conjuntos, os valores dos parâmetros e os resultados obtidos.

3.4.1 ESTRUTURA DE DADOS

Cada modelo matemático precisa dos dados em uma estrutura diferente. Em um determinado projeto de modelagem, pode-se criar um ou mais conjuntos, cujos índices podem indexar diversos parâmetros e variáveis, com diferentes combinações. A abordagem utilizada no GeMM associa a indexação dos elementos de modelagem com as chaves primárias das entidades de banco de dados. Cada entidade de banco de dados possui um conjunto de atributos que caracterizam a sua chave primária. Comparando com a definição dos modelos

matemáticos, cada atributo da chave primária pode ser associado com um índice e a chave primária completa pode ser comparada com a combinação dos índices que indexam os elementos de modelagem. Os valores propriamente ditos dos parâmetros e das variáveis, por exemplo, são representados por atributos que estão fora da chave primária da entidade. Assim como os valores dos atributos que compõem a chave primária não se repetem e identificam unicamente um registro de uma tabela do banco de dados, uma combinação específica dos índices de um parâmetro ou variável, identificam unicamente o seu valor.

A seguir é apresentado um Problema de Programação Linear (PPL) em sua forma canônica para ilustrar essa abordagem (BAZARAA, 1977):

$$Minimizar \quad \sum_{j=1}^{n} c_j . x_j \tag{1}$$

$$Sujeito \ \ a \quad \sum_{j=1}^{n} a_{ij} . x_j \geq b_i , \quad i = 1, \ldots, m \tag{2}$$

$$x_j \geq 0 \quad j = 1, \ldots, n \tag{3}$$

Os índices *i* e *j*, que aparecem nas expressões (1), (2) e (3), representam elementos de conjuntos. O índice *i* representa um elemento que pertence ao conjunto $I=\{1, 2, \ldots, m\}$, como apresentado na expressão (2) e o índice *j* representa um elemento que pertence ao conjunto $J=\{1, 2, \ldots, n\}$, como apresentado na expressão (3). Considerando um cenário de dados desse modelo, definido pelos conjuntos $I=\{1, 2, 3\}$ e $J=\{1,2\}$, uma estrutura tabular para armazenar tais conjuntos pode ser vista na Tabela 3.1.

Tabela 3.1: Estrutura de dados dos conjuntos *I* e *J*

I
1
2
3

J
1
2

Uma vez informados os elementos de cada um dos conjuntos, pode-se definir a estrutura de dados para armazenar e tratar os dados dos parâmetros e variáveis que são indexados pelos índices associados a esses conjuntos. Na modelagem do PPL em sua forma canônica, é possível identificar os parâmetros c_j, b_i e a_{ij}, e a variável x_j. A Tabela 3.2 mostra as três entidades de banco de dados criadas para armazenar os dados dos parâmetros e da variável, que são: a entidade cuja chave primária são os valores combinados de *i* e *j*, a

entidade cuja chave primária são os valores de *i* e a entidade cuja chave primária são os valores de *j*.

Tabela 3.2: Estrutura de dados dos parâmetros e variáveis

Entidade I e J

i	j	a_{ij}
1	1	
1	2	
2	1	
2	2	
3	1	
3	2	

Entidade I

i	b_i
1	
2	
3	

Entidade J

j	c_j	x_j
1		
2		

Essas estruturas de dados foram elaboradas observando os índices dos parâmetros e das variáveis. Neste exemplo, o parâmetro a_{ij} é indexado pelos índices $i \in I$ e $j \in J$. Logo, a entidade para armazenar os valores desse parâmetro deve ter como chave primária a combinação dos elementos dos conjuntos *I* e *J*. Dessa mesma forma foi desenvolvida a estrutura de dados para o parâmetro b_i, que deve estar em uma entidade cuja chave primária são os elementos do conjunto *I.* Por outro lado, o parâmetro c_j e a variável x_j devem estar em uma entidade cuja chave primária são os elementos do conjunto *J*. Assim, as colunas *i* e *j* das entidades definidas na Tabela 3.2 são chaves estrangeiras, respectivamente, das colunas *I* e *J*, definidas nas estruturas de dados da Tabela 3.1. Este exemplo ilustra as estruturas de dados necessárias para armazenar os dados associados ao modelo do PPL em sua forma canônica.

Na abordagem do GeMM, pode se afirmar que, para um dado modelo matemático, sempre vai existir uma entidade de banco de dados para cada um dos conjuntos, onde são armazenados seus elementos. Os valores dos elementos são a chave primária da entidade, uma vez que não existem elementos repetidos no mesmo conjunto. Para armazenar os valores dos parâmetros e das variáveis, sempre vai existir uma entidade de banco de dados para cada combinação diferente dos índices associados a eles. Essas entidades devem ter um atributo para cada índice que compõe a sua chave primária e mais um atributo para cada variável ou parâmetro que possuir a mesma combinação de índices. Todas as variáveis ou parâmetros que possuem a mesma combinação de índices, na mesma ordem, devem ter um atributo associado na mesma entidade de banco de dados, como ocorreu com o parâmetro c_j e a variável x_j, no exemplo anterior. Uma entidade de banco de dados sem atributos associados a índices também deve ser criada para conter os valores dos elementos que não são indexados, como a função objetivo e parâmetros e variáveis que não possuem índices.

Entretanto, esta forma de tratar os dados dos modelos matemáticos exige que cada modelo tenha uma estrutura de dados diferente. Isso ocorre porque cada modelo matemático tem suas próprias características, como, por exemplo, diferentes definições de conjuntos e índices, bem como dos elementos de modelagem e diferentes combinações de índices que indexam os parâmetros e as variáveis. Sendo assim, cada modelo matemático gerenciado pelo GeMM precisaria de um esquema de banco de dados diferente para armazenar os dados associados a eles. Como o GeMM se propõe a ser um ambiente de modelagem para gerenciar diversos modelos de PPLI, ele utiliza metamodelagem para armazenar e tratar as estruturas de dados necessárias para os modelos, independentemente da forma que estão modelados e da sua área de aplicação.

3.4.2 METAMODELAGEM

A ideia da metamodelagem em banco de dados, que foi utilizada por JEUSFELD e JOHNEN (1994), consiste na construção de uma estrutura de dados genérica para representar logicamente um modelo de dados em um banco de dados relacional, que atenda às especificações definidas. A Figura 3.3 mostra a estrutura de dados para armazenar esquemas de bancos de dados de forma lógica, ou seja, esquemas metamodelados. A classe *Esquema* representa os esquemas de bancos de dados. Um esquema é composto de todas as entidades de um mesmo domínio. A classe *Entidade* representa as entidades existentes no esquema lógico e é composta de atributos.

A classe *Atributo* representa as propriedades dos atributos associados às entidades no metamodelo. Esta classe deve conter as informações de integridade dos atributos necessárias para os sistemas gerenciadores de banco de dados, como a possibilidade de assumir um valor nulo, por exemplo. Os relacionamentos entre as entidades são representados pelas classes *ChavePrimária* e *ChaveEstrangeira*. A classe *ChavePrimária* indica quais atributos das entidades pertencem às suas respectivas chaves primárias, ou seja, quais atributos cujas combinações de valores nunca se repetem. A *ChaveEstrangeira* indica quais atributos da entidade representam as ligações com outras entidades relacionadas.

A Figura 3.3 também apresenta a classe *Registro*, que é a representação lógica de uma linha de uma tabela de banco de dados. A sua função é agrupar os valores dos atributos que compõem um único registro. A classe *Valor* e suas subclasses são identificadas unicamente por um *Registro* (linha) e um *Atributo* (coluna). Ela tem a função de armazenar os dados e as suas subclasses são usadas de acordo com o tipo do atributo: booleano, inteiro, real, cadeia de caracteres etc.

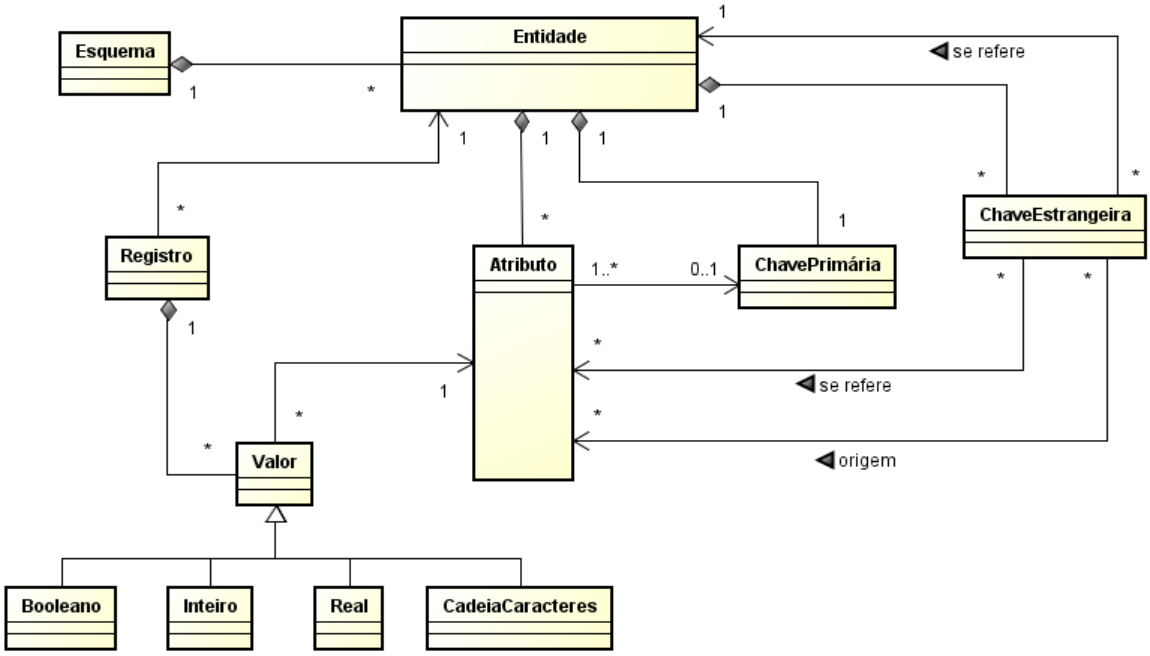


Figura 3.3: Estrutura de dados para armazenar esquemas metamodelados

Nessa estrutura de dados, podem ser identificados dois tipos de classes para a metamodelagem de esquemas: as classes que representam a estrutura de cada esquema, que na Figura 3.3 são *Esquema*, *Entidade*, *Atributo*, *ChavePrimária* e *ChaveEstrangeira;* e as classes que representam os valores dos dados metamodelados, que são *Registro*, *Valor* e suas subclasses.

A metamodelagem permite a utilização de um esquema de banco de dados único para armazenar diferentes modelos de dados. Essa ferramenta é útil quando a estrutura de dados é flexível e pode ser alterada através da manipulação dos dados contidos em uma tabela de banco de dados, e não com a redefinição da estrutura do esquema. Por essas características, o GeMM utiliza a metamodelagem para trabalhar com as estruturas de dados necessárias para armazenar e manejar os dados dos modelos matemáticos. A flexibilidade fornecida pela metamodelagem é necessária para possibilitar mudanças na definição dos modelos sem ter que alterar o esquema do banco de dados, permitindo que se gerencie mais de um modelo pelo mesmo banco de dados.

A estrutura proposta no GeMM para tratar os dados dos modelos através da metamodelagem é apresentada na Figura 3.4. Esta estrutura foi elaborada a partir da estrutura básica da metamodelagem, mostrada na Figura 3.3, com as adaptações necessárias para gerenciar os dados dos modelos matemáticos. As subclasses de *ElementoModelagem* são

classes para representar o modelo matemático e estão nessa ilustração apenas para mostrar como se relacionam com as classes para representação dos dados.

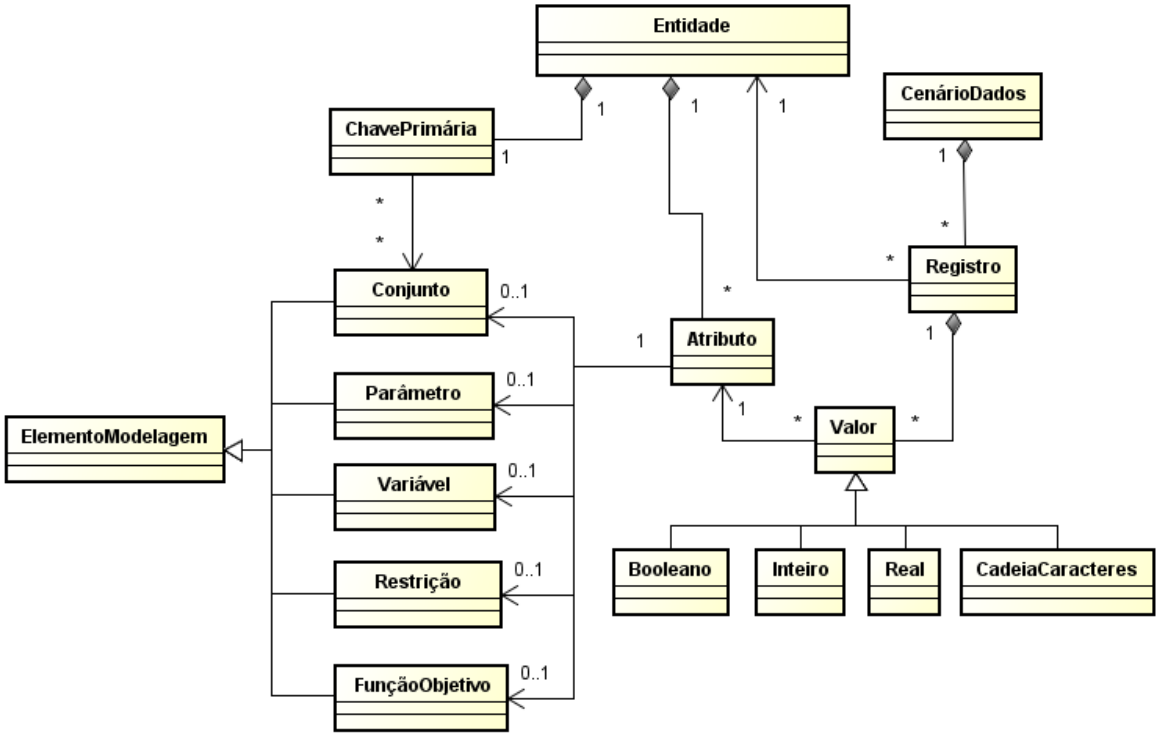


Figura 3.4: Estrutura de dados usando metamodelagem

Nessa abordagem, as entidades necessárias para representar a estrutura de dados dos modelos matemáticos são tratadas pela classe *Entidade*. Como descrito anteriormente, as chaves primárias das entidades são atributos relacionados com os índices dos conjuntos, que indexam os elementos de modelagem. Como os atributos relacionados aos índices dos conjuntos são os únicos que podem ser chaves estrangeiras, o conceito de chave estrangeira foi omitido do diagrama da Figura 3.4, já que está implícita a regra de consistência dos valores contidos nos conjuntos. Os elementos de modelagem que podem receber um valor, como os parâmetros e as variáveis, por exemplo, são associados a um atributo de uma entidade. A entidade que abriga um determinado atributo é definida pela combinação de índices que indexam o determinado elemento de modelagem associado.

Assim, as classes *Entidade*, *ChavePrimária* e *Atributo* se relacionam diretamente com os elementos de modelagem e definem o esquema de dados para um determinado modelo matemático. Cada modelo matemático, uma vez finalizado, possui apenas um esquema de dados. O que pode variar são os dados de entrada e os resultados desse modelo. As classes

CenárioDados, *Registro* e *Valor*, e suas derivadas, são utilizadas para armazenar os dados associados aos modelos.

A classe *Valor* é usada para guardar um valor de um elemento de modelagem. A classe *Atributo* conhece o tipo de dados do elemento de modelagem e a classe *Valor* sabe como armazená-lo. Os elementos de modelagem que possuem um ou mais índices devem ter diversas instâncias de *Valor*, uma para cada combinação dos valores dos índices que indexam esses elementos. A classe *Registro* é responsável por agrupar todos os valores da mesma entidade que possuem a mesma chave primária, ou seja, a mesma combinação dos valores dos índices associados à entidade. Finalmente, a classe *CenárioDados* é responsável por agrupar todos os dados de um único cenário de dados. Ela permite que o mesmo modelo matemático possua diferentes cenários, com valores de parâmetros diferentes, e, consequentemente, resultados diferentes.

3.5 CONTROLE DE VERSÃO

Uma funcionalidade importante do GeMM é o controle de versões dos modelos matemáticos e dos cenários de dados associados aos modelos. Na abordagem utilizada para o desenvolvimento do GeMM, podem existir vários usuários que elaboram e mantém os modelos matemáticos e outros que gerenciam os cenários de dados e executam a otimização dos problemas. Para acompanhar essa interação dos usuários com os modelos matemáticos e os dados, e garantir a integridade das informações, o GeMM possui controle de versões na sua concepção. Entre suas atribuições, destaca-se a capacidade de guardar o histórico das alterações e gerenciar a evolução dos modelos e dos dados ao longo do tempo, o que possibilita o controle de modificações corretivas e melhorias.

3.5.1 CONTROLE DE VERSÃO DOS MODELOS MATEMÁTICOS

CONRADI e WESTFECHTEL (1998) discutem principalmente as características e classificações dos softwares de gerência de configuração. Um ponto importante para esse tipo de software, no qual se inclui os sistemas de controle de versões, é a definição do espaço do produto. O espaço do produto descreve a estrutura completa do que se deseja versionar. No caso do GeMM, o produto é um modelo matemático, composto pelos elementos de modelagem, ou seja, os conjuntos, índices, parâmetros, variáveis, restrições e função objetivo. Ainda segundo CONRADI e WESTFECHTEL (1998), outro ponto importante a ser definido é o modelo de versionamento, que determina quais os itens devem ser versionados, como as versões são organizadas e como são criadas novas versões, assim como qual deve ser a sua

granularidade. A granularidade da versão é definida como o tamanho do menor objeto versionado.

Na abordagem do GeMM, a evolução dos modelos matemáticos ocorre com a criação de diferentes versões para um dado projeto de modelagem. O conceito de versão do projeto de modelagem consiste no grupo de versões dos elementos de modelagem que a compõem. A granularidade do controle de versões adotado pelo GeMM é o elemento de modelagem, ou seja, conjuntos, parâmetros, variáveis, restrições e função objetivo possuem suas versões independentes. Cada um desses elementos é versionado individualmente e a versão de um projeto de modelagem é dada pela maior versão dos seus elementos de modelagem. Uma nova versão é criada motivada pela necessidade de modificação de um ou mais elementos de modelagem dentro do projeto.

Para ilustrar o controle de versão dos modelos matemáticos do GeMM, foi usado o exemplo do projeto de modelagem de um PPL em sua forma canônica. Neste exemplo foram criadas três versões para o mesmo modelo matemático. No primeiro momento foi criado o projeto de modelagem denominado "PPL em sua forma canônica" e sua primeira versão contendo os conjuntos I e J, os parâmetros a_{ij} e b_i, a variável x_j, a restrição e a função objetivo, como mostra a Figura 3.5. Neste momento, esses elementos de modelagem estavam na primeira versão.

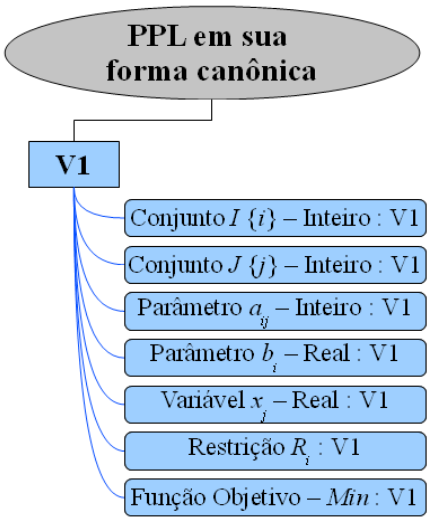


Figura 3.5: Versão inicial do PPL em sua forma canônica

Em seguida, identificou-se um erro na implementação do modelo do PPL em sua forma canônica: o parâmetro a_{ij} foi modelado como inteiro e não como real, como era de se esperar ao analisar a expressão (3). Essa alteração motivou a criação da segunda versão deste projeto de modelagem, apresentada na Figura 3.6.

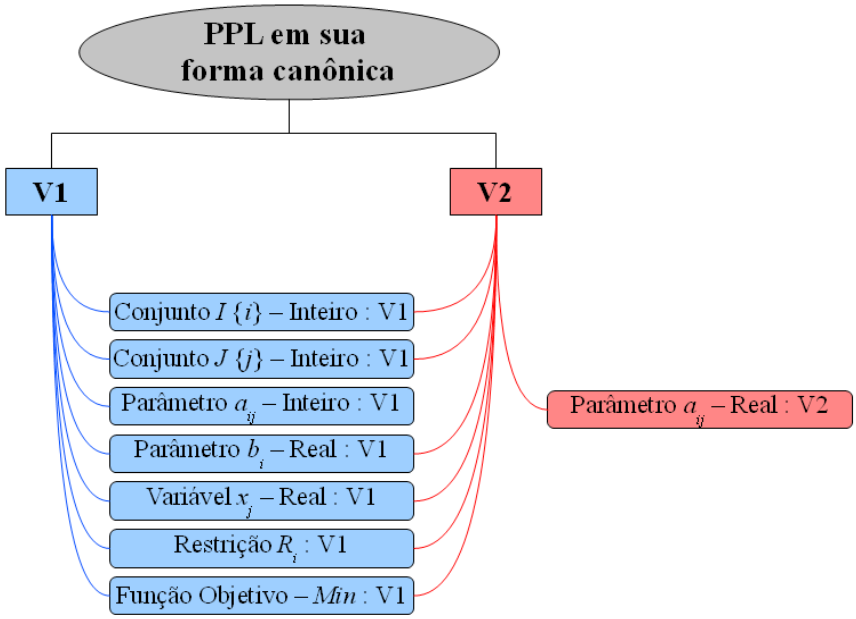


Figura 3.6: Segunda versão do PPL em sua forma canônica

Observa-se que todos os elementos de modelagem associados à segunda versão ainda estão na primeira versão, exceto o parâmetro a_{ij} que foi modificado. Os elementos de modelagem que não foram modificados estão individualmente na primeira versão, porém estão associados à primeira e à segunda versão do projeto de modelagem. Após a finalização da segunda versão, constatou-se mais uma diferença do modelo apresentado nas expressões (1), (2) e (3): os custos das variáveis na função objetivo foram implementados como constantes, de valor *1*, o que motivou a criação da terceira versão, apresentada na Figura 3.7. Nesta versão criou-se o parâmetro c_j e a expressão da função objetivo foi alterada para incluir esse parâmetro no seu equacionamento, em substituição do coeficiente *1* nas variáveis x_j. Como o projeto está na terceira versão, a função objetivo, que foi modificada, e o parâmetro c_j, que foi incluído, também devem estar nesta versão, independentemente da sua última alteração.

Desta forma, cada elemento possui a sua versão individualmente, porém seu ciclo de vida está sempre associado a um determinado projeto de modelagem. Uma vez editado, o

elemento passa ter a mesma versão do projeto em que foi modificado. Assim, é possível identificar em quais versões um único elemento de modelagem foi alterado e, paralelamente, dada uma versão do projeto de modelagem, quais os elementos foram modificados naquela versão. É importante destacar que as informações que não são alteradas, não são replicadas nas diversas versões, elas apenas ficam associadas às versões dos projetos de modelagem. Essa associação só é quebrada quando existe uma alteração no elemento e uma nova versão é criada.

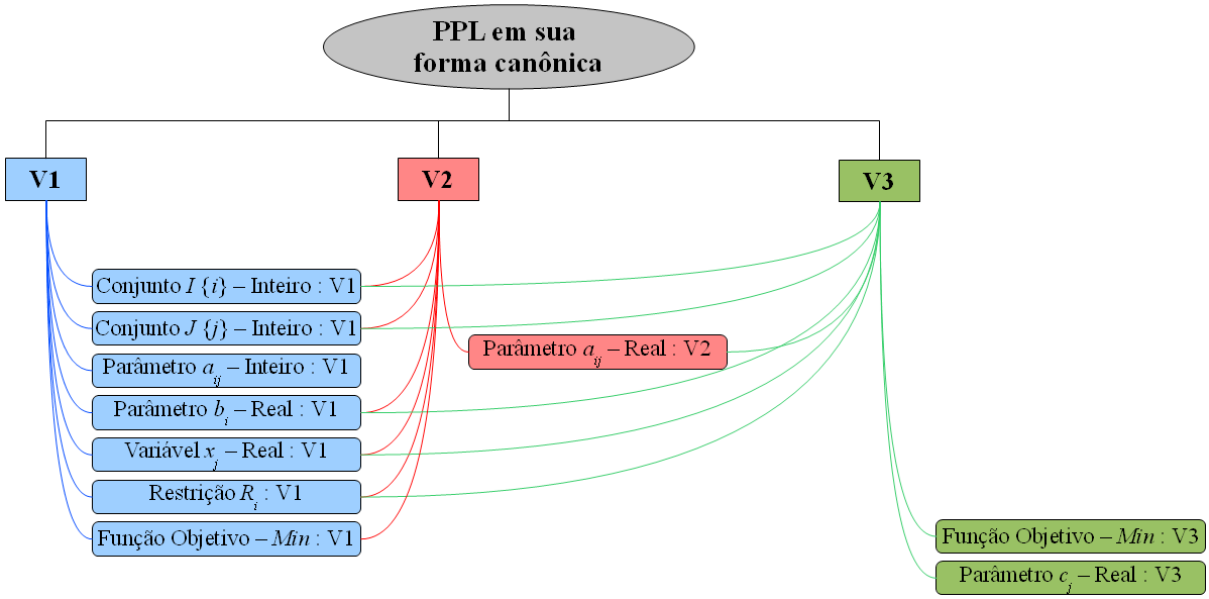


Figura 3.7: Terceira versão do PPL em sua forma canônica

O versionamento utilizado é do tipo global, definido por CONRADI e WESTFECHTEL (1998) como *Product Versioning*, no qual as versões dos elementos ficam no espaço da versão global do projeto de modelagem. Com esse tipo de versionamento, a leitura convencional, "Versão 3 da função objetivo", passa a ser feita na forma "Função objetivo na versão 3 do PPL em sua forma canônica".

Pela característica de ser cliente-servidor e multiusuário, o GeMM permite que mais de um usuário tenha acesso a um determinado projeto de modelagem, desde que possua autorização para isso. Assim, o sistema precisa ter uma política de controle de concorrência, a fim de evitar perda de informações. No trabalho de ESTUBLIER (2001) é discutida a necessidade de controle de concorrência. Em um cenário que *N* usuários trabalham em um mesmo projeto simultaneamente e aplicam as suas modificações em um produto, existem *N+1* produtos diferentes, um com as modificações de cada um dos usuários, mais o produto

original. No caso dessas modificações ocorrerem de forma paralela, é necessário fundir todas essas alterações para gerar o produto final, que conterá todas as alterações propostas.

De acordo com MENS (2002), a necessidade de juntar diferentes versões do mesmo projeto depende do mecanismo de controle de versão escolhido. Com o controle de versão pessimista, no qual apenas um usuário por vez pode modificar elementos de um projeto, ou seja, as alterações não ocorrem em paralelo, a necessidade de juntar modificações é reduzida. MENS (2002) afirma que o processo de fundir ou juntar diferentes modificações é complicado e suscetível a erros, pois é necessário combinar a sintaxe e a semântica do produto em questão. Na abordagem utilizada no GeMM, o controle de alteração é feito por meio da estratégia pessimista, ou seja, por bloqueio do projeto que está sendo alterado. Assim, por esse motivo, o GeMM não trata a junção de modificações (*merge*). Pela granularidade fina utilizada no controle de versão do GeMM, é possível identificar a diferença entre duas versões no nível dos elementos de modelagem, porém na abordagem utilizada não foi tratada a junção de versões de forma automática.

Uma vez que um usuário começa a editar o modelo matemático no GeMM, o projeto inteiro fica bloqueado para a edição deste usuário. Assim que for terminada a modificação, ele pode liberar o projeto para que outro o edite, criando uma nova versão, ou para que essa versão do projeto seja utilizada, com criação dos cenários de dados pelos tomadores de decisão. Na Figura 3.8, o projeto de modelagem está na sua quarta versão. A criação desta versão foi motivada por uma alteração que está sendo feita pelo usuário "Modelador 1". As versões anteriores, V1, V2 e V3, que já foram liberadas, só podem ser acessadas para leitura. Novas modificações só podem ser feitas com a criação de uma nova versão.

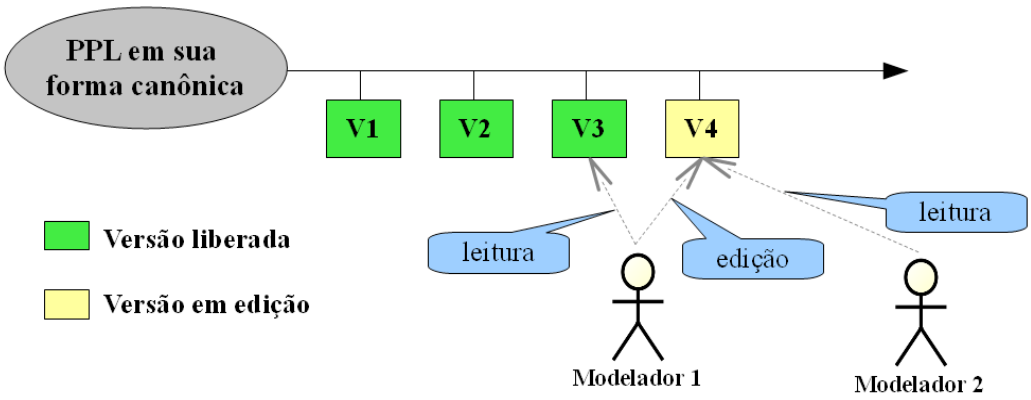


Figura 3.8: Controle de modificação de um projeto de modelagem

A quarta versão, mostrada na Figura 3.8, ainda está em edição pelo usuário "Modelador 1", logo qualquer outro usuário do GeMM, que tenha acesso a este projeto, só pode consultá-lo, até que o "Modelador 1" a libere ou a descarte. Quando uma versão é liberada, esta não pode mais ser removida ou editada. Desta forma, é garantido que todo o ciclo de vida do modelo está registrado. Antes da liberação, enquanto o usuário edita a versão do projeto, ele pode incluir, alterar e remover elementos livremente, como se estivesse na sua própria área de trabalho. Porém, uma vez liberada, qualquer modificação no projeto exige a criação de uma nova versão.

3.5.2 CONTROLE DE VERSÃO DOS CENÁRIOS DE DADOS

Além do controle de versão do projeto de modelagem, existe também o controle de versão dos dados associados. Cada usuário do modelo, na figura do tomador de decisão, pode criar cenários de dados para as versões liberadas do projeto de modelagem, uma vez que sua estrutura não pode ser mais modificada. Cada cenário possui as suas versões e o mesmo conceito do controle de concorrência do modelo matemático é aplicado. Neste caso, o produto a ser versionado são os dados associados aos modelos, ou seja, os valores dos elementos dos conjuntos e os valores dos parâmetros. A Figura 3.9 exibe um exemplo da associação dos cenários e suas versões com as versões dos projetos de modelagem.

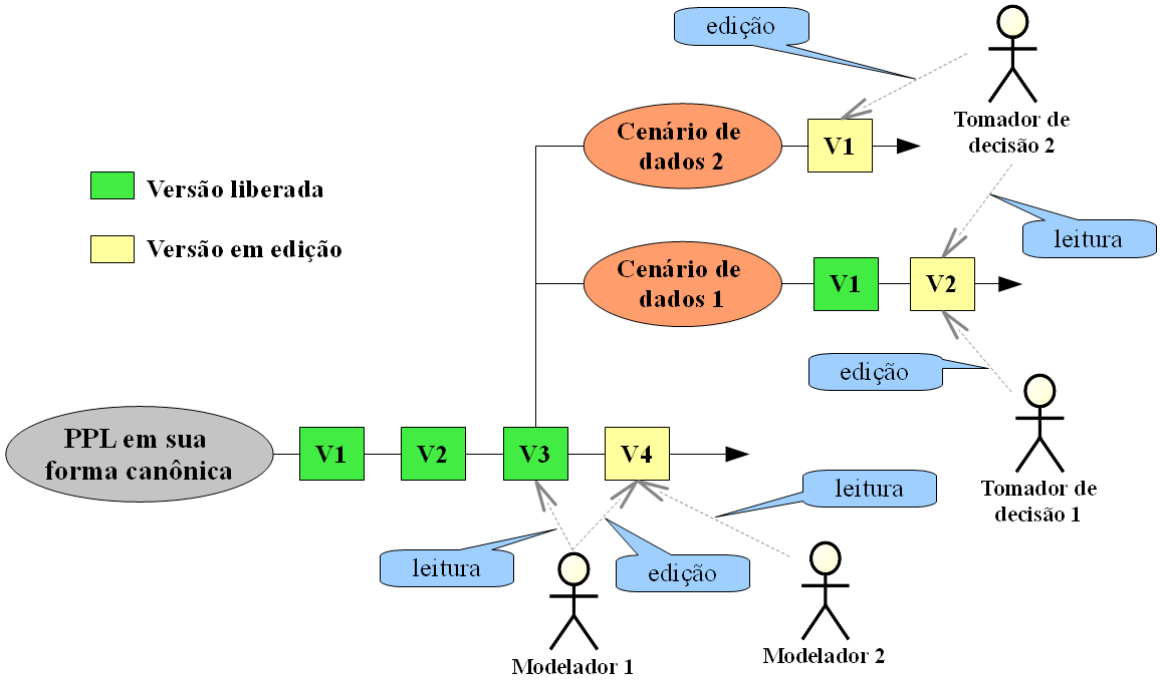


Figura 3.9: Versionamento do projeto de modelagem e dos dados associados

O mecanismo de controle de alteração utilizado também é o pessimista, logo, enquanto um cenário de dados estiver em edição, ele fica bloqueado e os demais usuários só podem visualizá-lo. Quando o cenário é liberado, pode ser modificado com a criação de uma nova versão. No exemplo da Figura 3.9, foram criados dois cenários de dados para a terceira versão do "PPL em sua forma canônica", que já está liberada. O "Cenário de dados 1" possui um determinado conjunto de valores para os elementos de modelagem, e está na sua segunda versão. Como o usuário "Tomador de decisão 1" está editando o "Cenário de dados 1", o usuário "Tomador de decisão 2" não pode alterá-lo, podendo apenas visualizar e copiar suas informações para outro cenário de dados. Neste exemplo, o "Tomador de decisão 2" criou seu próprio cenário para editar suas informações.

Desta forma, os cenários de dados também são gerenciados pelo GeMM. Uma vez que as versões dos cenários de dados que foram liberados não podem ser modificadas ou removidas, o histórico dos dados utilizados nas tomadas de decisão também é guardado. Assim, é possível ter o rastreamento exato do modelo matemático e dos dados que motivaram uma determinada decisão.

3.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou as principais características da abordagem para a elaboração do GeMM. A sua arquitetura distribuída foi projetada principalmente para atender aos modeladores e aos tomadores de decisão, além de permitir que os problemas de otimização sejam resolvidos em servidores de otimização separados. Esses servidores permitem compartilhar hardwares e softwares adequados para a otimização dos PPLI's. Neste ponto, o GeMM pode ser complementado com a proposta de FOURER *et al.* (2010), que permite a execução remota de problemas de otimização utilizando serviços *web* para trocar informações de instâncias de PPLI no formato XML.

Também foram apresentadas as estruturas de dados para tratamento e armazenamento dos modelos matemáticos e seus cenários em um banco de dados. A utilização da metamodelagem se mostrou adequada para armazenar as informações de diversos modelos em um mesmo esquema de banco de dados relacional. Por fim, foram exibidos os conceitos do controle de versões e do controle de concorrência contemplados pelo GeMM, tanto para os modelos matemáticos quanto para os cenários de dados associados. O Capítulo 4 mostra como o GeMM trata e implementa as características descritas neste capítulo.

CAPÍTULO 4 - IMPLEMENTAÇÃO E UTILIZAÇÃO DO GEMM

4.1 INTRODUÇÃO

O capítulo anterior apresentou as principais características da abordagem utilizada na elaboração do GeMM, enquanto este capítulo detalha as questões de implementação para o desenvolvimento do ambiente de modelagem. Para implementar o GeMM, utilizou-se a linguagem de programação Java, que possui muitas ferramentas de desenvolvimento e bibliotecas disponíveis, além de ser portável para diversas plataformas. Ela ainda permite a criação de diferentes módulos do software com a mesma linguagem, como a comunicação via rede e com os sistemas gerenciadores de banco de dados, além de possuir interoperabilidade com outras linguagens, entre elas C e C++, por exemplo.

É importante destacar que a principal função do GeMM é apoiar a modelagem matemática para PPLI. Usando como exemplo o PPL em sua forma canônica, apresenta-se na Seção 4.2 como a modelagem desse tipo de problema é feita no GeMM. Através desse exemplo, mostra-se de que maneira os usuários podem interagir com o ambiente para modelar os PPLI's. Em seguida, apresenta-se a forma como o GeMM armazena e trata as informações dos modelos matemáticos e dos cenários de dados a partir das estruturas de dados apresentadas no capítulo anterior. Utiliza-se um exemplo do problema da dieta (BASSI, 1976) para ilustrar como o GeMM trabalha com os cenários de dados. Na sequência são detalhadas as características de implementação do controle de versão e de concorrência, tanto para os modelos matemáticos quanto para os cenários de dados.

Por fim, descrevem-se as características dos servidores de otimização, elemento importante da arquitetura do GeMM. Os servidores são responsáveis por receber as solicitações de otimização dos cenários de dados dos modelos, gerar instâncias dos PPLI's e passá-las para os resolvedores. Detalha-se como o GeMM gera uma instância de um PPLI a partir de um cenário de dados e um modelo matemático particular. Este capítulo ainda apresenta as ferramentas que apoiaram o desenvolvimento e mostra como os trabalhos relacionados a ambientes de modelagem influenciaram na elaboração do GeMM.

4.2 MODELAGEM MATEMÁTICA

Para mostrar como o GeMM trata a modelagem matemática, utiliza-se o PPL em sua forma canônica descrito no Capítulo 3. O GeMM possui diversas funcionalidades para a modelagem de PPLI e estas são detalhadas com a apresentação de exemplos.

4.2.1 PROBLEMA DE PROGRAMAÇÃO LINEAR EM SUA FORMA CANÔNICA

Dado o PPL em sua forma canônica é possível identificar a definição dos conjuntos *I* e *J*, que possuem *m* e *n* elementos, respectivamente. Assim, de forma análoga, o PPL em sua forma canônica pode ser escrito da seguinte maneira:

$$Minimizar \quad \sum_{j \in J} c_j . x_j \tag{4}$$

$$Sujeito \quad a \quad \sum_{j \in J} a_{ij} . x_j \geq b_i , \quad \forall \; i \in I = \{1, \ldots , m\} \tag{5}$$

$$x_j \geq 0 \quad \forall \; j \in J = \{1, \ldots , n\} \tag{6}$$

Nesta modelagem, é possível identificar os parâmetros c_j, b_i e a_{ij}. Os parâmetros c_j representam os custos de cada variável x_j, os parâmetros b_i são o segundo membro de cada uma das restrições e os parâmetros a_{ij} são os coeficientes de cada variável x_j nas restrições definidas para todo $i \in I$. Os valores dos parâmetros devem ser informados, pois são dados de entrada do modelo. Nesta modelagem, eles são indexados pelos elementos dos conjuntos *I* e *J*. No PPL em sua forma canônica pode-se identificar também as variáveis de decisão não negativas x_j e as restrições, uma para cada $i \in I$. Por fim, a função objetivo deve ser modelada, que neste caso consiste em minimizar a soma dos produtos $c_j.x_j$ para todo $j \in J$.

Para fazer a modelagem no GeMM desse problema de programação linear, o primeiro passo é criar um projeto de modelagem. Neste momento, os únicos atributos necessários são o nome e a descrição do projeto de modelagem, como mostra a Figura 4.1.

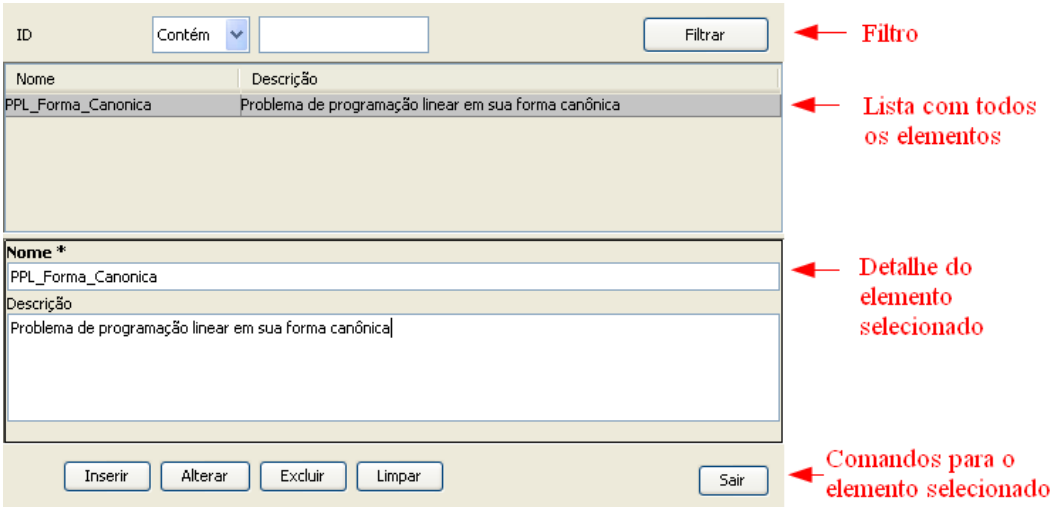


Figura 4.1: Projeto de modelagem do PPL em sua forma canônica

As telas do GeMM possuem formato padronizado, semelhante ao da tela mostrada na Figura 4.1. Na parte superior existem campos para filtrar os elementos na lista, que aparece logo abaixo e contém os elementos do mesmo tipo. Ao selecionar um elemento na lista, o formulário apresenta os seus detalhes para consulta ou edição. Através do formulário também é possível inserir novos elementos. Os campos marcados em negrito e com um asterisco são obrigatórios, como o campo *Nome* da Figura 4.1. Na parte inferior das telas estão os comandos que podem ser aplicados ao elemento selecionado.

A seguir, este modelo é construído com a definição de cada um dos elementos de modelagem. No GeMM a construção do modelo é feita através da criação dos elementos nos formulários do ambiente, no qual cada elemento de modelagem possui um formulário específico para que o usuário possa informar seus atributos. Diferentemente de algumas linguagens de modelagem, como GAMS (BISSCHOP e MEERAUS, 1982) e AMPL (FOURER *et al.*, 1990), em que os modelos matemáticos são criados textualmente através de uma linguagem específica, o GeMM constrói o modelo usando formulários para cada um dos elementos de modelagem.

Normalmente, os primeiros elementos definidos são os conjuntos e seus respectivos índices, uma vez que outros elementos de modelagem, como parâmetros, variáveis e restrições, podem ser indexados pelos índices definidos nos conjuntos. A Figura 4.2 apresenta a modelagem do conjunto *I*. A modelagem do conjunto *J* é feita de forma análoga.

Os atributos de um conjunto são o seu nome, sua descrição, a definição do tipo de dados de seus elementos e os identificadores dos índices. Nesta implementação do GeMM, os tipos de dados possíveis são números inteiros e reais, cadeias de caracteres e data. Como mostra a Figura 4.2, pode-se definir subconjuntos de um determinado conjunto. É importante destacar que, neste momento, apenas a definição dos conjuntos é feita; seus elementos não fazem parte do modelo, mas sim dos cenários de dados, conforme discutido no Capítulo 3.

A Figura 4.3 ilustra a definição dos parâmetros c_j (custos). Os parâmetros possuem um nome e uma descrição. Eles também devem ter seu tipo de dados definido que, nesse caso, além de números inteiros e reais, cadeias de caracteres e data, também podem ser booleanos. Os parâmetros se apresentam de dois tipos: primários, quando seus valores são definidos diretamente nos cenários de dados, ou seja, são dados de entrada; ou calculados, quando seus valores são avaliados a partir de expressões, que podem ter constantes e outros parâmetros. Também devem ser definidos os índices dos conjuntos que indexam o parâmetro.

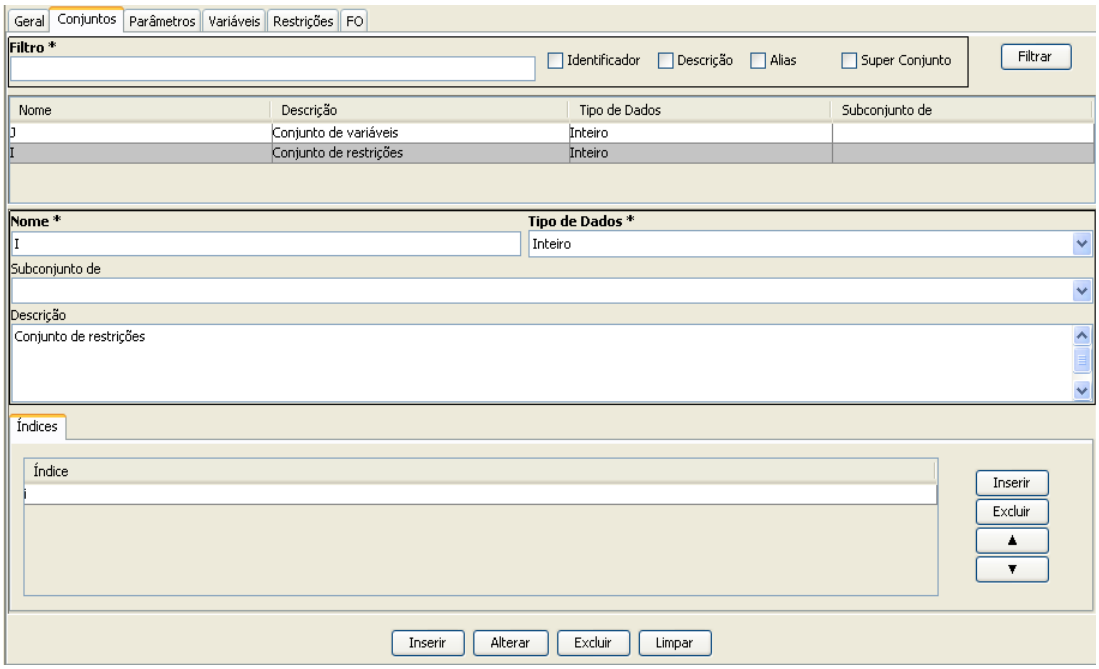


Figura 4.2: Conjuntos e índices do PPL em sua forma canônica

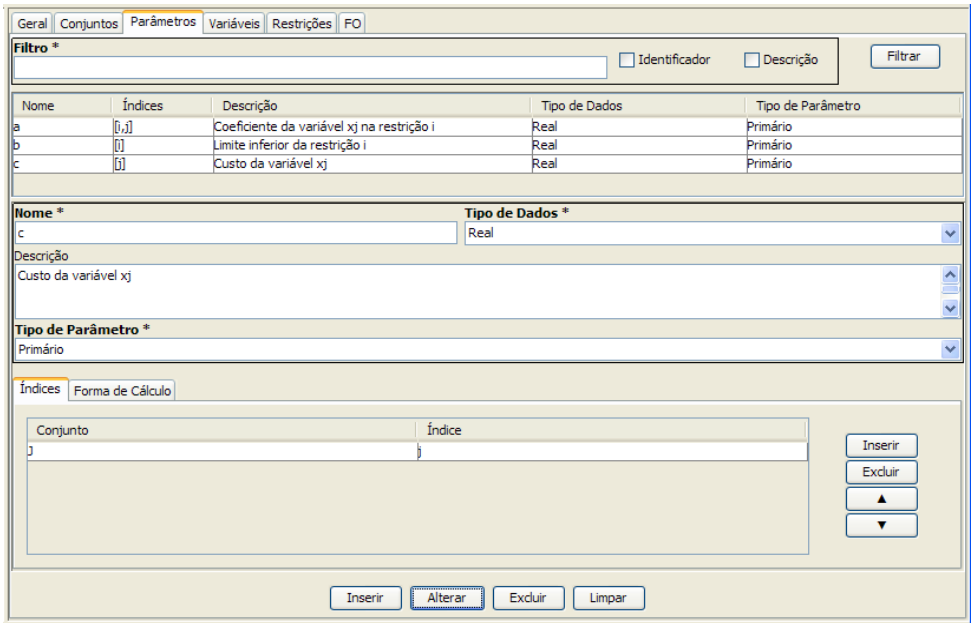


Figura 4.3: Parâmetros do PPL em sua forma canônica

Os operadores aritméticos possíveis nas expressões dos parâmetros calculados são mostrados na Tabela 4.1. A análise das expressões no GeMM é feita com o auxílio da biblioteca de código aberto JEP (JEP JAVA, 2011). Ela implementa os operadores da Tabela 4.1 e permite a adição de novas funções através da criação de uma classe em Java que implemente as interfaces definidas pelo JEP.

Tabela 4.1: Operadores aritméticos

Operador	Descrição
+	Soma
-	Subtração
%	Módulo
/	Divisão
*	Multiplicação
^	Potência
( )	Parênteses

As variáveis são declaradas no GeMM da mesma forma que os parâmetros, podem ser indexadas pelos índices dos conjuntos e ainda ser do tipo real, inteiro ou binário, como mostra a Figura 4.4.

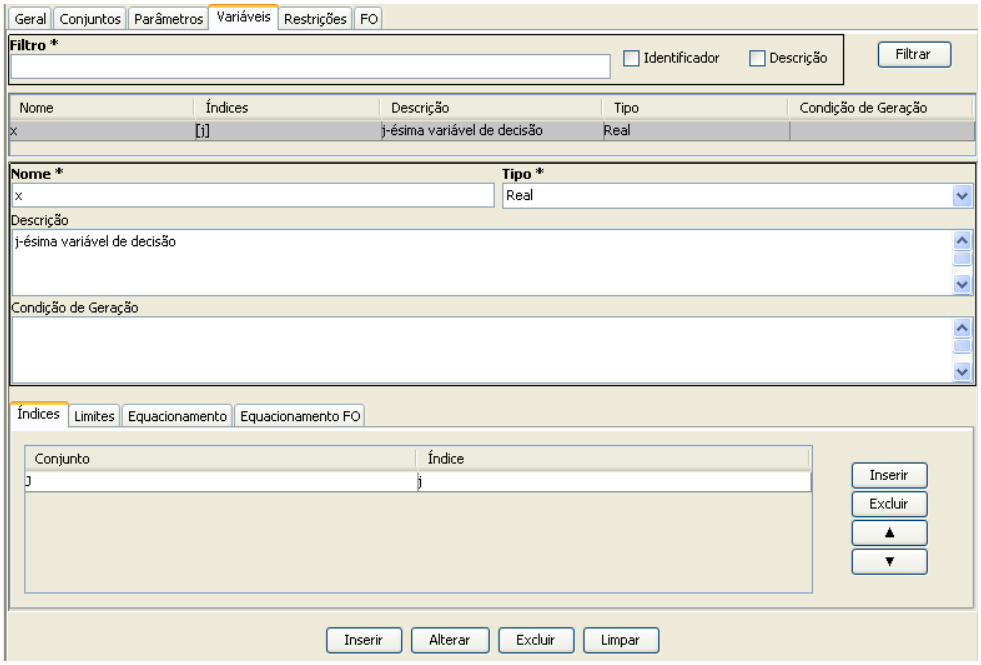


Figura 4.4: Variável do PPL em sua forma canônica

Também devem ser definidos os limites inferior e superior de cada variável. No caso do PPL em sua forma canônica, a variável x_j deve ser maior ou igual a zero para todo j, como destacado na Figura 4.5.

No GeMM, os atributos principais de uma restrição são a descrição e os índices. Na expressão (5) deste exemplo existe uma restrição associada a cada $i \in I$, logo i indexa as restrições do problema, como a Figura 4.6 mostra em destaque. O GeMM exige que uma restrição seja descrita pelo seu lado esquerdo (LHS), que envolve apenas variáveis; e pelo seu

lado direito (RHS), que é descrito por uma expressão invariante. Na restrição (5) do exemplo apresentado, o LHS é dado por $\sum_{j \in J} a_{ij} . x_j$, enquanto o RHS é dado por b_i.

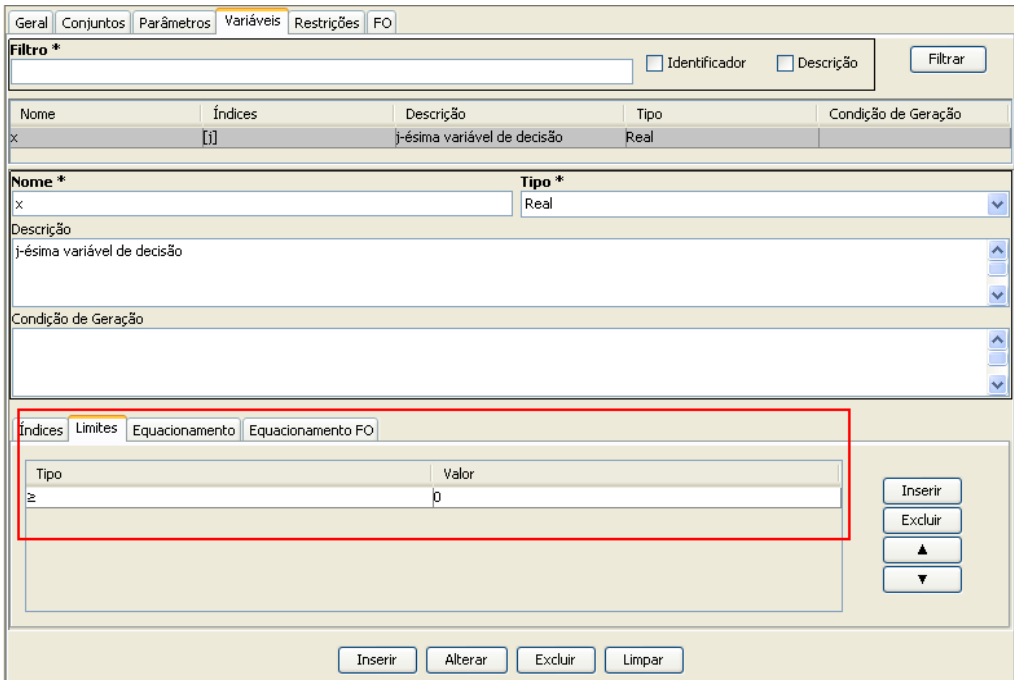


Figura 4.5: Limite inferior da variável do PPL em sua forma canônica

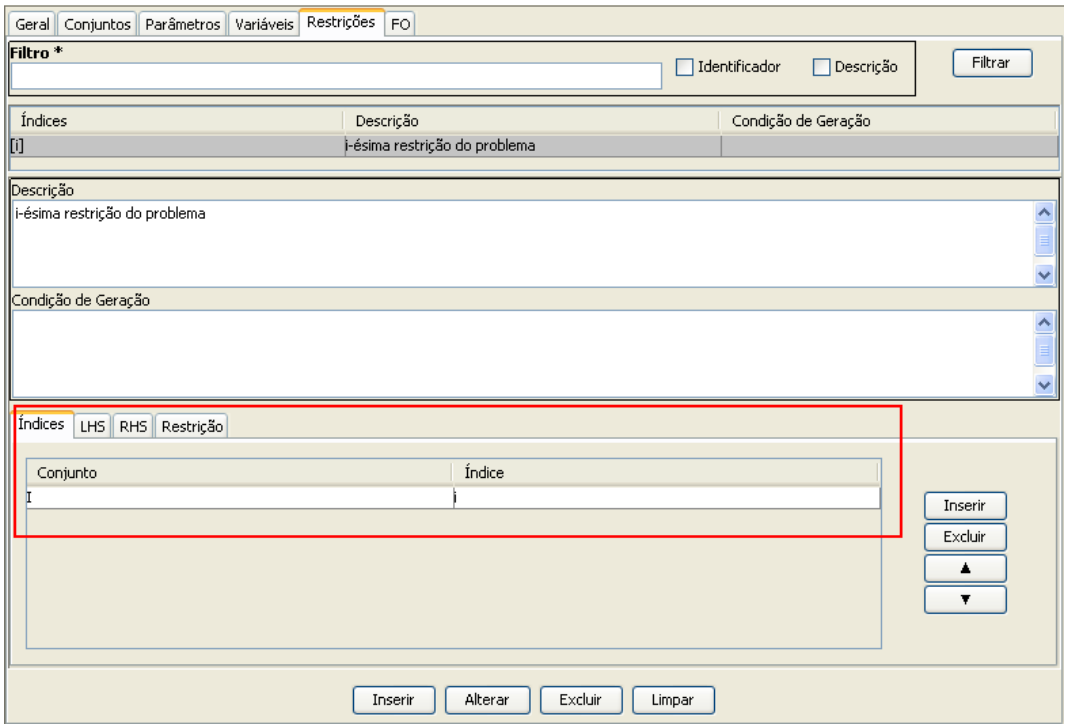


Figura 4.6: Restrições do PPL em sua forma canônica

Tanto o LHS como o RHS podem conter em suas expressões parâmetros primários, parâmetros calculados, constantes e os operadores da Tabela 4.1. A Figura 4.7 apresenta o LHS da restrição (5), no qual a variável x_j entra no equacionamento da restrição com o coeficiente a_{ij}. Não é preciso informar ao GeMM que existe um somatório sobre o índice *j*. Uma vez que a restrição (5) existe para todo *i*, está subentendido que x_j deve ser somado em *j*. Sempre que uma variável entrar no LHS com algum índice não contido nos índices da restrição, o GeMM automaticamente interpreta que esta variável deve ser somada sobre todos os índices que não indexam a restrição, como ocorre com o índice *j* neste exemplo.

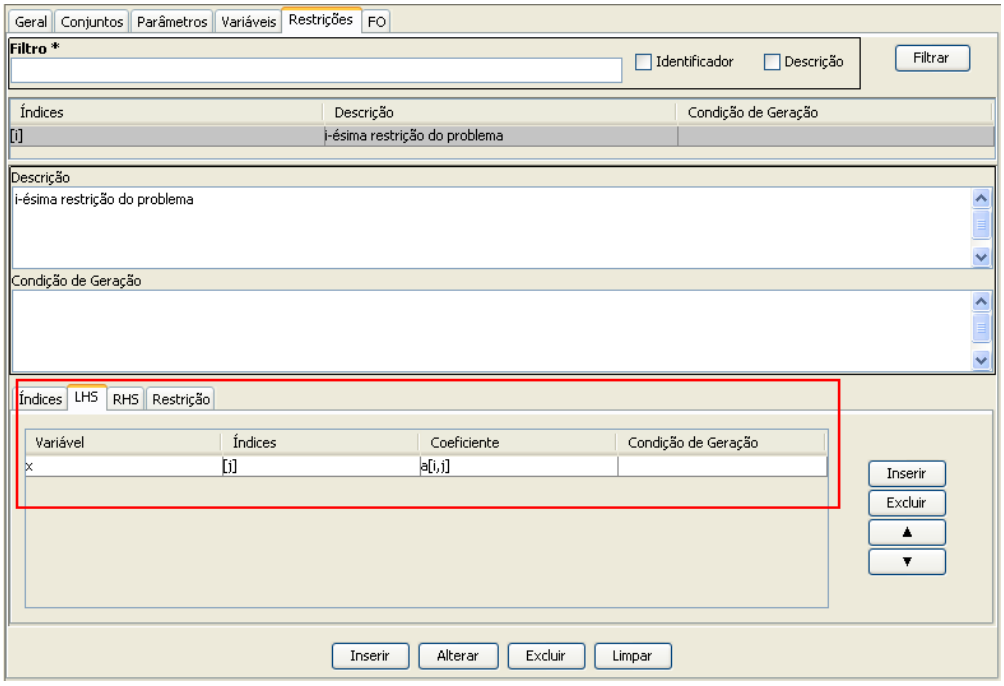


Figura 4.7: Modelagem da restrição do PPL em sua forma canônica

As restrições podem ser de igualdade (=) ou de desigualdade (≤ ou ≥). A Figura 4.8 apresenta o RHS da restrição (5), cujo valor é dado por b_i. Com as informações exibidas nas telas das Figuras 4.6, 4.7 e 4.8, o GeMM gera uma imagem da restrição passando por sua reescrita em LaTeX (LAMPORT, 2011), como mostra a Figura 4.9.

Por fim, a Figura 4.10 apresenta a definição da função objetivo. A função objetivo de maximização ou de minimização deve possuir um nome e uma descrição. A sua expressão é construída da mesma forma que o LHS de uma restrição, não é necessário informar ao GeMM que existe um somatório sobre algum índice na expressão da função objetivo. Como ela é única e não possui índices, o GeMM sempre interpreta que existe um somatório sobre todos

os índices definidos para as variáveis que estão na expressão da função objetivo. Como mostra a Figura 4.10, o GeMM também gera a imagem da sua expressão.

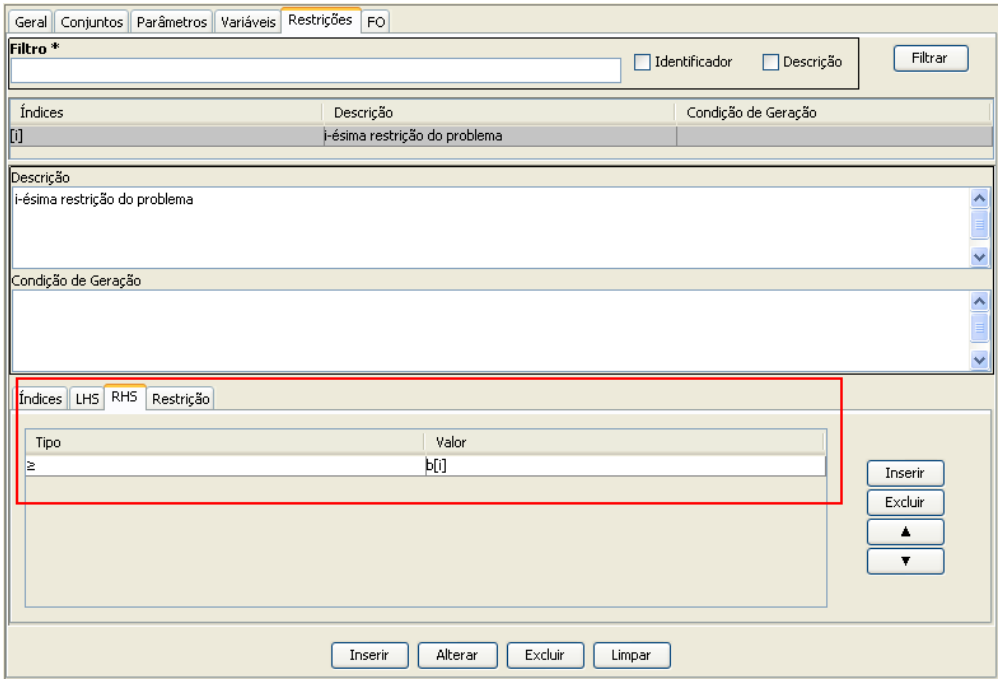


Figura 4.8: RHS da restrição do PPL em sua forma canônica

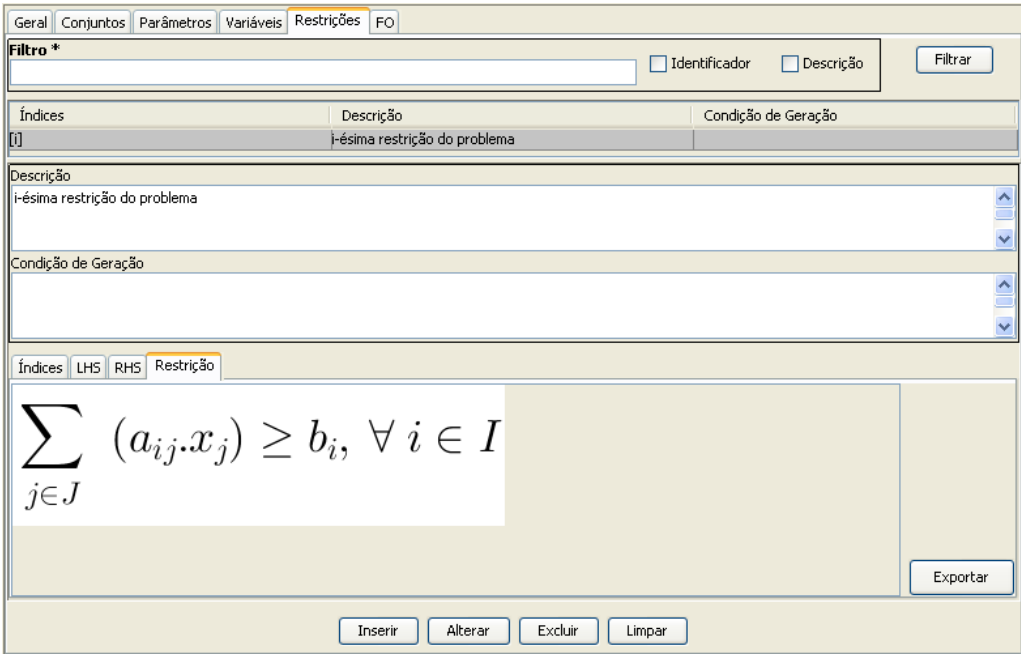


Figura 4.9: Restrições do PPL em sua forma canônica

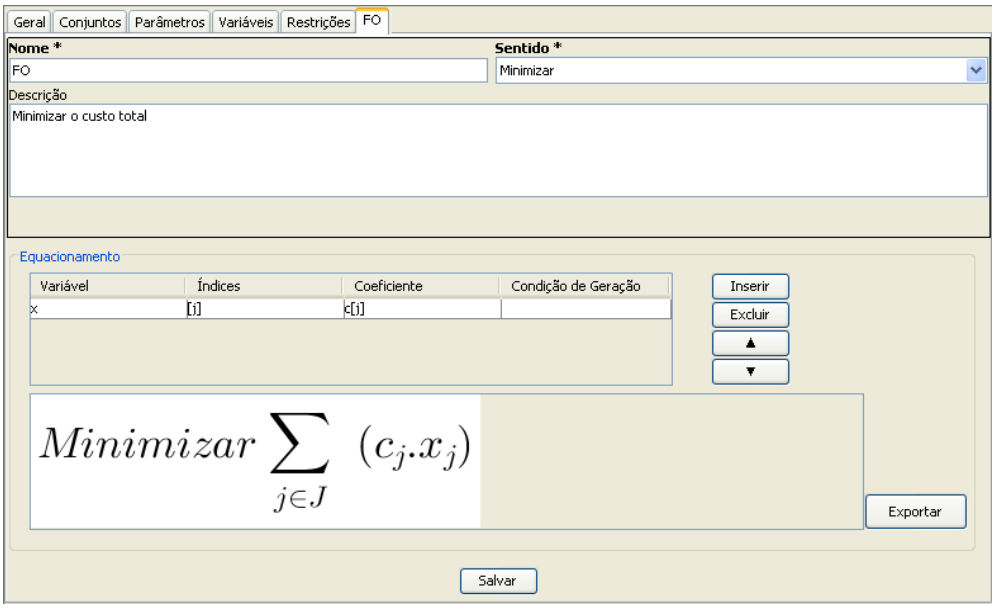


Figura 4.10: Função objetivo do PPL em sua forma canônica

Desta forma, a modelagem pode ser feita por meio de formulários, sem a necessidade do conhecimento de linguagens de modelagem ou de programação. O exemplo do PPL em sua forma canônica é intencionalmente simples para facilitar o aprendizado do ambiente de modelagem. O Capítulo 5 ilustra o uso do GeMM aplicado a modelos mais complexos.

4.2.2 CONDIÇÕES DE GERAÇÃO

As condições de geração são importantes para a construção de modelos reais (FOURER *et al.*, 1990). Para permitir a modelagem de problemas mais complexos do que o exemplo apresentado anteriormente, o GeMM trata as condições de geração de variáveis e restrições. Elas são expressões booleanas que podem usar constantes, parâmetros (primários ou calculados) e valores de índices para determinar a condição em que as variáveis e restrições devem ser geradas na instância do PPLI que é passado para o resolvedor. São permitidos nessas expressões os operadores aritméticos da Tabela 4.1, assim como operadores lógicos e de comparação da Tabela 4.2.

A Figura 4.4 e a Figura 4.6 mostram o campo opcional para a escrita da expressão booleana da condição de geração para uma variável e para uma restrição, respectivamente. O GeMM também permite a uso de condições de geração de variáveis no equacionamento das restrições e função objetivo. A Figura 4.7 mostra o campo opcional para a condição de geração da variável no LHS da restrição com determinado coeficiente. Da mesma forma, a Figura 4.10 mostra o campo para a condição de geração de uma variável na expressão da

função objetivo. O Capítulo 5 apresenta modelos matemáticos que precisam usar condições de geração para sua implementação no GeMM.

Tabela 4.2: Operadores lógicos e de comparação

Comparação		**Lógico**	
Operador	***Descrição***	***Operador***	***Descrição***
>	Maior	!	Negação
>=	Maior ou igual	&&	E
<	Menor	||	Ou
<=	Menor ou igual		
==	Igual		
!=	Diferente		

4.2.3 DOCUMENTAÇÃO DO MODELO

A partir das informações fornecidas pelos usuários no momento da construção do modelo, o GeMM permite a geração automática de um documento em LaTeX com todos os dados fornecidos para o problema modelado. Por utilizar LaTeX na construção do documento, as expressões são todas escritas com símbolos matemáticos, facilitando a validação e a documentação do modelo. Os atributos nome e descrição dos elementos de modelagem nas telas do GeMM permitem que o modelo seja documentado no momento do seu desenvolvimento.

O quadro da Figura 4.11 contém o código LaTeX correspondente ao modelo do exemplo do PPL em sua forma canônica, cuja modelagem no GeMM está apresentada nas Figuras de 4.1 a 4.10. A Figura 4.12 apresenta a imagem do documento gerado a partir deste código.

4.2.4 REPRESENTAÇÃO DO MODELO MATEMÁTICO

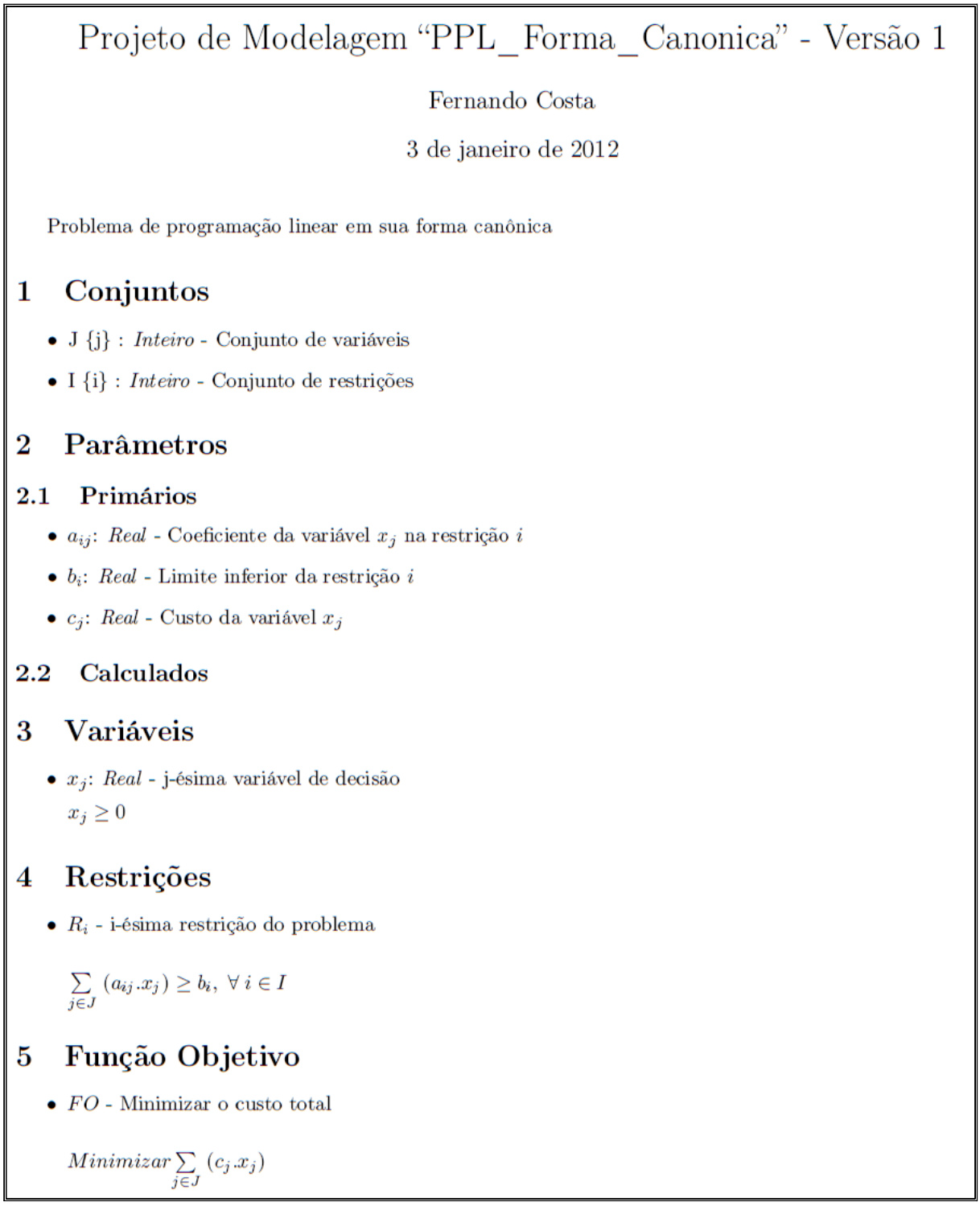
O Capítulo 3 apresenta a estrutura de dados para a representação dos modelos matemáticos. Essa representação é utilizada no GeMM, tanto para o tratamento dos dados na aplicação, quanto para o seu armazenamento em banco de dados. Nesta seção são utilizadas estruturas semelhantes às apresentadas no Capítulo 3, porém com detalhes de implementação. É importante salientar que, para o usuário do GeMM, essas estruturas são totalmente transparentes, pois a interação com o ambiente de modelagem é feita através dos formulários apresentados na Seção 4.2.1.

A Figura 4.13 mostra o diagrama de classes que representa os modelos matemáticos no GeMM. Este diagrama detalha como o GeMM trata e armazena em bancos de dados os

modelos. Para melhor elucidação das estruturas de dados, optou-se por utilizar diagramas de classes da UML (BOOCH *et al.*, 2006) com os tipos de dados da linguagem Java, usada na implementação. Assim, a descrição do tratamento do modelo matemático pelo GeMM é feita por meio de classes, que são equivalentes às tabelas de banco de dados criadas.

```
%% Gerado pelo GeMM
\documentclass[brazil]{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{geometry}
\geometry{verbose,a4paper,headheight=0cm,headsep=0cm,footskip=0cm}
\usepackage{amsmath}
\usepackage{color}
\usepackage{babel}

\begin{document}

\title{Projeto de Modelagem {}``PPL\_Forma\_Canonica'' - Versão 1}
\author{Fernando Costa}
\date{\today}

\maketitle
Problema de programação linear em sua forma canônica

\section{Conjuntos}

\begin{itemize}
\item J \{j\} : \emph{Inteiro} - Conjunto de variáveis
\item I \{i\} : \emph{Inteiro} - Conjunto de restrições
\end{itemize}

\section{Parâmetros}

\subsection{Primários}
\begin{itemize}
\item $a_{ij}$: \emph{Real} - Coeficiente da variável $x_j$ na restrição $i$
\item $b_{i}$: \emph{Real} - Limite inferior da restrição $i$
\item $c_{j}$: \emph{Real} - Custo da variável $x_j$
\end{itemize}

\subsection{Calculados}

\section{Variáveis}

\begin{itemize}
\item $x_{j}$: \emph{Real} - j-ésima variável de decisão
$x_{j} \geq 0$
\end{itemize}

\section{Restrições}

\begin{itemize}
\item $R_{i}$ - i-ésima restrição do problema\\
$\underset{{j\in J}}{\sum}\; (a_{ij}.x_{j})\geq b_{i},\; \forall \; i\in I$
\end{itemize}

\section{Função Objetivo}

\begin{itemize}
\item $FO$ - Minimizar o custo total\\
$Minimizar\underset{{j\in J}}{\sum}\; (c_{j}.x_{j})$
\end{itemize}

\end{document}
```

Figura 4.11: Código em LaTeX



Projeto de Modelagem "PPL_Forma_Canonica" - Versão 1

Fernando Costa

3 de janeiro de 2012

Problema de programação linear em sua forma canônica

1 Conjuntos

- J {j} : *Inteiro* - Conjunto de variáveis
- I {i} : *Inteiro* - Conjunto de restrições

2 Parâmetros

2.1 Primários

- a_{ij}: *Real* - Coeficiente da variável x_j na restrição i
- b_i: *Real* - Limite inferior da restrição i
- c_j: *Real* - Custo da variável x_j

2.2 Calculados

3 Variáveis

- x_j: *Real* - j-ésima variável de decisão

  $x_j \geq 0$

4 Restrições

- R_i - i-ésima restrição do problema

  $\sum_{j \in J} (a_{ij}.x_j) \geq b_i, \ \forall \ i \in I$

5 Função Objetivo

- *FO* - Minimizar o custo total

  $Minimizar \sum_{j \in J} (c_j.x_j)$

Figura 4.12: Documento gerado pelo GeMM do modelo do PPL em sua forma canônica

A classe principal é a *ProjetoModelagem*, que agrega todos os elementos de um modelo. Ela é composta de elementos de modelagem, que são representados pelas classes

ElementoModelagem e *ElementoModelagemVersionado*, cuja diferença é o tratamento de atributos versionados. O controle de versão do GeMM é tratado posteriormente neste capítulo, porém a classe *ElementoModelagemVersionado* é importante nesse momento, uma vez que todos os atributos dos elementos de modelagem têm a versão controlada. Apesar da classe *ElementoModelagem* não possuir atributos, já que todos estes são versionados, ela é importante para identificar os objetos das diferentes versões do mesmo elemento de modelagem. Os atributos básicos de todos os elementos de modelagem são o nome, que deve ser um identificador único, e uma descrição, utilizada para o usuário documentar o modelo.

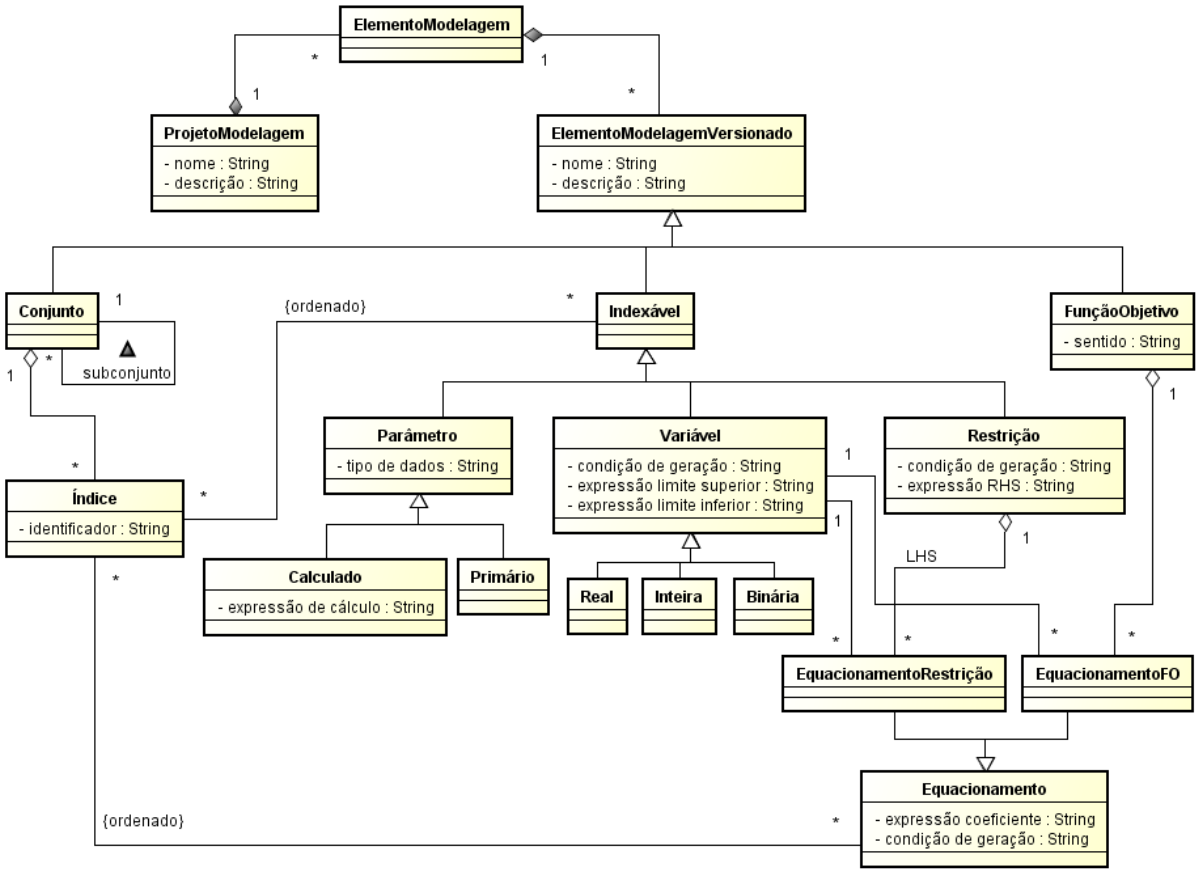


Figura 4.13: Diagrama de classes do modelo matemático

As subclasses de *ElementoModelagemVersionado* são *Conjunto*, *Indexável* e *FunçãoObjetivo*. A classe *Conjunto* representa os conjuntos que podem ser criados nos modelos matemáticos. Cada conjunto pode ter diversos índices, representados pela classe *Índice*. Os índices são utilizados para referenciar um elemento de um determinado conjunto no modelo, logo devem possuir identificadores únicos. A classe *FunçãoObjetivo* possui o atributo *sentido* para determinar se o problema é de maximização ou minimização.

A classe *Indexável* representa os elementos de modelagem que podem receber um *Índice* na sua definição. O relacionamento entre *Indexável* e *Índice* deve ser ordenado, uma vez que a ordem dos índices é importante no momento da avaliação das expressões. Os elementos de modelagem que podem receber índices são: *Parâmetro*, *Variável* e *Restrição*. O diagrama de classes da Figura 4.13 explicita os dois tipos de parâmetros tratados: *Primário* e *Calculado*. Também é possível identificar os três tipos de variáveis tratadas pelo GeMM: *Real*, *Inteira* e *Binária*. A classe *Variável*, além do nome e da descrição, possui as expressões dos limites inferior e superior, e, opcionalmente, uma condição de geração. O terceiro tipo de *Indexável* é a *Restrição*, que possui a expressão do seu RHS e também pode ter uma condição de geração.

Por fim, existem as classes que representam a expressão do LHS das restrições, assim como a da função objetivo, a classe *Equacionamento* e suas derivadas, *EquacionamentoRestrição* e *EquacionamentoFO*. A classe *EquacionamentoRestrição* é aquela que representa os termos do LHS das restrições. Ela referencia um objeto da classe *Variável* e possui a expressão do coeficiente do termo da variável na restrição. Da mesma maneira, a classe *EquacionamentoFO* representa os termos da expressão da *FunçãoObjetivo*. Ela referencia um objeto da classe *Variável* e também possui a expressão do coeficiente da variável na função objetivo. No momento em que o GeMM gera uma instância do PPLI para o resolvedor, tanto o LHS das restrições, quanto a expressão da função objetivo, são interpretados como somatórios dos termos do equacionamento, que são o produto das variáveis com os seus coeficientes. Os coeficientes podem ser expressões que contêm constantes, parâmetros (primários e calculados) e operadores da Tabela 4.1.

Os termos dos equacionamentos também podem ter uma condição de geração, como visto anteriormente, indicando a situação em que determinado termo existe na restrição ou na função objetivo. O relacionamento entre *Equacionamento* e *Índice*, que representa os índices de uma determinada variável no equacionamento do LHS ou da função objetivo, deve indicar a ordem dos índices, uma vez que a ordem dos índices possui significado na modelagem.

4.3 CENÁRIO DE DADOS

De acordo com FOURER *et al.* (1990), os valores dos elementos dos conjuntos e dos parâmetros, que compõem um cenário de dados, devem ser combinados com um modelo matemático para gerar uma instância de um PPLI. O Capítulo 3 discute a separação dos dados e dos modelos. Este capítulo mostra como eles podem ser combinados para criar uma instância de um PPLI que deve ser otimizada por um resolvedor.

4.3.1 PROBLEMA DA DIETA

Um problema conhecido, que pode ser escrito como um PPL em sua forma canônica, é o problema da dieta (BASSI, 1976). Por exemplo, suponha-se uma dieta alimentar restrita a leite desnatado, carne bovina magra, peixe e salada verde, por meio da qual um indivíduo deva segui-la de forma a atender aos requisitos nutricionais de vitaminas A, C e D. De forma simples, o problema da dieta consiste em determinar a quantidade que deve ser consumida de cada um dos alimentos especificados, a fim de minimizar os gastos com eles e atender aos requisitos de ingestão das vitaminas.

Os dados necessários para este problema são: a quantidade de cada vitamina em cada um dos alimentos, o custo de cada um deles e a quantidade mínima diária que um indivíduo deve consumir de cada vitamina para atender aos requisitos nutricionais. Tais valores podem ser vistos na Tabela 4.3.

Tabela 4.3: Dados do problema da dieta

Alimentos / **Vitaminas**	**Leite (litro)**	**Carne (Kg)**	**Peixe (Kg)**	**Salada (500g)**	**Requisitos Mínimos**
A	2 mg	2 mg	4 mg	20 mg	11 mg
C	50 mg	10 mg	50 mg	10 mg	70 mg
D	2 mg	90 mg	40 mg	30 mg	250 mg
Custo	R$ 2,5	R$ 15,7	R$ 10,9	R$ 5,7	

Considera-se como variável de decisão a quantidade x_j a ser consumida de cada alimento, para $j \in \{leite, carne, peixe, salada\}$. O modelo de programação linear pode ser escrito da seguinte forma:

$$Minimizar: \; 2{,}5 . x_{leite} + 15{,}7 . x_{carne} + 10{,}9 . x_{peixe} + 5{,}7 . x_{salada} \tag{7}$$

Sujeito a:

$$2 . x_{leite} + 2 . x_{carne} + 4 . x_{peixe} + 20 . x_{salada} \geq 11 \tag{8}$$

$$50 . x_{leite} + 10 . x_{carne} + 50 . x_{peixe} + 10 . x_{salada} \geq 70 \tag{9}$$

$$2 . x_{leite} + 90 . x_{carne} + 40 . x_{peixe} + 30 . x_{salada} \geq 250 \tag{10}$$

$$x_{leite}, x_{carne}, x_{peixe}, x_{salada} \geq 0 \tag{11}$$

Como o objetivo do problema é minimizar os gastos com os alimentos, a função objetivo é definida pelo somatório dos produtos da quantidade consumida pelo preço de cada

alimento, como na expressão (7). As restrições (8), (9) e (10) se referem ao consumo mínimo de vitaminas para atender às necessidades nutricionais. Além disso, a quantidade consumida dos alimentos deve ser sempre positiva, conforme a restrição (11).

Como este problema da dieta é um caso particular do PPL em sua forma canônica, é possível afirmar que o modelo apresentado nas expressões de (7) a (11) é equivalente ao modelo descrito nas expressões de (4) a (6), desde que sejam explicitados os elementos dos conjuntos I = {*vitamina A, vitamina C, vitamina D*} e J = {*leite, carne, peixe, salada*}, e informados os valores para os parâmetros c_j, b_i *e* a_{ij}, referentes ao custo dos alimentos, aos requisitos mínimos de consumo das vitaminas e à quantidade de cada uma em cada alimento, respectivamente. Estes valores estão na Tabela 4.3.

Logo, o modelo construído no GeMM para o PPL em sua forma canônica também se aplica ao problema da dieta. Uma vez pronto o modelo matemático, o GeMM cria automaticamente a estrutura de dados para armazenar as informações e as telas para entrada dos dados e visualização dos resultados. A Figura 4.14 apresenta as telas geradas pelo GeMM para o modelo do PPL em sua forma canônica. Nessas telas, é possível ver os dados dos conjuntos I e J referentes ao cenário de dados do problema da dieta.

As Figuras 4.15, 4.16 e 4.17 mostram as telas para entrada dos valores dos parâmetros primários. Na Figura 4.15, estão os valores do parâmetro c_j, custo de cada um dos alimentos. Nesta tela existe uma caixa de seleção para escolher o elemento do conjunto e uma caixa de texto para edição do valor do parâmetro, isso porque o parâmetro c_j é indexado pelos elementos do conjunto J. Da mesma forma, a Figura 4.16 apresenta a tela para entrada do parâmetro b_i, os requisitos mínimos de cada uma das vitaminas.

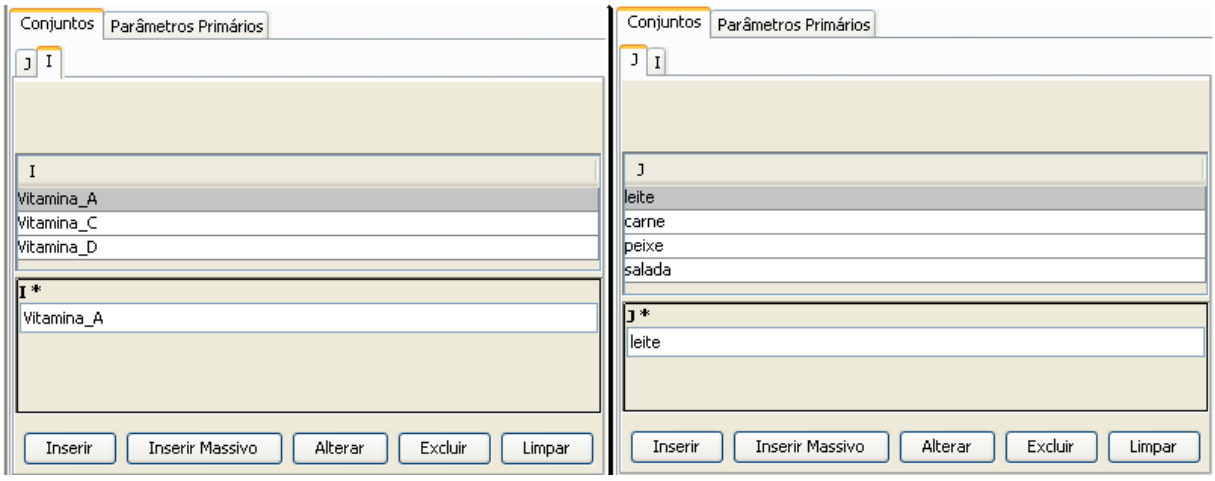


Figura 4.14: Dados dos conjuntos do problema da dieta

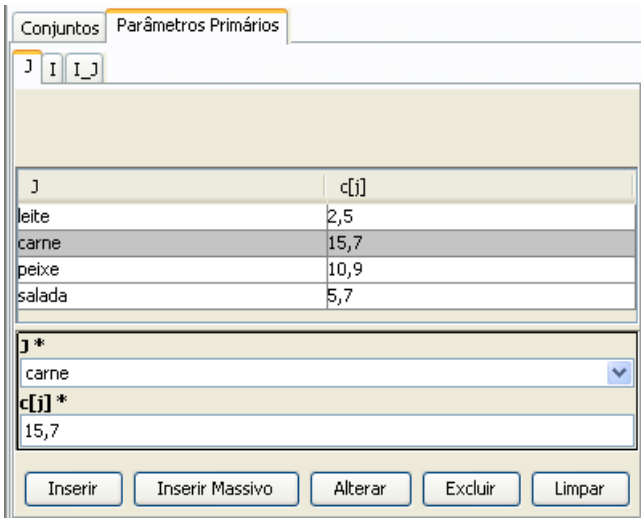


Figura 4.15: Custo dos alimentos do problema da dieta

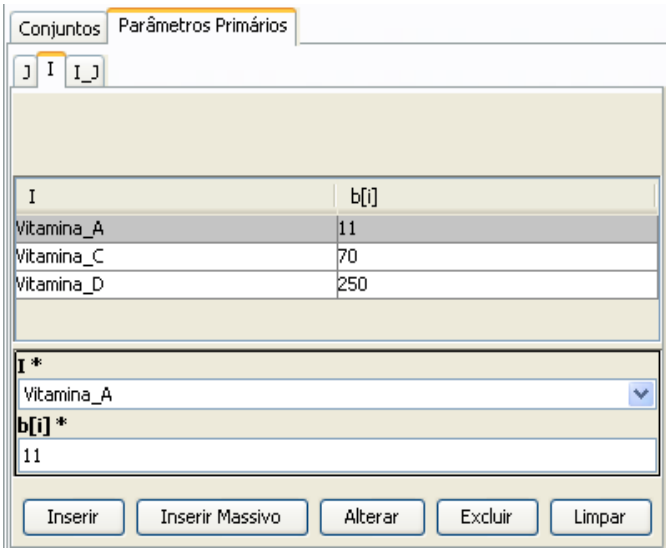


Figura 4.16: Requisitos mínimos de consumo das vitaminas do problema da dieta

Por fim, a Figura 4.17 mostra a tela gerada para entrada dos valores do parâmetro a_{ij}, que representa a quantidade de cada vitamina contida em cada alimento. Nesta tela é possível observar duas caixas de seleção para a escolha dos elementos dos conjuntos *I* e *J*, cujos índices indexam o parâmetro, e a caixa de texto para edição dos valores do parâmetro.

Com as telas geradas pelo GeMM a partir do modelo, é possível fornecer os dados necessários para que se crie uma instância do problema e esta seja transferida para um resolvedor. Nas situações em que muitos dados devem ser informados, o GeMM permite a inclusão massiva destes através do comando "Inserir Massivo", que pode ser visto nas Figuras

4.14, 4.15, 4.16 e 4.17. Através deste comando, os dados são importados a partir de arquivos de texto ou de planilhas eletrônicas.

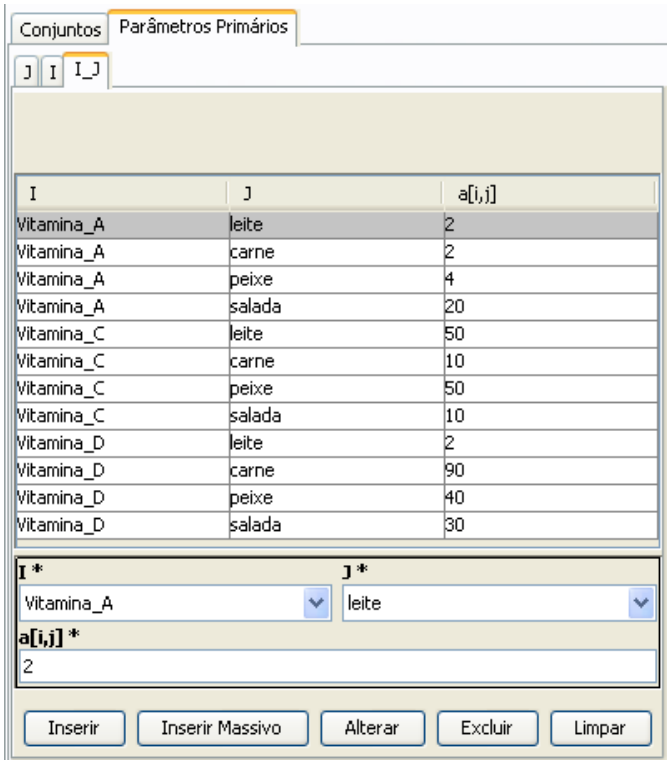


Figura 4.17: Quantidade de cada vitamina em cada alimento do problema da dieta

O GeMM também gera as telas para visualização dos resultados. Este exemplo do problema da dieta foi resolvido pelo CPLEX (IBM CORPORATION, 2010) e a sua solução pode ser vista nas imagens das telas das Figuras 4.18, 4.19 e 4.20. A Figura 4.18 mostra a tela de resumo da execução com o valor da função objetivo, a data de execução e o estado da otimização, que neste caso é "Ótimo", pois o resolvedor encontrou uma solução viável ótima para o problema exemplificado.

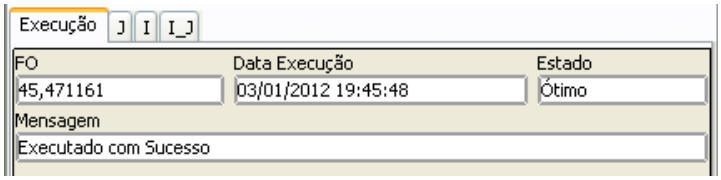
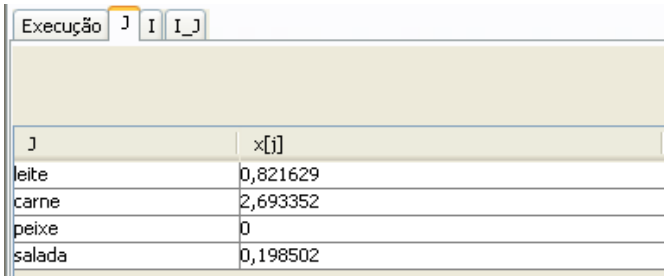


Figura 4.18: Resumo do resultado do problema da dieta

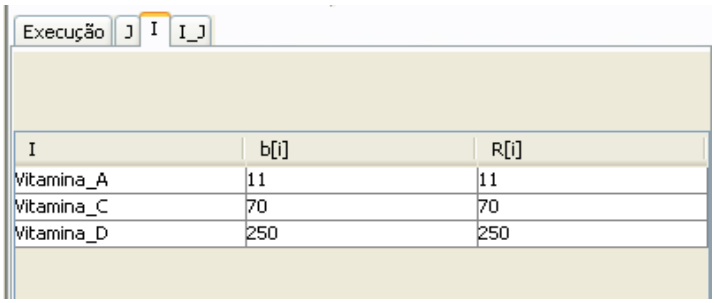
A Figura 4.19 apresenta os valores que devem ser consumidos de cada um dos alimentos, ou seja, os valores ótimos das variáveis x_j para $j \in J$.



J	x[j]
leite	0,821629
carne	2,693352
peixe	0
salada	0,198502

Figura 4.19: Resultado das variáveis do problema da dieta

Finalmente, a Figura 4.20 apresenta os valores calculados para os LHS's das restrições (8), (9) e (10). Comparando com os valores dos RHS's dessas restrições, é possível concluir que todas elas estão ativas.



I	b[i]	R[i]
Vitamina_A	11	11
Vitamina_C	70	70
Vitamina_D	250	250

Figura 4.20: Resultado das restrições do problema da dieta

4.3.2 REPRESENTAÇÃO DOS CENÁRIOS DE DADOS

Para tratar os cenários de dados, projetou-se uma estrutura capaz de atender qualquer modelo que possa ser elaborado pelo GeMM. A Figura 4.21 apresenta o diagrama de classes para representação dos cenários de dados. Esse diagrama de classes mostra em detalhes como o GeMM implementa o tratamento dos cenários de dados e como eles são persistidos em um banco de dados. Assim como foi feito para a representação dos modelos matemáticos, utiliza-se diagramas de classes da UML com os tipos de dados da linguagem Java para detalhar a implementação.

Conforme descrito no Capítulo 3, utiliza-se o conceito da metamodelagem para tratar os cenários de dados dos modelos matemáticos, uma vez que cada modelo diferente precisa dos seus dados em uma estrutura específica. Assim, com a utilização da metamodelagem, é possível ter um único esquema de banco de dados para atender qualquer modelo matemático de PPLI.

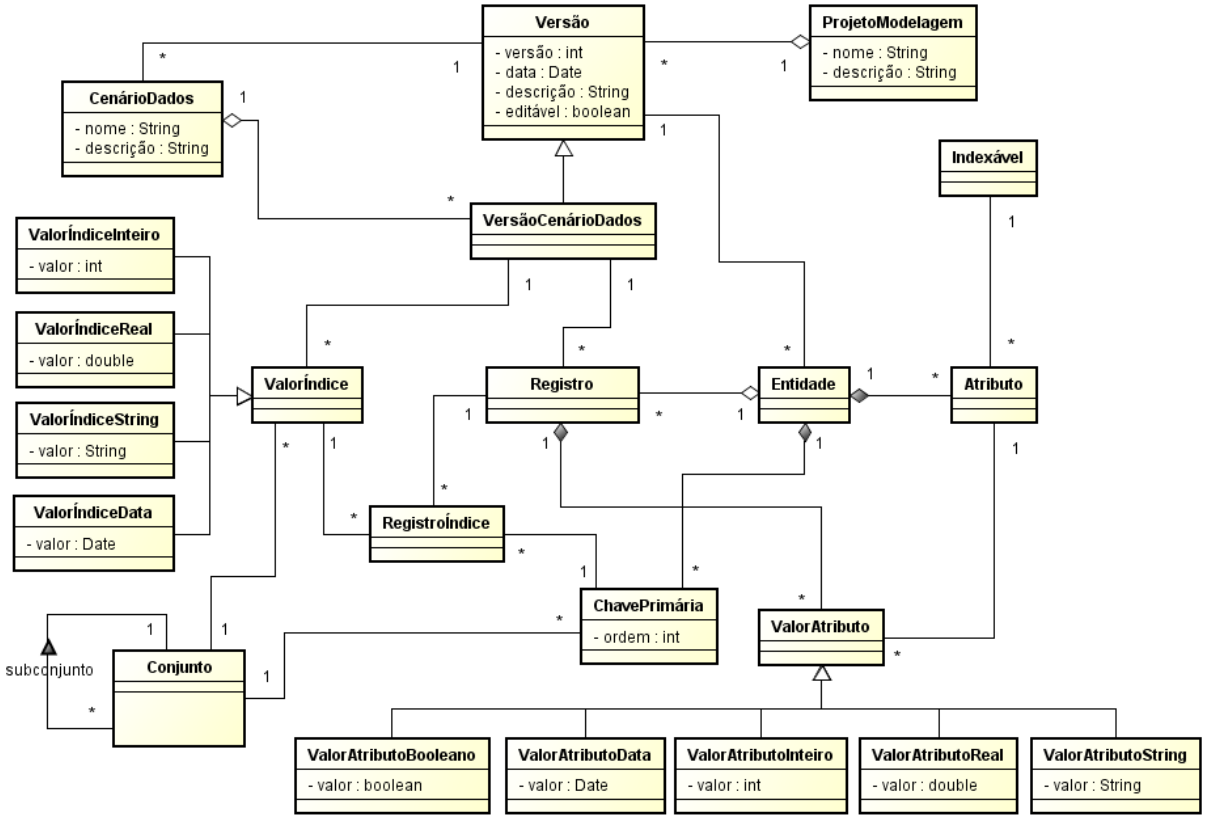


Figura 4.21: Diagrama de classes dos cenários de dados

As classes principais são a *CenárioDados* e a *VersãoCenárioDados*. Apesar do versionamento dos cenários de dados ser descrito posteriormente neste capítulo, é importante o conhecimento da classe *VersãoCenárioDados*. A *CenárioDados* representa um cenário específico criado por um usuário e possui apenas um nome e uma descrição. Um cenário de dados pertence a uma determinada versão do projeto de modelagem no GeMM. Cada um deles pode ter várias versões, que são representadas pela classe *VersãoCenárioDados*.

A classe *ValorÍndice* representa os valores possíveis para os índices, ou seja, são os elementos pertencentes aos conjuntos. Os valores dos elementos dos conjuntos estão associados a uma versão específica de um cenário de dados. Suas subclasses *ValorÍndiceInteiro*, *ValorÍndiceReal*, *ValorÍndiceString* e *ValorÍndiceData* são usadas para implementar o tipo de dados definido para o conjunto. As classes *Entidade*, *ChavePrimária* e *Atributo* permitem fazer a ligação do modelo matemático com a estrutura de dados metamodelada. Estas não possuem relação direta com as informações dos cenários de dados, mas representam a estrutura de dados metamodelada para armazenar seus dados. Cada combinação diferente dos índices que indexam os elementos de modelagem, deve ter um objeto da classe *Entidade* criado. No exemplo do modelo do PPL em sua forma canônica,

descrito na Seção 4.2.1, existem três diferentes entidades: a entidade dos elementos de modelagem indexados somente por *i*, dos indexados somente por *j* e dos indexados por *i* e *j*, como mostrado no Capítulo 3.

A classe *ChavePrimária* indica quais os índices de uma entidade formam a sua chave primária. Ela possui um atributo *ordem*, uma vez que a ordem dos índices nos elementos de modelagem é importante. Elementos de modelagem que são indexados pelos mesmos índices, porém em uma ordem diferente, devem estar associados a entidades diferentes. A classe *ChavePrimária* relaciona a classe *Entidade* com a classe *Conjunto*. A classe *Atributo* representa o próprio elemento de modelagem, que é indexado pela combinação de índices específica da entidade a que pertence. Assim, *Atributo* tem relação com um determinado *Indexável*, cujas subclasses são *Restrição*, *Variável* e *Parâmetro*, e com *Entidade*. A classe *Atributo* é análoga a uma coluna de uma tabela de banco de dados.

No exemplo do PPL em sua forma canônica, os atributos da entidade dos elementos de modelagem indexados somente por *i* são o parâmetro b_i e a restrição (5), e a chave primária desta entidade é o índice *i*; os atributos da entidade dos elementos de modelagem indexados somente por *j* são o parâmetro c_j e a variável x_j, e a chave primária desta entidade é o índice *j*; por fim, o atributo da entidade dos elementos de modelagem indexados por *i* e *j* é o parâmetro a_{ij}, e a chave primária desta entidade é dada pelos índices *i* e *j*.

Os valores dos atributos das entidades metamodeladas, ou seja, os valores dos parâmetros, variáveis e do LHS das restrições avaliadas, são tratados pela classe *ValorAtributo* e suas subclasses, *ValorAtributoBooleano*, *ValorAtributoData*, *ValorAtributoInteiro*, *ValorAtributoReal* e *ValorAtributoString*. As subclasses de *ValorAtributo* são usadas de acordo com o tipo de dado definido para os parâmetros ou o tipo das variáveis e das restrições. A classe *Registro* é usada para agrupar todos os valores de atributos que são indexados pelos mesmos valores de índices em uma entidade. Um objeto dessa classe representa um registro de banco de dados, ou seja, uma linha de uma tabela. A classe *RegistroÍndice* é um relacionamento ternário entre *Registro*, *ValorÍndice* e *ChavePrimária,* utilizada para armazenar quais são os valores de índices que identificam unicamente um determinado registro de uma entidade.

As classes *ValorÍndice*, *Registro*, *RegistroÍndice* e *ValorAtributo* do diagrama de classes da Figura 4.21 estão relacionadas diretamente com uma versão de um cenário de dados. Os objetos dessas classes variam com os diferentes cenários de dados e suas versões

para um mesmo modelo matemático. Já as classes *Entidade*, *ChavePrimária* e *Atributo* representam a estrutura de dados metamodelada para tratar os cenários de dados do modelo. Sendo assim, os objetos dessas classes representam a estrutura de dados de um determinado modelo matemático. Eles não mudam se o modelo matemático não for alterado, são os mesmos objetos para tratar qualquer cenário de dados.

Uma vez pronto o modelo matemático, o GeMM gera classes que representam o modelo de dados referente ao modelo matemático, que está metamodelado pelas classes *Entidade*, *ChavePrimária* e *Atributo* do diagrama da Figura 4.21. Essas classes são usadas para tratar os dados nos formulários gerados para os cenários de dados dos modelos matemáticos. Na Figura 4.22 está o diagrama de classes que representa a estrutura para tratar os dados do modelo do PPL em sua forma canônica e as Figuras 4.14, 4.15, 4.16, 4.17, 4.19 e 4.20 apresentam as telas geradas.

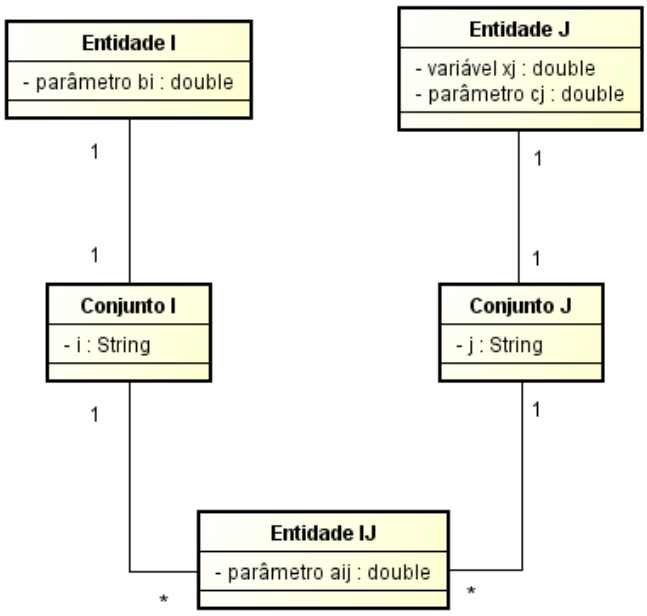


Figura 4.22: Diagrama de classes das entidades do PPL em sua forma canônica

4.4 CONTROLE DE VERSÃO

Além de permitir a modelagem de PPLI e gerar as telas e estruturas de dados para tratar os cenários de dados dos respectivos problemas de otimização, o GeMM provê a funcionalidade de controle de versão. Conforme visto no Capítulo 3, o controle de versão é aplicado tanto nos modelos matemáticos quanto nos cenários de dados.

4.4.1 CONTROLE DE VERSÃO DOS MODELOS MATEMÁTICOS

No controle de versão do GeMM para os modelos matemáticos, o versionamento é tratado em granularidade fina e é baseado nos conceitos do domínio em questão, a modelagem matemática. A granularidade fina permite que cada elemento de modelagem tenha a sua própria versão. A classe *ElementoModelagemVersionado,* que pode ser vista no diagrama de classes da Figura 4.23, representa os atributos dos elementos de modelagem que estão sob controle de versão. Já a classe *ElementoModelagem* representa um determinado elemento de modelagem em um modelo matemático, independentemente da sua versão. Sua função é agrupar as versões referentes ao mesmo elemento de modelagem, uma vez que todos os atributos estão sob controle de versão. Assim, a troca de nome de um elemento, por exemplo, é considerada uma alteração e não uma exclusão e criação de um novo elemento.

O diagrama de classes da Figura 4.23 representa a estrutura de dados para fazer o controle de versão dos modelos matemáticos. Como visto anteriormente, a classe *ProjetoModelagem* representa a identificação do problema de otimização que se quer resolver. Ela possui como atributos um nome e uma descrição. As informações do modelo matemático propriamente dito estão associadas a uma versão do projeto de modelagem, representada pela classe *Versão*. Cada versão está associada a um usuário do GeMM. A classe *ElementoModelagemVersionado* possui dois relacionamentos com a classe *Versão*: um que indica a versão do elemento de modelagem e outro que indica em quais as versões do projeto de modelagem o elemento está presente. Desta forma, o GeMM implementa a granularidade do controle de versão no elemento de modelagem, já que cada elemento dentro de um projeto de modelagem possui a sua própria versão. A associação nomeada "*elemento na versão do projeto"* no diagrama da Figura 4.23 indica quais as versões dos elementos de modelagem estão em uma versão do projeto de modelagem. A versão do elemento de modelagem é representada pela associação "*versão do elemento"*, no mesmo diagrama. Assim, dada uma versão do projeto de modelagem, o GeMM consegue identificar o modelo matemático associado e permite a criação de cenários de dados para ele.

O GeMM implementa o controle de concorrência sobre os modelos matemáticos ao utilizar o atributo *editável* da classe *Versão*. Este indica quando a versão de um projeto de modelagem pode ou não ser editada. Se puder, a alteração só pode ser feita pelo usuário que criou a versão. Qualquer outro usuário não pode editar esse projeto, até que o atual libere ou descarte a sua versão. Cada projeto de modelagem só pode possuir uma versão editável, uma vez que o GeMM não trata a edição em paralelo do mesmo projeto de modelagem. Quando o

projeto de modelagem não possuir nenhuma versão editável, qualquer usuário pode criar uma nova versão e fazer a sua edição. Se esta for a primeira, o GeMM cria uma versão vazia, senão, cria uma nova associada com os elementos de modelagem da versão anterior.

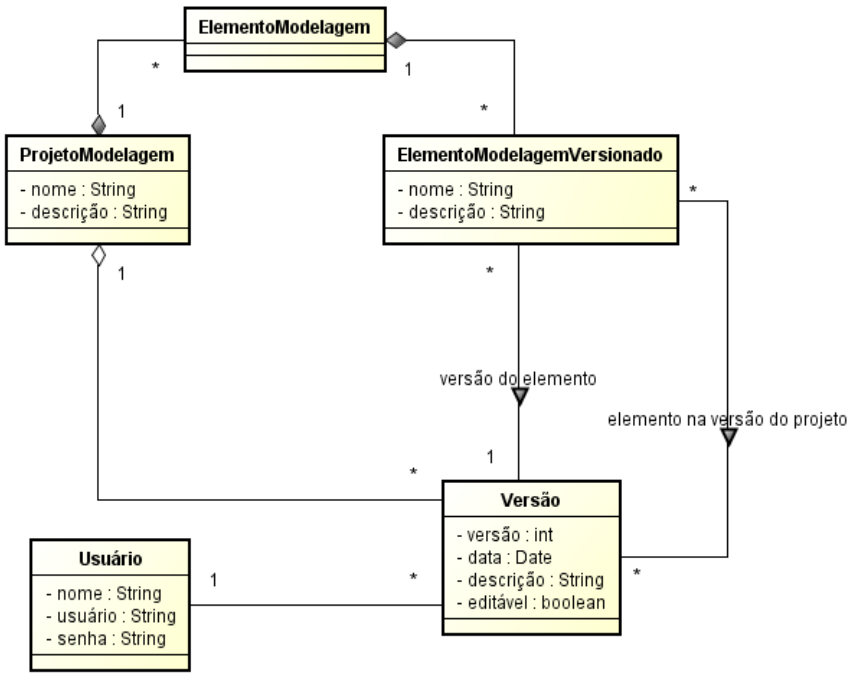


Figura 4.23: Diagrama de classes do controle de versão dos modelos matemáticos

A Figura 4.24 mostra a tela do GeMM para o versionamento do modelo matemático do exemplo do PPL em sua forma canônica, descrito anteriormente neste capítulo. Nesta figura é possível ver que o modelo está em sua primeira versão, está em *Edição* e que todas as operações feitas foram inclusões dos elementos de modelagem que representam esse modelo matemático.

A Figura 4.25 mostra a tela para edição de uma versão do projeto de modelagem, neste caso, do modelo do exemplo do PPL em sua forma canônica. À esquerda desta figura, pode-se ver uma árvore com a visão geral desta versão do modelo, contendo todos os seus elementos de modelagem e suas respectivas versões individuais. É importante destacar que as versões dos elementos de modelagem devem ser iguais ou menores que a versão do projeto de modelagem, uma vez que um novo elemento de modelagem é criado com a versão atual do projeto de modelagem. Esta estratégia também é adotada em sistemas de controle de versão convencionais, como o Subversion (THE APACHE SOFTWARE FOUNDATION, 2011).

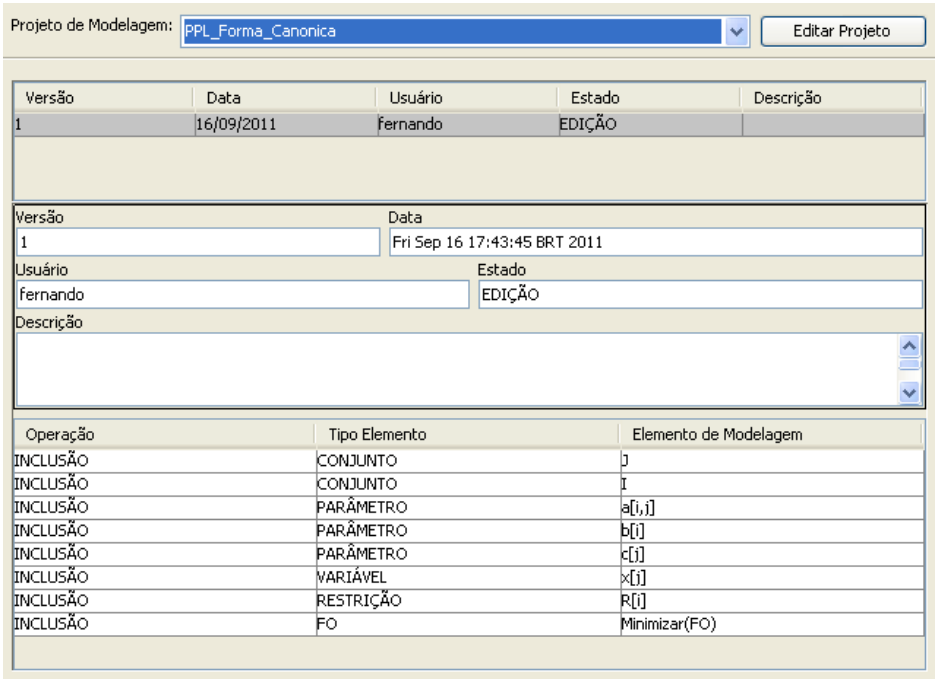


Figura 4.24: Versionamento do projeto de modelagem do PPL em sua forma canônica

Na parte inferior da Figura 4.25 existem dois comandos: liberar e descartar a versão. No caso da ação liberar, o GeMM entende que esta versão está finalizada e o atributo editável fica igual a falso, indicando que qualquer usuário pode editar o projeto criando a versão seguinte. Assim, a versão que foi liberada fica imutável. Em sistemas de controle de versão convencionais, essa ação é semelhante ao comando de *checkin* ou *commit*. Na ação descartar, o GeMM remove a versão do projeto de modelagem, ou seja, apaga o objeto da classe *Versão*. Essa ação só pode ser aplicada em versões que estão em edição e pelo próprio usuário que as criou. As versões dos projetos de modelagem que foram liberadas não podem mais ser alteradas, descartadas ou removidas, garantindo o armazenamento do histórico de alterações de um modelo matemático. Sempre é possível fazer modificações na modelagem com a criação de uma nova versão, semelhante à última versão liberada.

A tela do GeMM da Figura 4.25 é a tela para edição e criação dos modelos matemáticos. As telas mostradas nas Figuras 4.2, 4.3, 4.4, 4.6 e 4.10 são respectivamente as abas *Conjuntos*, *Parâmetros*, *Variáveis*, *Restrições* e *FO* (Função Objetivo) da Figura 4.25.

4.4.2 CONTROLE DE VERSÃO DOS CENÁRIOS DE DADOS

O GeMM também implementa o controle de versão dos cenários de dados que podem ser criados para os modelos matemáticos. O diagrama de classes da Figura 4.26 representa a estrutura de dados que implementa o versionamento dos cenários de dados. A classe

VersãoCenárioDados representa uma versão de um determinado cenário de dados. Da mesma forma que é feita para os modelos matemáticos, todas as informações contidas nos cenários de dados estão associadas a uma determinada versão. Neste caso, a granularidade do controle de versão é o próprio cenário de dados, logo os dados dentro de um cenário não possuem versão independente, estando associados sempre a uma determinada versão de um cenário de dados. A classe *CenárioDados* representa apenas a identificação de um cenário do modelo matemático. Seus atributos são apenas nome e descrição. Analogamente, a classe *VersãoCenárioDados* está para *CenárioDados*, assim como *Versão* está para a classe *ProjetoModelagem*.

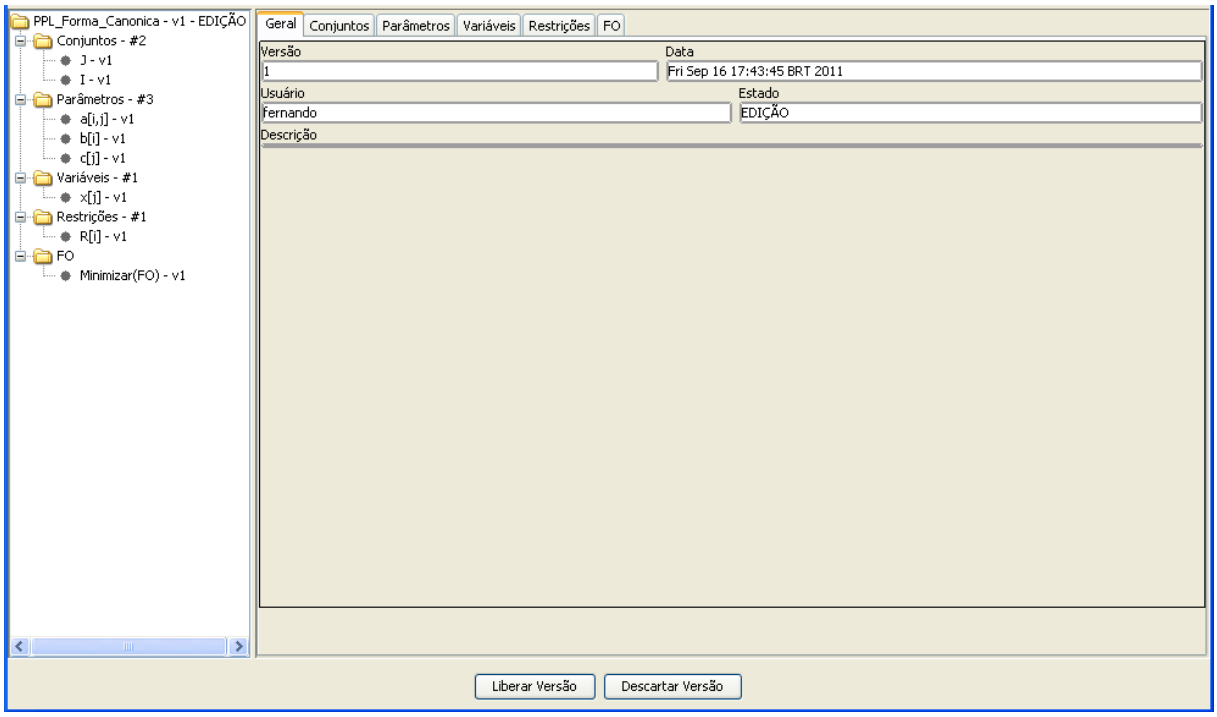


Figura 4.25: Visão geral da primeira versão do modelo do PPL em sua forma canônica

O controle de versão e concorrência é feito da mesma forma dos modelos matemáticos. O atributo *editável* na classe *VersãoCenárioDados* indica quando uma versão do cenário de dados pode ser editada ou não. Quando o valor deste atributo é verdadeiro, apenas o usuário que criou a versão pode editá-la, liberá-la ou removê-la. Quando o usuário libera a versão do cenário de dados, semelhante ao *checkin* ou *commit* de sistemas de controle de versão convencionais, outro usuário pode editá-lo, criando uma nova versão do cenário. Para editar as informações do cenário de dados, sempre é criada uma nova versão para ele. Se o cenário ainda não possui versão, ao editá-lo é criada uma versão vazia, sem informação, para que o usuário informe todos os dados. Se o cenário de dados já possuir alguma versão, é

criada uma nova versão semelhante à última existente. A remoção das versões dos cenários de dados só pode ser feita pelo usuário que as criou e só enquanto estiver em edição, pois uma vez que ela é liberada não pode ser mais excluída ou alterada, mantendo assim o histórico dos dados associados aos modelos matemáticos. Desta forma, é possível concluir que apenas a última versão pode ser removida ou alterada, se ela já não tiver sido liberada.

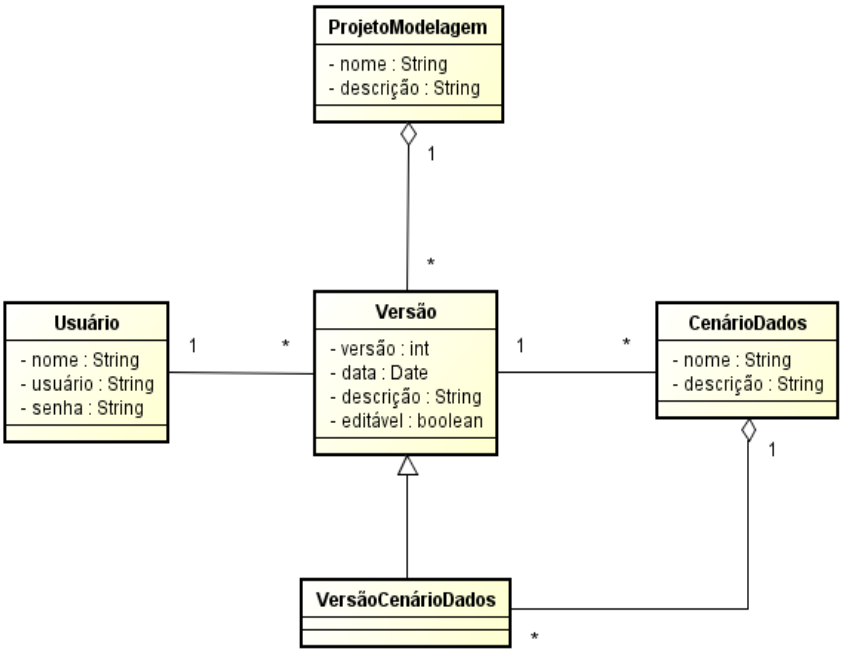


Figura 4.26: Diagrama de classes do controle de versão dos cenários de dados

A relação dos cenários de dados, dos projetos de modelagem e de suas versões pode ser vista na Figura 4.27. Um determinado cenário de dados está associado diretamente a uma versão de um projeto de modelagem, que por sua vez está associada a um projeto de modelagem. Uma versão de um cenário de dados está associada a apenas um cenário de dados. Para facilitar o uso, o GeMM possui funcionalidades para duplicar os dados e criar novos cenários de dados a partir de versões de cenários já existentes, assim como possibilita a importação e exportação dos dados a partir de fontes externas, como arquivos de texto.

A tela do GeMM mostrada na Figura 4.28 trata o versionamento do cenário de dados do exemplo do problema da dieta. O problema da dieta foi elaborado como um cenário de dados do modelo matemático do PPL em sua forma canônica e os dados apresentados na Tabela 4.3 estão na primeira versão deste cenário. As telas apresentadas nas Figuras 4.14 até 4.20 são todas referentes à primeira versão do cenário de dados do problema da dieta.

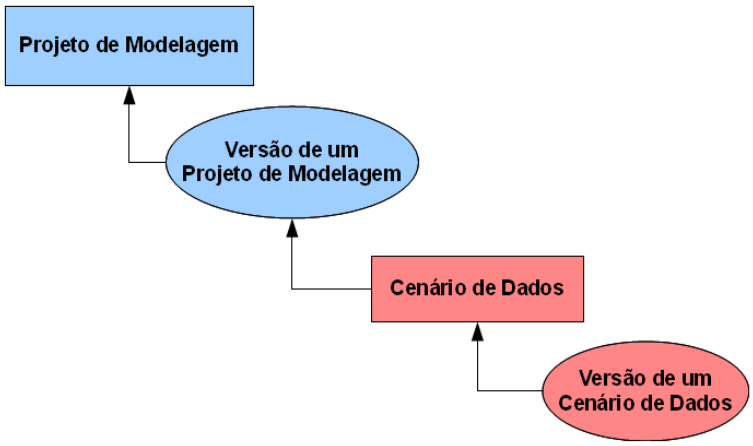


Figura 4.27: Associação entre os projetos de modelagem e os cenários de dados

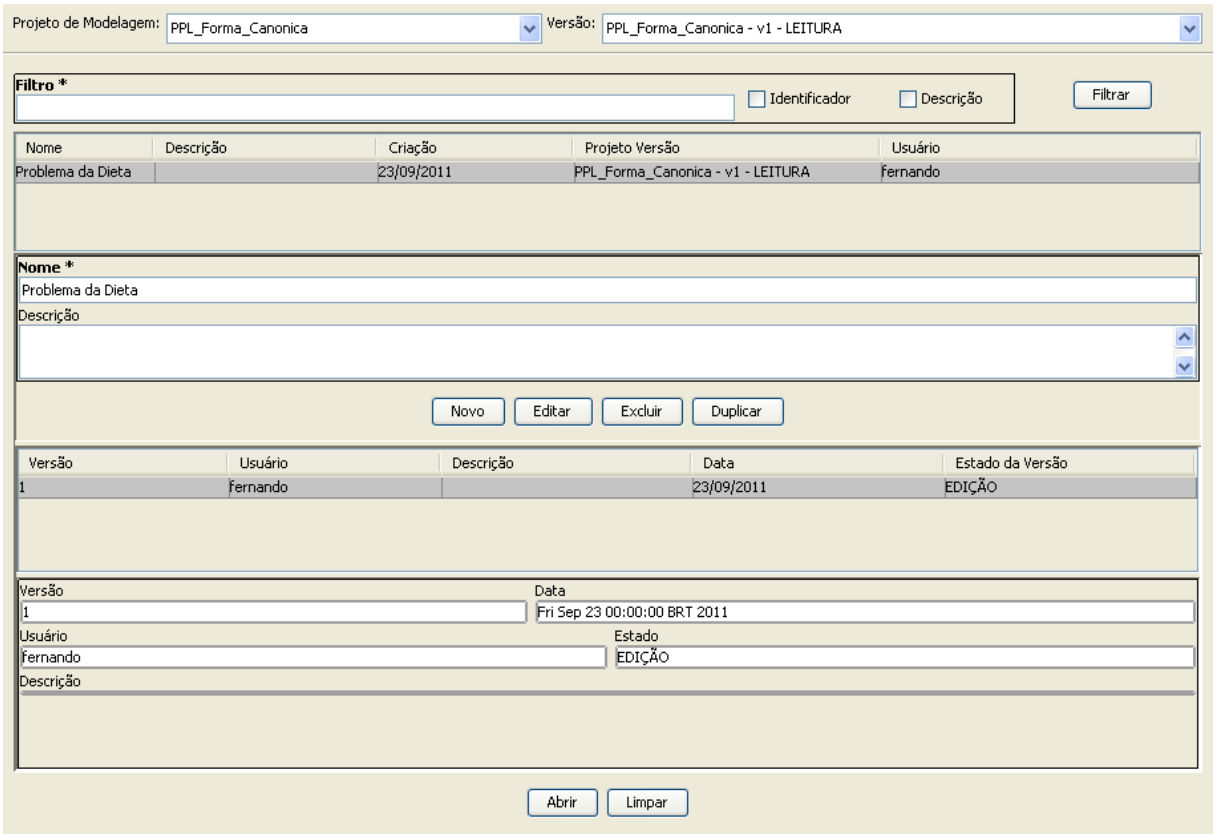


Figura 4.28: Versionamento do cenário de dados do problema da dieta

4.5 SERVIDORES DE OTIMIZAÇÃO

Os servidores de otimização são responsáveis por executar as solicitações de otimização dos PPLI's modelados pelo GeMM. As suas principais atribuições são receber uma requisição de um usuário para otimizar uma versão de um cenário de dados, gerar uma instância de um PPLI a partir das informações do modelo matemático e do cenário de dados,

passar esta instância para um resolvedor, obter a solução do problema e retornar esta solução para o usuário. De acordo com a arquitetura mostrada no Capítulo 3, os servidores de otimização podem ser implantados em máquinas separadas da aplicação central do GeMM. Também é possível ter uma instalação monolítica, na qual a aplicação central e os servidores de otimização estão na mesma máquina.

Independente da forma de implantação, o servidor de otimização tem a função de tratar as solicitações de otimização e se comunicar com um resolvedor. A diferença é que, no caso do servidor de otimização estar separado da aplicação central, as solicitações para execução da otimização de cenários de dados são feitas usando serviços *web*, através da rede. É importante destacar que o serviço *web* é usado apenas para notificar o servidor de otimização que um cenário deve ser otimizado, pois as informações do cenário de dados e do respectivo modelo matemático são obtidas diretamente do banco de dados, como mostra o esquema da Figura 4.29. Os resultados também são armazenados no banco de dados. A transferência de dados e a chamada da otimização é feita pela API do próprio resolvedor, que deve, assim, estar instalado e configurado na mesma máquina do servidor de otimização.

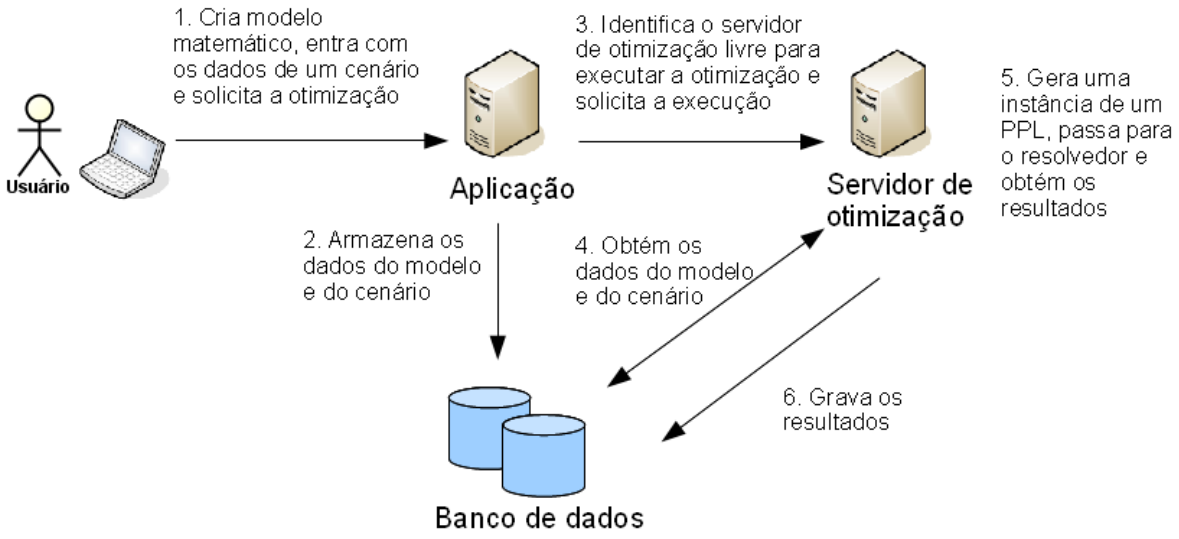


Figura 4.29: Esquema para execução de otimizações

4.6 COMUNICAÇÃO COM OS RESOLVEDORES

Uma tarefa importante dos servidores de otimização é a criação de uma instância de um PPLI a partir de um modelo matemático e um cenário de dados. Como visto anteriormente, um modelo matemático criado no GeMM permite a utilização de expressões com constantes, parâmetros e os operadores da Tabela 4.1 para definir os coeficientes das

variáveis, o RHS das restrições, a fórmula de cálculo de parâmetros e das condições de geração. Assim, o GeMM, no momento da criação de uma instância de um PPLI, deve avaliar todas essas expressões para gerar os dados no formato que os resolvedores entendem. Uma estrutura de dados comum para uma instância de um PPLI é o formato MPS (IBM CORPORATION, 1975), que é um arquivo contendo todos os dados da instância.

A Figura 4.30 mostra o diagrama de classes que o GeMM utiliza para tratar uma instância de um PPLI. Uma instância é gerada a partir dos dados de um modelo matemático, cuja estrutura de dados está descrita no diagrama da Figura 4.13, e de um cenário de dados, cuja estrutura de dados está descrita no diagrama da Figura 4.21. Apesar do diagrama de classes da Figura 4.13 possuir algumas classes com o mesmo nome de classes do diagrama da Figura 4.30, elas são diferentes, uma vez que o primeiro diagrama representa a definição do modelo matemático e o segundo representa uma instância de um PPLI.

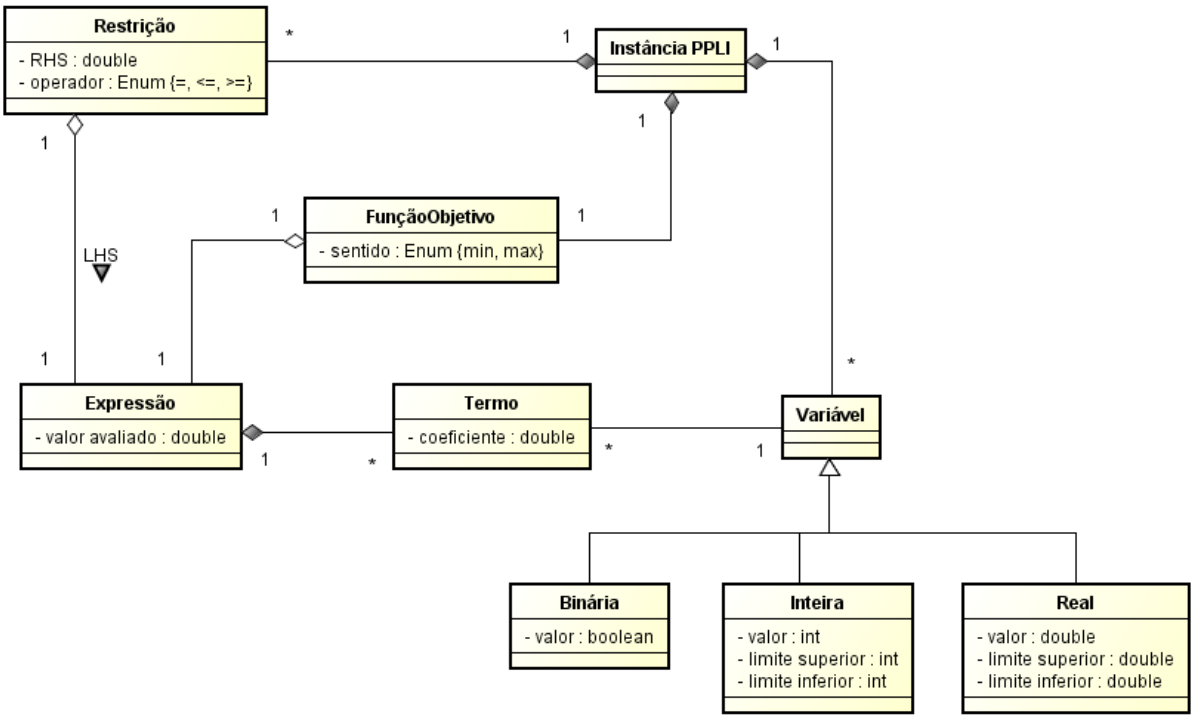


Figura 4.30: Diagrama de classes para representação de uma instância de um PPLI

Com os dados na estrutura apresentada na Figura 4.30, o GeMM pode tanto passar os dados para os resolvedores através de suas API's, quanto gerar um arquivo no formato MPS ou LP, que foram desenvolvidos para o CPLEX, mas também são utilizados por outros resolvedores. Como exemplo, a Figura 4.31 e a Figura 4.32 apresentam o cenário de dados do problema da dieta no formato MPS e no formato LP, respectivamente.

```
NAME          ProblemaDaDieta
ROWS
 N  obj
 G  Rest0
 G  Rest1
 G  Rest2
COLUMNS
    x_leite         obj           2.5
    x_leite         Rest0         2
    x_leite         Rest1         50
    x_leite         Rest2         2
    x_carne         obj           15.7
    x_carne         Rest0         2
    x_carne         Rest1         10
    x_carne         Rest2         90
    x_peixe         obj           10.9
    x_peixe         Rest0         4
    x_peixe         Rest1         50
    x_peixe         Rest2         40
    x_salada        obj           5.7
    x_salada        Rest0         20
    x_salada        Rest1         10
    x_salada        Rest2         30
RHS
    rhs             Rest0         11
    rhs             Rest1         70
    rhs             Rest2        250
ENDATA
```

Figura 4.31: Problema da dieta no formato MPS

```
\Problem name: ProblemaDaDieta

Minimize
 obj:  2.5 x_leite    + 15.7 x_carne  + 10.9 x_peixe + 5.7 x_salada
Subject To
 Rest0: 2 x_leite     + 2 x_carne  + 4 x_peixe   + 20 x_salada >= 11
 Rest1: 50 x_leite    + 10 x_carne + 50 x_peixe  + 10 x_salada >= 70
 Rest2: 2 x_leite     + 90 x_carne + 40 x_peixe  + 30 x_salada >= 250
End
```

Figura 4.32: Problema da dieta no formato LP

O quadro da Figura 4.33 apresenta o pseudocódigo que detalha a criação dos objetos das classes do diagrama da Figura 4.30 a partir de um modelo matemático e de um cenário de dados. Neste pseudocódigo pode-se observar a integração dos cenários de dados e do modelo matemático para a geração de uma instância de um PPLI. Nas linhas 18, 34, 45 e 66 foi feita a comparação da condição de geração com *VERDADEIRO* apenas para aumentar a clareza do pseudocódigo, uma vez que essas linhas podiam ser escritas omitindo este termo, ficando somente *"SE condição de geração ENTÃO"*.

No trecho da linha 01 até a linha 08 está apresentado o processamento dos parâmetros calculados, ou seja, para cada parâmetro deste tipo, seu valor deve ser avaliado de acordo com a sua fórmula de cálculo. Nesta implementação do GeMM, a fórmula de cálculo não pode conter parâmetros calculados, apenas parâmetros primários e constantes.

```
01: // Avaliação da expressão de cálculo dos parâmetros calculados
02: PARA cada parâmetro calculado no modelo matemático FAÇA
03:   Identificar os parâmetros primários e seus índices na expressão de cálculo;
04:   PARA cada combinação dos valores dos índices do parâmetro calculado no cenário FAÇA
05:     Identificar no cenário os valores dos parâmetros dado os valores dos índices;
06:     Analisar a expressão de cálculo do parâmetro e guardar seu valor no cenário;
07:   FIM PARA
08: FIM PARA
09:
10: // Identificação das variáveis do modelo matemático
11: PARA cada variável no modelo matemático FAÇA
12:   Identificar os parâmetros e seus índices na expressão da condição de geração;
13:   Identificar os parâmetros e seus índices na expressão do limite inferior;
14:   Identificar os parâmetros e seus índices na expressão do limite superior;
15:   PARA cada combinação dos valores dos índices da variável no cenário de dados FAÇA
16:     Identificar no cenário os valores dos parâmetros dado os valores dos índices;
17:     Analisar a expressão da condição de geração da variável;
18:     SE condição de geração = VERDADEIRO ENTÃO
19:       Identificar o tipo da variável no modelo (binária, inteira ou real);
20:       Analisar a expressão do limite inferior da variável e guardar seu valor;
21:       Analisar a expressão do limite superior da variável e guardar seu valor;
22:       Criar e adicionar variável na instância do PPLI;
23:     FIM SE
24:   FIM PARA
25: FIM PARA
26:
27: // Identificação das restrições do modelo matemático
28: PARA cada restrição no modelo matemático FAÇA
29:   Identificar os parâmetros e seus índices na expressão da condição de geração;
30:   Identificar os parâmetros e seus índices na expressão do limite da restrição (RHS);
31:   PARA cada combinação dos valores dos índices da restrição no cenário de dados FAÇA
32:     Identificar no cenário os valores dos parâmetros dado os valores dos índices;
33:     Analisar a expressão da condição de geração da restrição;
34:     SE condição de geração = VERDADEIRO ENTÃO
35:       Analisar a expressão do limite da restrição (RHS) e guardar seu valor;
36:       Criar e adicionar restrição na instância do PPLI;
37:       // Monta a expressão do LHS da restrição
38:       PARA cada termo no LHS da restrição FAÇA
39:         Identificar os índices da variável no termo que não existem na restrição;
40:         Identificar os parâmetros e seus índices na expressão do coeficiente do termo;
41:         Identificar os parâmetros e seus índices na condição de geração do termo;
42:         PARA cada combinação dos valores dos índices que não existem na restrição FAÇA
43:           Identificar os valores dos parâmetros dado os valores dos índices;
44:           Analisar a expressão da condição de geração do termo no LHS da restrição;
45:           SE condição de geração = VERDADEIRO ENTÃO
46:             Analisar a expressão do coeficiente do termo no LHS da restrição;
47:             Identificar a variável na instância do PPLI dado os valores dos índices;
48:             Criar e adicionar o termo na restrição na instância do PPLI;
49:           FIM SE
50:         FIM PARA
51:       FIM PARA
52:     FIM SE
53:   FIM PARA
54: FIM PARA
55: // Identificação da função objetivo do modelo matemático
56: Identificar o sentido da função objetivo;
57: Criar e definir função objetivo na instância do PPLI;
58: // Monta a expressão do equacionamento da função objetivo
59: PARA cada termo no equacionamento da expressão da função objetivo FAÇA
60:   Identificar os índices e os respectivos conjuntos da variável no termo;
61:   Identificar os parâmetros e seus índices na expressão do coeficiente do termo;
62:   Identificar os parâmetros e seus índices na da condição de geração do termo;
63:   PARA cada combinação dos valores dos índices da variável no termo FAÇA
64:     Identificar no cenário os valores dos parâmetros dado os valores dos índices;
65:     Analisar a expressão da condição de geração do termo do equacionamento;
66:     SE condição de geração = VERDADEIRO ENTÃO
67:       Analisar a expressão do coeficiente do termo do equacionamento da FO;
68:       Identificar a variável na instância do PPLI dado os valores dos índices;
69:       Criar e adicionar termo na expressão da FO na instância do PPLI;
70:     FIM SE
71:   FIM PARA
72: FIM PARA
```

Figura 4.33: Pseudocódigo para a criação de uma instância de um PPLI

Nos laços iniciados nas linhas 11 e 28 são percorridas as variáveis e restrições do modelo, respectivamente. Esses laços se referem às definições desses elementos de modelagem. Por exemplo, no PPL em sua forma canônica apresentado anteriormente, os laços se referem à variável x_j e à restrição (5). Já os laços da linha 15 e da linha 31 percorrem os valores dos índices que indexam as variáveis e as restrições. Por exemplo, no cenário de dados do problema da dieta, apresentado como um cenário do PPL em sua forma canônica, cujos dados estão na Tabela 4.3, o laço da linha 15 percorre os elementos do conjunto J = {*leite, carne, peixe, salada*}, que indexam a variável x_j, e o laço da linha 31 percorre os elementos do conjunto I = {*vitamina A, vitamina C, vitamina D*}, que indexam a restrição do problema. As linhas 22 e 36 criam as variáveis e as restrições, respectivamente, na instância do PPLI, representada no diagrama de classes da Figura 4.30. Assim, para este modelo matemático e este cenário de dados existem quatro variáveis e três restrições na instância do PPLI.

O LHS da restrição do problema é criado no laço da linha 38 do pseudocódigo. Este laço percorre todos os termos da expressão do LHS definidos no modelo, ou seja, percorre todos os objetos da classe *EquacionamentoRestrição*, que pode-se verificar no diagrama de classes da Figura 4.13. O exemplo do PPL em sua forma canônica possui um termo na restrição (5), dado por $a_{ij}.x_j$. Conforme dito anteriormente, o GeMM identifica que existe um somatório na restrição, observando os índices na variável do termo que não existem na restrição. O laço da linha 42 percorre os elementos dos conjuntos dos índices da variável que não existem entre os índices da restrição. No exemplo do modelo do PPL em sua forma canônica, a restrição é indexada por i e a variável do termo $a_{ij}.x_j$ é indexada por j, logo esse laço percorre todos os elementos do conjunto J = {*leite, carne, peixe, salada*}, dado pelo cenário de dados do problema da dieta. Assim, é possível afirmar que este laço da linha 42 resolve o somatório em j da restrição (5) do modelo. A linha 48 cria um objeto da classe *Termo*, do diagrama da Figura 4.30, o associa ao respectivo objeto da classe *Expressão,* associado ao objeto da classe *Restrição* criado na linha 36, e com o objeto da classe *Variável*, criado na linha 22 e identificado unicamente pelos valores dos índices na linha 47.

Na sequência do pseudocódigo, da linha 56 a 72, é criada e definida a função objetivo na instância do PPLI. O seu sentido, minimizar ou maximizar, é definido na linha 56 e a sua expressão é criada da mesma forma que o LHS da restrição. Os laços das linhas 59 e 63 são análogos aos laços das linhas 38 e 42, respectivamente, da expressão do LHS da restrição. A única diferença é que a função objetivo não possui índices, logo o laço da linha 63 é feito

sobre a combinação dos elementos dos conjuntos de todos os índices da variável definida no termo. No exemplo do PPL em sua forma canônica, o único termo da função objetivo é dado por $c_j.x_j$ e o laço da linha 63 percorre todos os elementos do conjunto J = {*leite, carne, peixe, salada*}, dado pelo cenário de dados do problema da dieta, e representa o somatório em j da função objetivo (4).

4.7 FERRAMENTAS UTILIZADAS

O GeMM utiliza um banco de dados para armazenar suas informações. A sua implementação permite o uso do Oracle (ORACLE, 2011a) ou do HSQLDB (HSQLDB, 2011). A opção pelo Oracle se deve ao fato de ser um SGBD robusto e amplamente utilizado no mercado. Já o uso do HSQLDB foi motivado por ser um software de banco de dados de código aberto, compacto e desenvolvido em Java, permitindo que o GeMM seja usado em uma instalação monolítica, na qual todos os componentes de software são executados no mesmo computador, sem a necessidade de uma implantação cliente-servidor. Apesar da utilização desses dois SGBD's, a maior parte dos SGBD's que implementam o SQL (*Structured Query Language*) padrão, podem ser usados de forma transparente ou com a alteração apenas de alguns parâmetros.

Para tratar algumas funcionalidades do GeMM, foram utilizadas algumas bibliotecas Java. Para fazer o tratamento e auxiliar na avaliação das expressões matemáticas, foi usado o avaliador de expressão JEP (SINGULAR SYSTEMS, 2011), que também possui um projeto de código aberto (JEP JAVA, 2011). Esta biblioteca é importante para fazer a análise sintática das expressões matemáticas e, dados os valores dos parâmetros, avaliar numericamente as expressões. Ela é usada, por exemplo, na avaliação das condições de geração, nas expressões dos coeficientes das variáveis, nos parâmetros calculados etc. Para exibir graficamente as expressões das restrições e da função objetivo no GeMM, usando símbolos matemáticos, como visto na Figura 4.9 e na Figura 4.10, utilizou-se a biblioteca JLaTeXMath (JLATEXMATH, 2011). Esta biblioteca, de código aberto, gera uma imagem de fórmulas matemáticas escritas em LaTeX (LAMPORT, 2011). A tradução das expressões modeladas no GeMM em código LaTeX é implementada diretamente no GeMM. Para a construção dinâmica das telas dos cenários de dados, a fim de permitir a configuração de componentes gráficos através de arquivos XML, como, por exemplo, a etiqueta dos campos, a largura das colunas nas tabelas etc., foi usada a biblioteca Swingbean (GUERRA, 2008). Com o intuito de fazer a persistência dos dados em banco de dados, utilizou-se a biblioteca de mapeamento Objeto/Relacional EclipseLink JPA (ECLIPSE FOUNDATION, 2011). A biblioteca

EclipseLink JPA é importante, pois o GeMM é orientado a objetos e seus dados são armazenados em um banco de dados relacional. Logo, ela permitiu o mapeamento entre esses dois paradigmas.

Alguns softwares externos ao GeMM também foram usados para seu projeto e apoio. A geração da documentação do modelo matemático em formato PDF (ADOBE SYSTEMS, 1993) pode ser feita usando a ferramenta pdfTeX (HAN THE THANH, 1998), uma vez que o GeMM gera o arquivo em formato LaTeX e este pode ser convertido em PDF com esse software. A codificação do GeMM foi feita com o uso da IDE Netbeans (ORACLE, 2011b). Para projetar o ambiente de modelagem, foram usados dois softwares: um para modelagem do sistema, usando diagramas UML, e outro de modelagem de banco de dados, para elaboração do DER (Diagrama de Entidade e Relacionamento) e geração dos códigos SQL para criação e alteração das definições da estrutura do banco de dados. Os diagramas UML foram feitos através do Astah Community (CHANGE VISION, 2011), software cuja licença permite o uso acadêmico. A modelagem de banco de dados foi feita por meio do DBDesigner (FABFORCE.NET, 2011), software que permite a elaboração de DER's e geração de código SQL.

Além disso, foi implementada a comunicação com três diferentes resolvedores: o CPLEX (IBM CORPORATION, 2010), o Gurobi (GUROBI, 2011) e o GLPK (MAKHORIN, 2012). O CPLEX e o Gurobi foram escolhidos por serem softwares bastante robustos e reconhecidos como bons resolvedores comerciais de PPLI, e também por possuírem licença para uso acadêmico. Utilizou-se também o GLPK por ser uma opção de resolvedor com código aberto. Outros resolvedores também podem ser utilizados através da geração de um arquivo no formato MPS (IBM CORPORATION, 1975) ou LP (IBM CORPORATION, 2010), que representa uma instância de um PPLI, ou implementando a comunicação do GeMM com esse resolvedor diretamente, desde que este forneça uma API que possa ser chamada através da linguagem Java ou outra linguagem que interopere com Java, como C, por exemplo.

4.8 INFLUÊNCIA DOS TRABALHOS RELACIONADOS

As características descritas por GEOFFRION (1989) ajudaram a direcionar as pesquisas acerca do que se espera de um ambiente de modelagem. No desenvolvimento do GeMM procurou-se atender as necessidades, tanto dos profissionais envolvidos com o processo de tomada de decisão, quanto dos modeladores. Também buscou-se gerenciar a evolução dos modelos matemáticos e dos recursos ao seu redor.

Para o tratamento da modelagem matemática, o trabalho de MURPHY *et al.* (1992) forneceu uma análise abrangente sobre as possibilidades de representação dos PPLI's. O GeMM utiliza bancos de dados para armazenar os modelos e MURPHY *et al.* (1992) destacam a importância de dois requisitos distintos neste tipo de representação, que devem estar relacionados: a necessidade de registrar informações sobre a estrutura do modelo matemático e a de armazenar os dados do problema e os resultados. Esses requisitos estão de acordo com a proposta de LEE (1991), que aborda a separação entre o modelo e os dados.

Alguns trabalhos, como o de FOURER (1997), utilizam bancos de dados para guardar as informações do modelo e dos dados associados, porém são soluções para problemas específicos ou para uma determinada área de aplicação. A abordagem utilizada no desenvolvimento do GeMM busca tratar de forma geral o armazenamento em bancos de dados de qualquer modelo de PPLI e promover a independência entre modelo, dados e resolvedores, como proposto por RAMIREZ *et al.* (1993).

As representações de modelos de PPLI através de grafos e diagramas de fluxos em rede, abordadas por COLLAUD e PASQUIER-BOLTUCK (1994), FORSTER e MEVERT (1994) e CHARI e SEN (1998), foram descartadas pela dificuldade da construção e manutenção de modelos de grande porte. Porém, no desenvolvimento do GeMM, optou-se por não utilizar uma abordagem puramente textual para modelar os PPLI's, como as representações algébricas do GAMS e do AMPL, por exemplo. A proposta do GeMM procura balancear a agilidade no desenvolvimento de modelos, a flexibilidade e a manutenibilidade, entre outras características levantadas por MAROS e KHALIQ (2002), mesclando a utilização de formulários e linguagem textual algébrica na interface com o usuário. O GeMM se aproxima do AIMMS em sua forma de modelar, apesar das diferenças de suas arquiteturas.

Quanto à arquitetura do sistema, os conceitos utilizados no GeMM se aproximam das ideias do trabalho de MATURANA *et al.* (2004). Assim como esse trabalho, o GeMM gera automaticamente a estrutura de banco de dados e a interface com o usuário a partir do modelo matemático, procurando, principalmente, agilizar o desenvolvimento de sistemas para apoio à decisão. O tratamento do controle de versão é uma das características que diferem a abordagem do GeMM do trabalho de MATURANA *et al.* (2004). A produtividade no desenvolvimento de modelos e a facilidade de uso destes nos processos de tomada de decisão são fatores determinantes para a utilização dos modelos matemáticos nos processos decisórios das organizações. Pode-se observar nos estudos analisados no Capítulo 2 que características

como o controle de versão e a gerência de configuração dos modelos matemáticos são pouco exploradas e estudadas, apesar de bastante relevantes, como destaca MAKOWSKI (2005).

4.9 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais conceitos utilizados na abordagem do GeMM, descritos no Capítulo 3, com o foco na implementação do ambiente de modelagem. Inicialmente, mostrou-se como o usuário modela um PPLI no GeMM, através do exemplo do PPL em sua forma canônica. Foram apresentadas as estruturas de dados para tratar e armazenar em banco de dados as informações dos modelos matemáticos.

Na abordagem utilizada para implementação do GeMM, elaborou-se formulários para que os usuários modelassem os PPLI's, como mostram as Figuras de 4.1 até 4.10. Esses formulários foram utilizados por conveniência, pois permitem que a estrutura de dados apresentada na Figura 4.13 seja preenchida. Porém, esta interface poderia ser substituída por uma linguagem textual como a AMPL, por exemplo. Neste caso, um editor e um analisador dessa linguagem deveriam ser desenvolvidos para fazer a interface dos usuários com as estruturas de dados do GeMM. Como as funcionalidades propostas se baseiam nas estruturas de dados dos PPLI's, elas não seriam perdidas se a interface para edição dos modelos fosse trocada. Este trabalho não tem o propósito de avaliar qual seria a melhor interface.

Este capítulo também descreveu a implementação dos cenários de dados, utilizando o problema da dieta como exemplo. A estrutura de dados para armazenar as informações dos cenários utiliza a ideia da metamodelagem, permitindo usar o mesmo esquema de banco de dados para tratar qualquer cenário referente aos problemas que podem ser modelados pelo GeMM. Outra característica importante é a geração dinâmica das telas de interface com os usuários de acordo com o modelo matemático criado. Em seguida detalhou-se as características do controle de versão, tanto para modelos quanto para os cenários de dados, e dos servidores de otimização, assim como o GeMM cria instâncias de PPLI a partir dos cenários de dados e do modelo matemático. Por fim, mostrou-se todas as ferramentas e bibliotecas que foram usadas para o projeto e a implementação do GeMM, e a influência dos trabalhos relacionados, analisados no Capítulo 2.

O Capítulo 5 apresenta os modelos matemáticos cujas modelagens são mais complexas do que o PPL em sua forma canônica, exemplificado neste capítulo. Esses modelos foram usados para validar o GeMM e mostrar como funcionalidades mais avançadas podem ser usadas para modelar problemas complexos.

CAPÍTULO 5 - VALIDAÇÃO DO GEMM

5.1 INTRODUÇÃO

Este capítulo apresenta a validação do GeMM como ambiente de modelagem de PPLI. Para avaliar a sua capacidade de modelar, foram escolhidos dois problemas de otimização que exploram suas principais características. O primeiro problema apresentado tem por objetivo programar as rodadas de um torneio envolvendo times de diferentes cidades, denominado *The Traveling Tournament Problem with Predefined Venues* (TTPPV) ou problema do torneio viajante com locais pré-definidos. MELO *et al.* (2009) propuseram três modelagens diferentes para este problema, cuja quantidade de variáveis em cada modelo varia com o número de times participantes em $O(n^3)$, $O(n^4)$ e $O(n^5)$, onde n é o número de times no torneio.

A seguir, este capítulo apresenta as implementações dos modelos com $O(n^3)$ e $O(n^5)$ variáveis. As principais características que motivaram a utilização desses modelos na validação do GeMM foram: o tratamento de condições de geração de variáveis e de restrições, uso de parâmetros calculados a partir de outros parâmetros e a mistura de variáveis inteiras com variáveis contínuas em modelos mistos.

O segundo problema utilizado para validar o GeMM é o Problema de Programação de Entrega de Pedidos (PPEP) de produtos derivados de petróleo. Este é um problema de uma subsidiária da Petrobras, responsável por fazer tais entregas. O modelo matemático deste problema é mais simples do que os modelos do TTPPV, porém foi escolhido para validar o GeMM pela quantidade de dados disponíveis, o que possibilitou exercitar o GeMM na geração de instâncias de PPLI com dezenas de milhares de variáveis e restrições.

Para exemplificar o tratamento dos cenários de dados, é utilizado um torneio fictício para o TTPPV, que permitiu a geração e a otimização de instâncias do problema. Os resultados foram comparados com os resultados obtidos pelos mesmos modelos e dados implementados em um ambiente de modelagem comercial, a fim de garantir o correto tratamento deles pelo GeMM. O mesmo procedimento foi feito para o PPEP. Como os modelos apresentados envolvem variáveis inteiras, na otimização dos cenários foi utilizada uma tolerância relativa máxima de 0,2%, que representa a maior diferença percentual possível entre a função objetivo da solução ótima obtida pela relaxação linear e a função objetivo da melhor solução inteira encontrada.

5.2 *THE TRAVELING TOURNAMENT PROBLEM WITH PREDEFINED VENUES*

O TTPPV consiste em programar um torneio em turno único, envolvendo times de diferentes cidades, no qual os jogos e seus locais são pré-definidos (MELO *et al.*, 2009). O objetivo deste problema de otimização é definir em quais rodadas os jogos ocorrem, minimizando a distância total viajada por todos os times participantes do torneio. Neste problema, todos os times estão em suas cidades sede antes do início do torneio e devem retornar para elas após a última rodada. Outra condição deste problema é que cada time não deve jogar mais de três jogos consecutivos fora de casa ou em casa.

Neste problema, *n* times participam de um torneio jogado em *n-1* rodadas, onde *n* é um número par. Cada time possui seu próprio local de jogo na sua cidade sede. A distância entre as cidades do time *i* e do time *j* é dada pelo parâmetro $d_{ij} \geq 0$. O parâmetro *G* é o conjunto de jogos representados por pares ordenados. O jogo entre o time *i* e o time *j*, jogado na casa do time *i* é representado pelo par *(i, j)*. O par *(j, i)* representa o mesmo jogo, porém acontecendo na casa do time *j*. Como o TTPPV é um torneio de apenas um turno, ou $(i, j) \in G$, ou $(j, i) \in G$. Os parâmetros *G* e d_{ij} são dados de entrada do problema.

5.2.1 MODELO TTPPV – $O(n^3)$ VARIÁVEIS

Este modelo proposto por MELO *et al.* (2009) define duas variáveis binárias de decisão: z_{tjk}, que deve valer *1*, se o time *t* joga em casa contra o time *j* na rodada *k*, senão deve valer *0;* e y_{tij}, que deve valer *1*, se o time *t* viaja da cidade do time *i* para a cidade do time *j*, senão deve valer *0*. As variáveis *z* representam as rodadas em que os jogos ocorrem, enquanto as variáveis *y* representam as viagens dos times entre as cidades. Segue o modelo do TTPPV com $O(n^3)$ variáveis:

$$Minimizar \quad \sum_{t=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \cdot y_{tij} \tag{12}$$

Sujeito a

$$\sum_{q=1}^{n-1} z_{tjq} = 1, \quad \forall (t, j) \in G \tag{13}$$

$$\sum_{q=1}^{n-1} z_{tjq} = 0, \quad \forall (j, t) \in G \tag{14}$$

$$\sum_{\substack{j=1 \\ j \neq t}}^{n} (z_{tjk} + z_{jtk}) = 1, \quad \forall t = 1, \ldots, n, \quad k = 1, \ldots, n-1 \tag{15}$$

$$y_{tij} \geq z_{it,k-1} + z_{jtk} - 1, \quad \forall t, i, j = 1, \ldots, n \ \ com \ \ t \neq i \neq j, k = 2, \ldots, n-1 \tag{16}$$

$$y_{tit} \geq z_{it,k-1} + \sum_{\substack{j=1 \\ j \neq t}}^{n} z_{tjk} - 1, \quad \forall t, i = 1, \ldots, n \ \ com \ \ t \neq i, k = 2, \ldots, n-1 \tag{17}$$

$$y_{tti} \geq \sum_{\substack{j=1 \\ j \neq t}}^{n} z_{tj,k-1} + z_{itk} - 1, \quad \forall t, i = 1, \ldots, n \ \ com \ \ t \neq i, k = 2, \ldots, n-1 \tag{18}$$

$$y_{tti} \geq z_{it1}, \quad \forall t, i = 1, \ldots, n \ \ com \ \ t \neq i \tag{19}$$

$$y_{tit} \geq z_{it,n-1}, \quad \forall t, i = 1, \ldots, n \ \ com \ \ t \neq i \tag{20}$$

$$\sum_{q=k}^{k+3} \sum_{\substack{j=1 \\ j \neq t}}^{n} z_{jtq} \leq 3, \quad \forall t = 1, \ldots, n, k = 1, \ldots, n-4 \tag{21}$$

$$\sum_{q=k}^{k+3} \sum_{\substack{j=1 \\ j \neq t}}^{n} z_{jtq} \geq 1, \quad \forall t = 1, \ldots, n, k = 1, \ldots, n-4 \tag{22}$$

$$z_{tjk} \in \{0,1\}, \quad \forall t, j = 1, \ldots, n, k = 1, \ldots, n-1 \tag{23}$$

$$0 \leq y_{tij} \leq 1, \quad \forall t, i, j = 1, \ldots, n \tag{24}$$

A função objetivo (12) busca minimizar a distância percorrida pelos times durante o torneio. A restrição (13) garante que cada jogo contido em *G* ocorre exatamente uma vez, enquanto a restrição (14) define com *0* as variáveis z_{tjk} que representam os jogos entre os times *t* e *j* na casa do time *t,* se estes jogos estão predefinidos para serem jogados na casa do time *j*. A restrição (15) garante que cada time joga somente um jogo por rodada. A restrição (16) garante que o time *t* deve fazer uma viagem do local do time *i* para o local do time *j* se ele joga dois jogos consecutivos fora de casa contra os times *i* e *j*, respectivamente. A restrição (17) força o time *t* fazer uma viagem da cidade do time *i* para a sua cidade sede, se existe um jogo fora de casa contra *i* e um jogo em casa na rodada seguinte. A restrição (18) garante que o time *t* deve viajar de sua cidade para a cidade do time *i*, se tiver um jogo em casa na rodada anterior e um fora de casa na rodada atual contra o time *i*. A restrição (19) garante que o time *t* viaja para a cidade do time *i* se na primeira rodada ele joga contra *i* fora de casa, uma vez que todos os times começam o torneio na sua cidade. A restrição (20) garante que o time *t* viaja da cidade do time *i* para a sua cidade, se ele joga na última rodada contra *i* fora de casa, já que os times devem estar em casa após a última rodada. A restrição (21) estabelece que o time *t* não

pode jogar mais de três partidas consecutivas fora de casa. Analogamente, a restrição (22) estabelece que o time *t* não pode jogar mais de três partidas consecutivas em casa. A restrição (23) define que as variáveis *z* devem ser binárias. As variáveis *y* atuam como limites superiores nas restrições (16) e (20). Desde que as distâncias d_{ij} sejam todas não negativas, as variáveis *y* sempre vão assumir os menores valores possíveis, que são necessariamente *0* ou *1*. Assim, as variáveis *y* podem assumir valores reais ao invés de binários, desde que atendam a restrição (24).

Esse modelo foi escolhido para validar o GeMM por explorar características de modelagem mais avançadas do que o PPL em sua forma canônica apresentado no Capítulo 4. A modelagem deste problema envolve o tratamento de condições de geração, como por exemplo, a restrição (13), que só vai existir quando o par *(t, j)* pertencer ao conjunto *G*. As restrições (15), (17), (18), (21) e (22) possuem também uma condição de geração dos termos do LHS das restrições. Nelas, o somatório sobre o índice *j* só deve ocorrer para os termos cujo valor do índice *j* seja diferente do valor do índice *t* ($j \neq t$). Outra peculiaridade envolvendo os somatórios dos termos das restrições são os somatórios sobre um determinado índice, cujos limites são dados por outro índice, como, por exemplo, o somatório sobre o índice *q* nas restrições (21) e (22), que é limitado pelo valor do índice *k*. Outra característica interessante do modelo é o fato dele ser misto, envolvendo variáveis binárias e reais.

5.2.2 IMPLEMENTAÇÃO DO MODELO TTPPV – $O(n^3)$ VARIÁVEIS

Para modelar este problema no GeMM, o primeiro passo foi identificar os conjuntos. Dados a descrição do problema e o modelo matemático, podem-se definir dois conjuntos: o conjunto dos times participantes, com *n* elementos, e o conjunto das rodadas do torneio, com *n-1* elementos. Podem-se identificar também os índices de cada conjunto: *t*, *i* e *j* representam um time participante e *q* e *k* uma rodada específica do torneio. A Figura 5.1 mostra a definição do conjunto *TIMES* e seus índices no GeMM. O tipo de dados deste conjunto foi definido como cadeia de caracteres (*String*), uma vez que o conjunto pode ser composto pelos próprios nomes dos times.

O passo seguinte para a implementação do modelo no GeMM foi a definição dos parâmetros. Foram identificados três parâmetros no modelo, como a Figura 5.2 mostra: *n*, que é o número total de times no torneio; *G*, que indica os jogos definidos para o torneio e o local onde eles ocorrem; e d_{ij}, que fornece a distância entre as cidades sede dos times *i* e *j*.

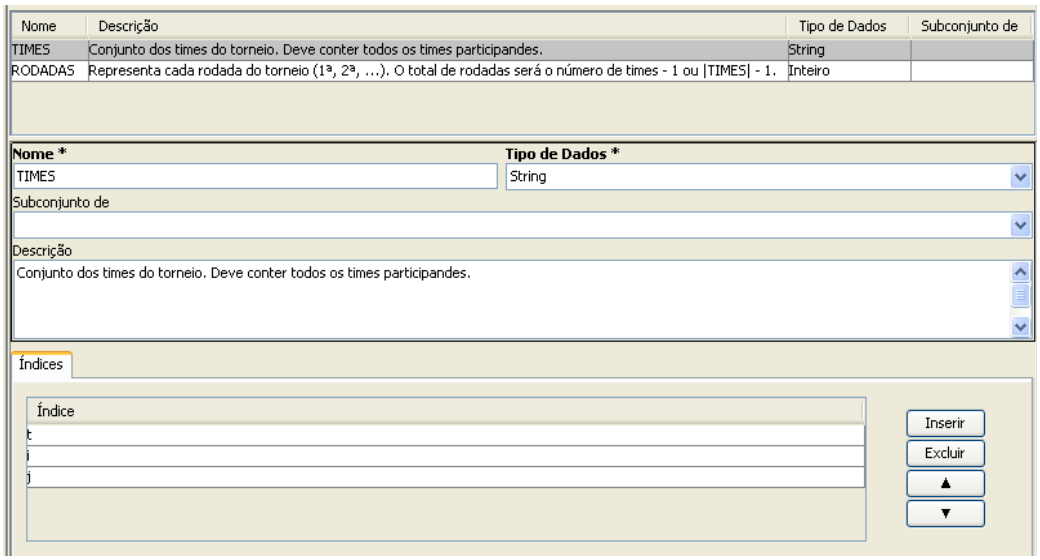


Figura 5.1: Conjuntos do modelo TTPPV com O(n^3) variáveis

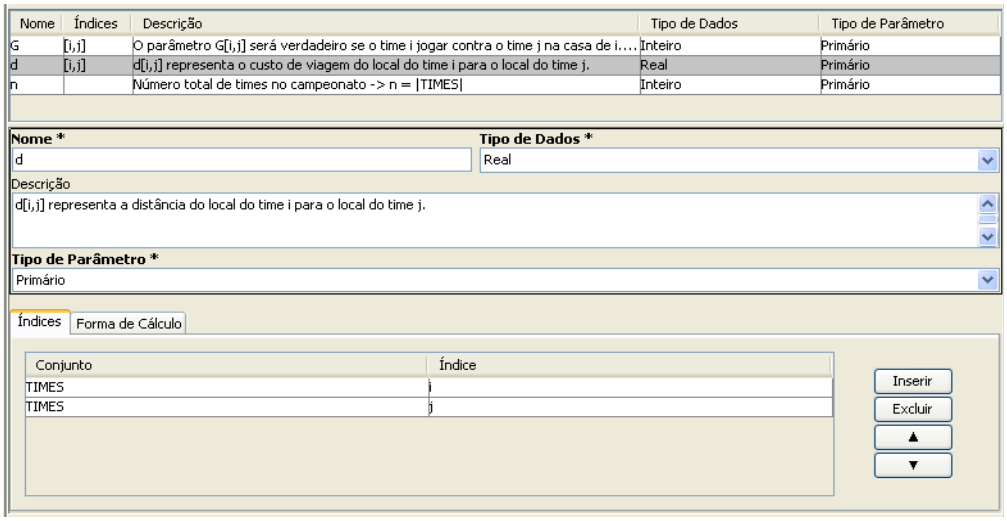


Figura 5.2: Parâmetros do modelo TTPPV com O(n^3) variáveis

O parâmetro *G* apresentado na definição deste problema como um conjunto de pares ordenados que representam os jogos do torneio, foi implementado no GeMM como uma matriz de inteiros, *G[i,j]*, que contém o valor *1* se o jogo entre os times *i* e *j* é jogado na cidade de *i*, e o valor *0* caso contrário. O parâmetro d_{ij} foi implementado como uma matriz de números reais, *d[i,j]*, que armazena a distância da cidade do time *i* para a cidade do time *j*, como exibe a Figura 5.2. E o parâmetro *n*, um número inteiro, escalar, que guarda o total de times do torneio, ou seja, é a própria cardinalidade do conjunto *TIMES*.

Com a criação dos conjuntos, índices e parâmetros foi possível definir as variáveis, restrições e função objetivo do modelo apresentado. Conforme descrito anteriormente, a variável z_{tjk} é uma variável binária que deve valer *1* quando o time *t* jogar em casa contra o

time *j* na rodada *k*, caso contrário deve valer *0*. Sua definição no GeMM como uma variável binária já modela a restrição de integralidade (23). A variável y_{tij} é uma variável real que está limitada entre *[0,1]*, como descrito na restrição (24). Ela representa as viagens dos times e deve receber valor *1* se o time *t* viaja da cidade do time *i* para a cidade do time *j*. A Figura 5.3 mostra a definição da variável y_{tij} no GeMM, o seu tipo *Real*, sua descrição, seus índices *t*, *i*, *j* e seus limites superior e inferior dado pela restrição (24).

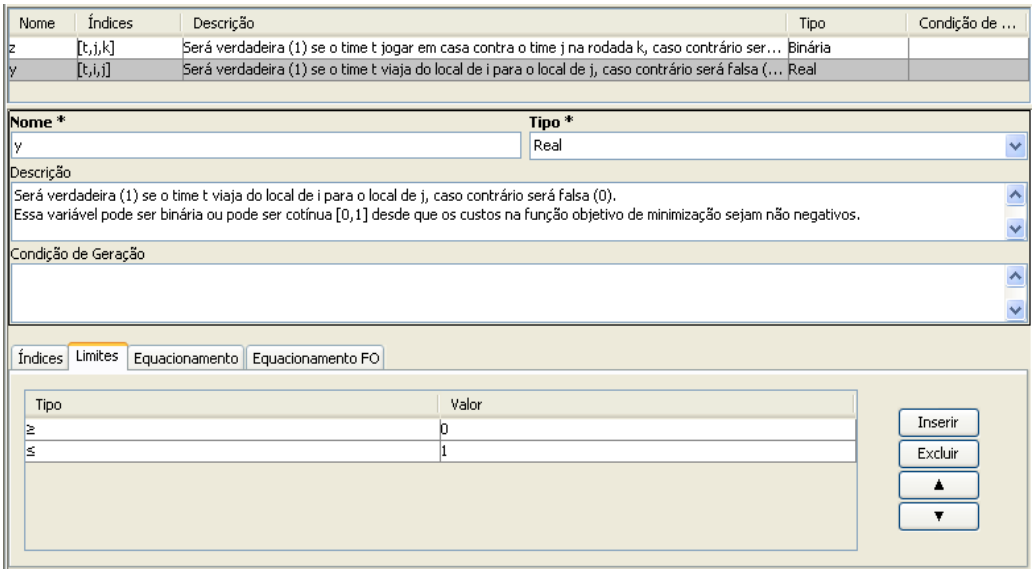


Figura 5.3: Variáveis do modelo TTPPV com O(n^3) variáveis

A função objetivo (12) foi implementada no GeMM como mostra a Figura 5.4. A sua definição é simples, precisando apenas indicar o seu sentido como *Minimizar*, sua descrição e o equacionamento da sua expressão. O equacionamento é dado pelo produto da variável y_{tij} por seu coeficiente, o parâmetro d_{ij}. Os índices que aparecem no termo $d_{ij}.y_{tij}$ são *t*, *i* e *j,* todos do conjunto *TIMES*. Logo, a expressão da função objetivo possui três somatórios sobre esses índices, como apresentado na Figura 5.4.

Até então, a definição desse modelo não precisou de funcionalidades do GeMM muito diferentes daquelas requeridas para a modelagem do PPL em sua forma canônica, apresentadas no Capítulo 4. Nas restrições definidas a seguir, destacam-se as funcionalidades ainda não exploradas. A restrição (13) existe para todo time *t* e time *j*, cujo jogo entre *t* e *j* ocorre na casa do time *t*. Assim, pode-se definir que essa restrição é indexada por *t* e *j* e que ela só deve existir quando o parâmetro *G[t,j]* valer *1*. Esta é a condição de geração dessa restrição na instância do problema, que no GeMM deve ser escrita na forma de uma expressão booleana. Conforme mostrado no Capítulo 4, as expressões das condições de geração de

variáveis e restrições podem ser construídas com valores constantes, parâmetros, índices, os operadores aritméticos da Tabela 4.1 e os operadores lógicos e de comparação da Tabela 4.2. Logo, a condição de geração da restrição (13), na notação do GeMM, pode ser escrita como *G[t,j]==1*, de acordo com a Figura 5.5.

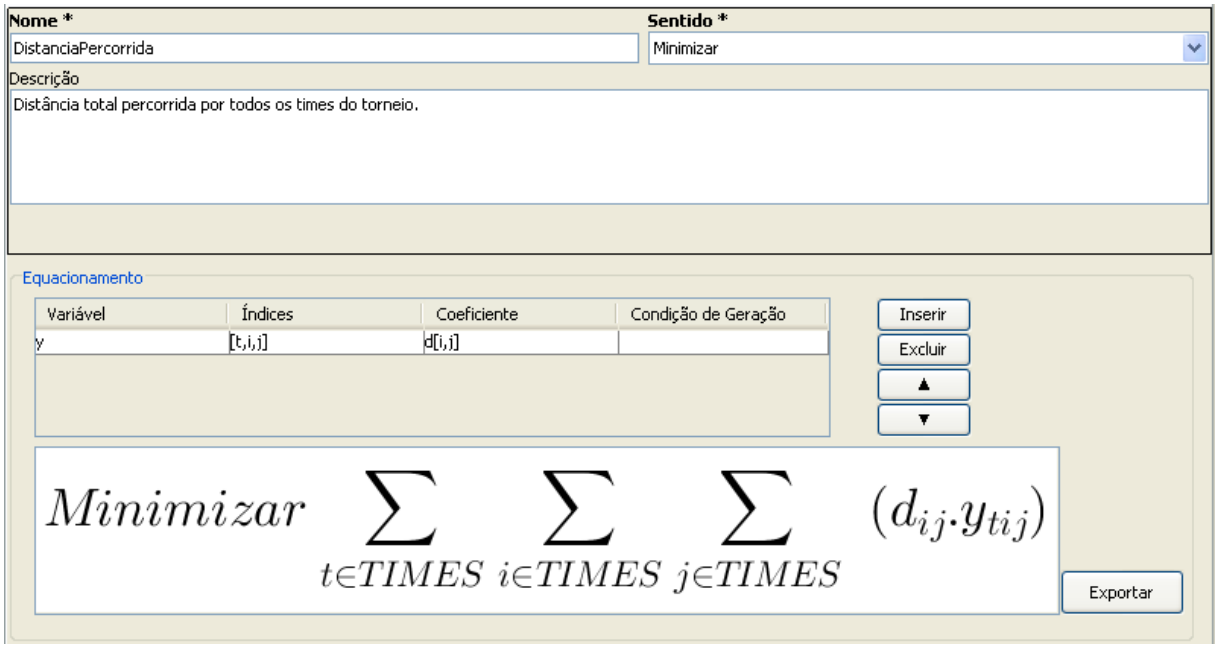


Figura 5.4: Função objetivo do modelo TTPPV com O(n³) variáveis

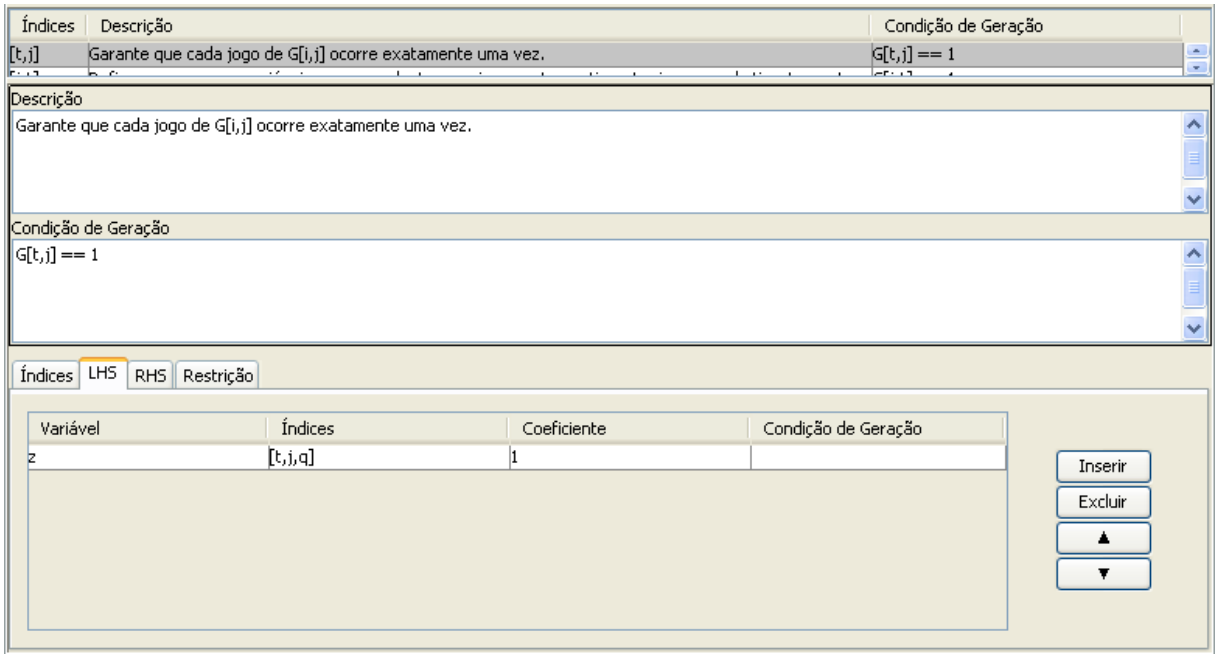


Figura 5.5: Restrição (13) *do modelo TTPPV com O(n³) variáveis*

A Figura 5.5 mostra o LHS da restrição (13). A variável *z* entra no equacionamento

com os índices *t*, *j* e *q,* porém, ela foi definida com os índices *t*, *j* e *k*. Como *q* e *k* são índices do mesmo conjunto *RODADAS*, eles podem ser usados um no lugar do outro. Além disso, pode-se destacar na Figura 5.5 que o coeficiente da variável *z* na restrição (13) é o valor *1*, que é o valor utilizado por padrão no GeMM, por ser o valor neutro da multiplicação. A Figura 5.6 exibe o RHS da restrição, assim como a indicação de que é uma restrição de igualdade.

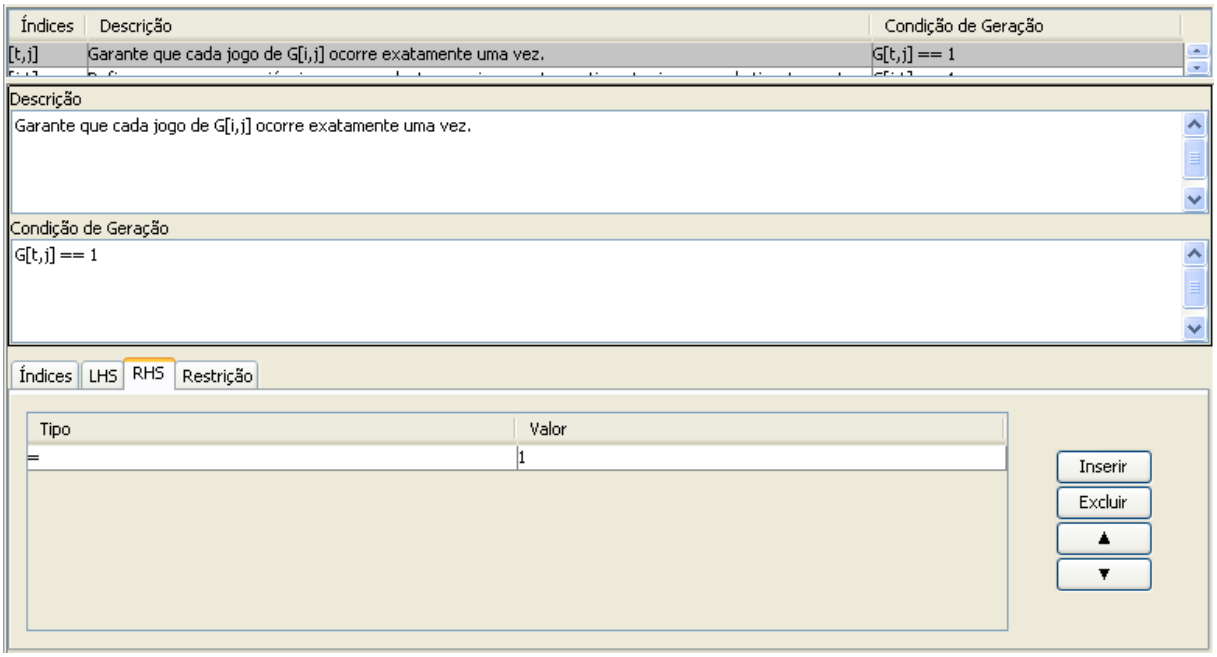


Figura 5.6: RHS da restrição (13) *do modelo TTPPV com O(n^3) variáveis*

Por fim, a Figura 5.7 mostra a expressão da restrição interpretada pelo GeMM, semelhante à expressão (13). A restrição (14) pode ser implementada no GeMM de forma análoga à restrição (13), apenas trocando o RHS de *1* para *0* e a condição de geração para *G[j,t]==1*.

A Figura 5.8 apresenta as informações no GeMM da restrição (15). Esta restrição existe para todo time *t* e para toda rodada *k*, logo é indexada por *t* e *k,* e não possui condição de geração específica. A peculiaridade desta restrição é que a expressão $(z_{tjk} + z_{jtk})$ deve ser somada para todo time *j*, desde que $j \neq t$. Logo, os termos devem respeitar a condição $j \neq t$ para entrarem no somatório. O LHS da restrição (15) foi modelado no GeMM com dois termos: a variável z_{tjk}, com coeficiente *1* e a condição de que $j \neq t$, e a variável z_{jtk}, também com coeficiente *1* e a condição $j \neq t$. Assim, a Figura 5.8 mostra que a variável *z* entra duas vezes no LHS da restrição com a mesma condição e o mesmo coeficiente, porém com diferentes índices. A restrição (15) é de igualdade e o RHS é igual a *1*.

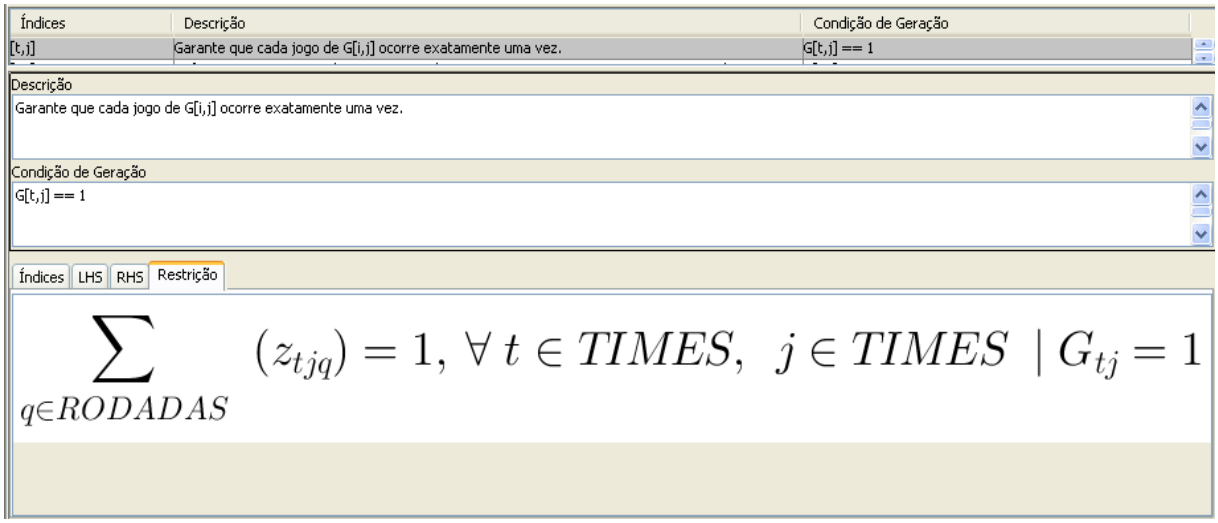


Figura 5.7: Expressão da restrição (13) *do modelo TTPPV com O(n³) variáveis*

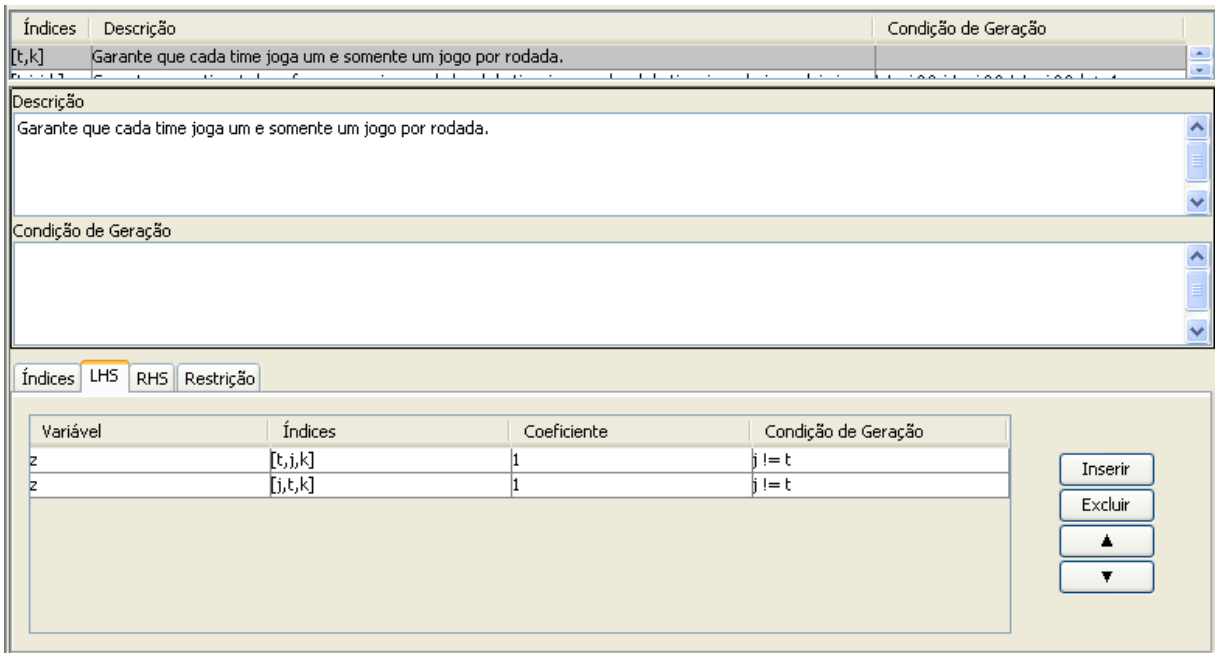


Figura 5.8: Restrição (15) *do modelo TTPPV com O(n³) variáveis*

Como a restrição (15) é indexada por t e k, e os índices j, t e k aparecem no equacionamento do LHS da restrição, existe um somatório sobre o índice j, único índice que não indexa a restrição, desde que $j \neq t$. A Figura 5.9 mostra a expressão da restrição, nela aparecem dois somatórios, um para cada termo, pois o GeMM processa termo por termo e não agrupa somatórios idênticos, porém matematicamente ela é equivalente à restrição (15).

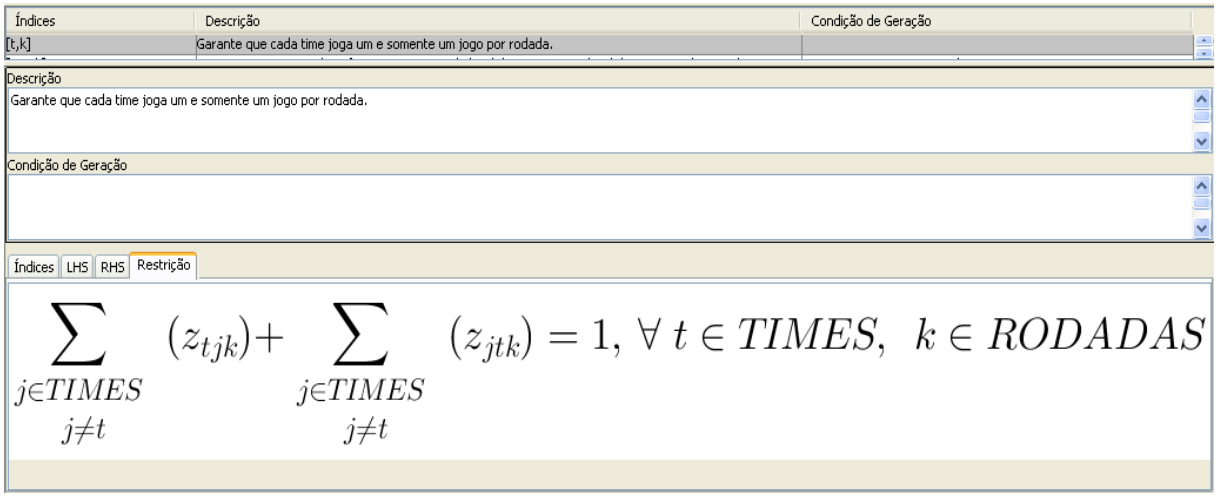


Figura 5.9: Expressão da restrição (15) *do modelo TTPPV com O(n³) variáveis*

A restrição (16) deve ser modificada para sua implementação no GeMM. Como discutido no Capítulo 4, no GeMM as restrições devem ser escritas de forma que se destaque o LHS e o RHS, no qual o RHS só deve conter invariantes, constante e parâmetros, e o LHS deve estar no formato de um somatório de termos compostos pelo produto de uma variável por seu coeficiente. A restrição (16) não está neste formato e deve ser modificada para atender a esse requisito do GeMM. Para isso, basta passar os termos com variáveis do RHS para o LHS, e reescrevê-la da seguinte forma:

$$y_{tij} \geq z_{it,k-1} + z_{jtk} - 1, \quad \forall\, t, i, j = 1,\ldots,n \;\; com \;\; t \neq i \neq j, k = 2,\ldots,n-1 \qquad (16)$$

$$y_{tij} - z_{it,k-1} - z_{jtk} \geq -1, \quad \forall\, t, i, j = 1,\ldots,n \;\; com \;\; t \neq i \neq j, k = 2,\ldots,n-1 \qquad (16')$$

A restrição (16) deve existir para todo *k*, *t*, *i* e *j* desde que $t \neq i \neq j$ e $k > 1$, uma vez que esta restrição não vale para a primeira rodada do torneio. Seus índices são *t, i, j* e *k* e a sua condição de geração é dada por $t \neq i \neq j$ *E* $k > 1$, ou, utilizando os operadores lógicos e de comparação definidos na Tabela 4.2 para a notação do GeMM, *t != i && i != j && t != j && k > 1*, como apresenta a Figura 5.10.

Outra peculiaridade desta restrição é o termo $-z_{it,k-1}$. A indexação deste termo por *k-1,* ao invés de *k*, exige que seja usado um índice alternativo do conjunto *RODADAS* para identificar a rodada *k-1*. Como apresenta a Figura 5.10, o índice *k-1* deste termo foi substituído pelo índice *q* e a condição de geração deste termo na restrição é: *q == k - 1*. Os outros termos do LHS foram modelados da mesma forma que as restrições anteriores. A Figura 5.11 mostra o RHS da restrição (16').

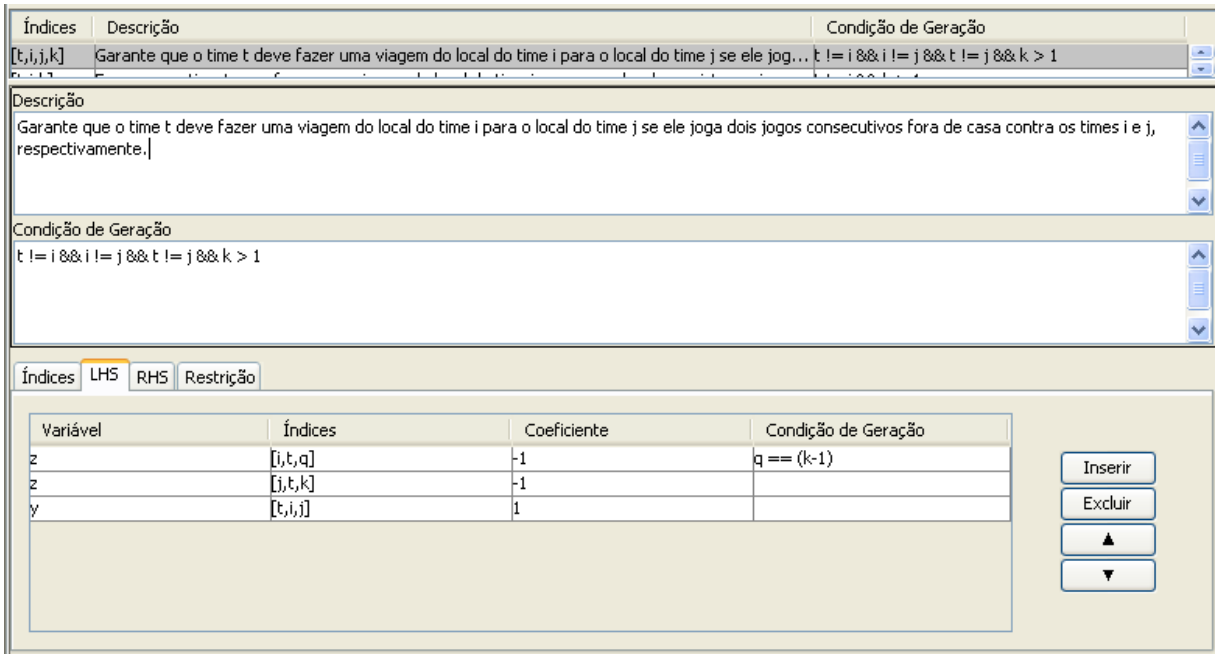


Figura 5.10: Restrição (16) *do modelo TTPPV com O(n^3) variáveis*

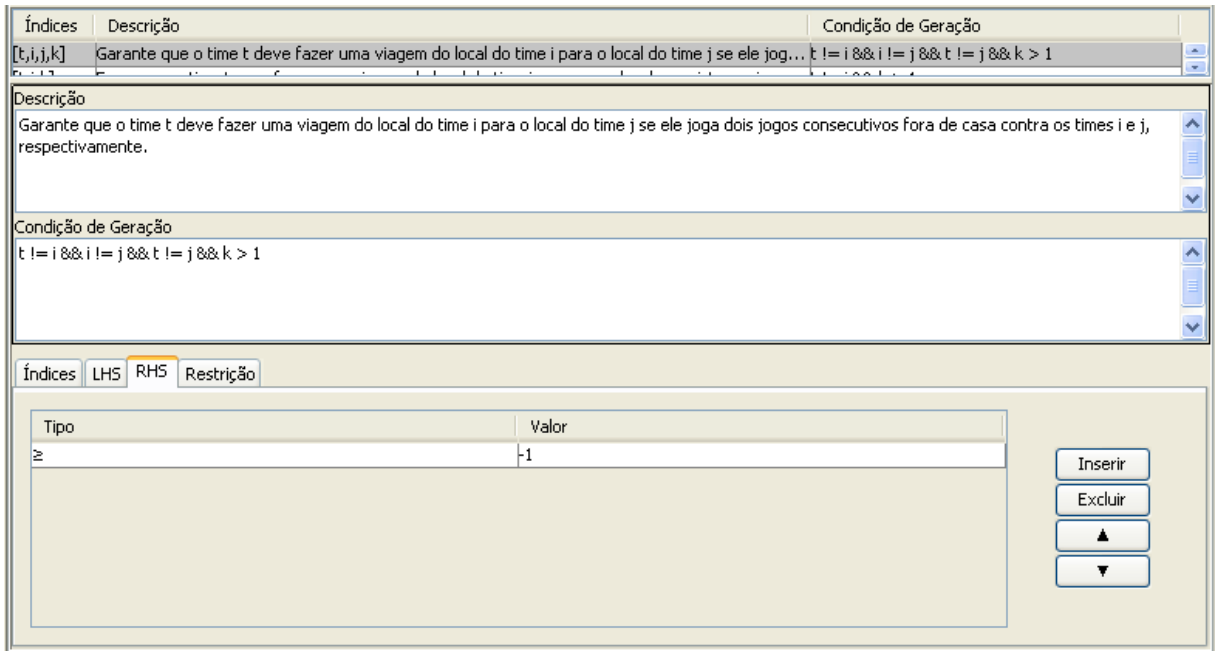


Figura 5.11: RHS da restrição (16) *do modelo TTPPV com O(n^3) variáveis*

A restrição (17) também exige que seja feito o mesmo tratamento da restrição (16). Os dois termos da variável *z* que estão no RHS da restrição devem ser colocados no LHS, deixando apenas a constante no RHS, como na restrição (17') a seguir:

$$y_{tit} \geq z_{it,k-1} + \sum_{\substack{j=1 \\ j \neq t}}^{n} z_{tjk} - 1, \quad \forall t, i = 1,\ldots,n \;\; com \;\; t \neq i, k = 2,\ldots,n-1 \tag{17}$$

$$y_{tit} - z_{it,k-1} - \sum_{\substack{j=1 \\ j \neq t}}^{n} z_{tjk} \geq -1, \quad \forall t, i = 1,\ldots,n \;\; com \;\; t \neq i, k = 2,\ldots,n-1 \tag{17'}$$

A restrição (17) deve existir para todo *t, i* e *k*, desde que $t \neq i$ e $k > 1$, uma vez que esta restrição não vale para a primeira rodada do torneio. Assim, ela é indexada por *t*, *i* e *k* e sua condição de geração é dada por $t \neq i \; E \; k > 1$, ou na notação do GeMM, *t != i && k > 1*, como mostra a Figura 5.12. A modelagem desta restrição é parecida com a restrição anterior: o RHS é o mesmo, *-1,* e o termo $-z_{it,k-1}$ do LHS é modelado da mesma forma. A diferença fica por conta do termo y_{tit}, que nesta restrição é indexado pelo índice *t* duas vezes, e do termo $-z_{tjk}$, que é indexado por *j*, que não indexa a restrição (17) e possui a condição de geração $j \neq t$, como mostra a Figura 5.12.

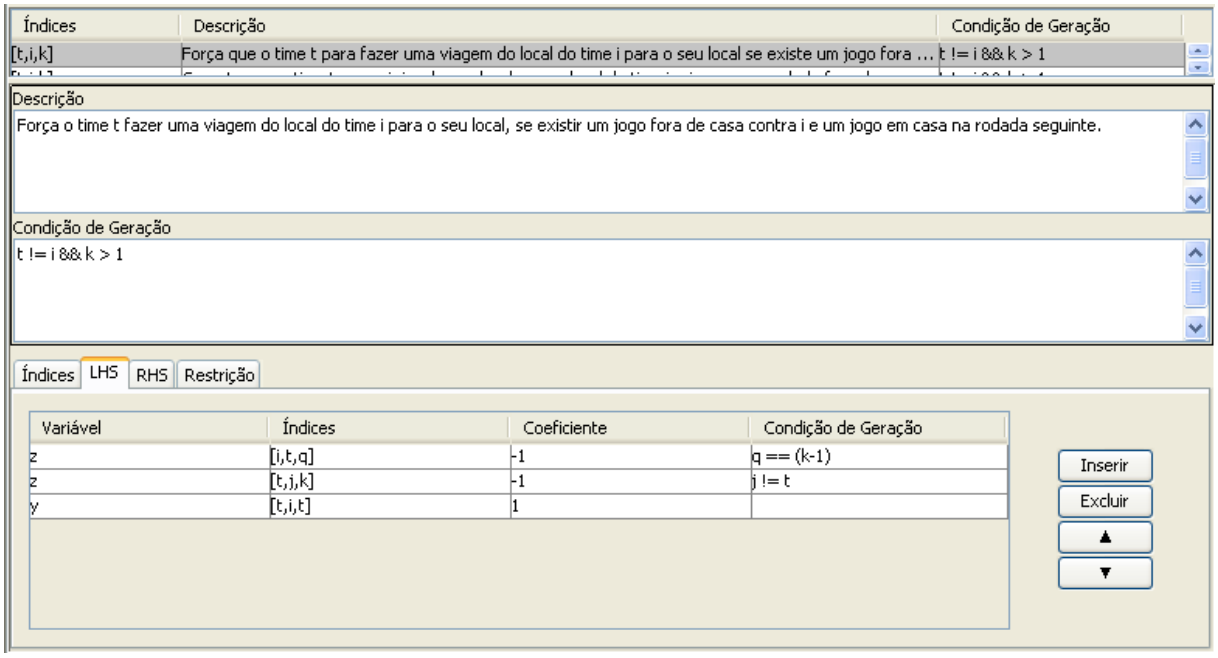


Figura 5.12: Restrição (17) *do modelo TTPPV com O(n^3) variáveis*

A restrição (18) é bastante parecida com a restrição (17), assim sua modelagem no GeMM pode ser feita de forma análoga. A restrição (19) também precisa da transferência do termo z_{it1} para o LHS, ficando da seguinte forma:

$$y_{tti} \geq z_{it1}, \quad \forall t, i = 1,\ldots,n \;\; com \;\; t \neq i \tag{19}$$

$$y_{tti} - z_{it1} \geq 0, \quad \forall t, i = 1, \ldots, n \quad com \quad t \neq i \qquad (19')$$

A Figura 5.13 apresenta a modelagem no GeMM da restrição (19). Ela é indexada por *t* e *i*, e deve ser gerada sempre que $t \neq i$. O termo z_{it1} se refere à primeira rodada, pois nesta restrição este termo só deve existir quando o índice *k*, do conjunto *RODADAS*, for igual a *1*. No caso de uma variável ser indexada por um valor constante, no GeMM esse valor deve entrar na condição de geração do termo na restrição, como mostra a Figura 5.13. O coeficiente deste termo na restrição é *-1*. Conforme a restrição (19'), o RHS da restrição é dado por *0*.

A restrição (20) pode ser modelada no GeMM de forma análoga à restrição (19). Nela o termo $z_{it,n-1}$ também é indexado por uma constante no lugar do índice *k*, onde $k = (n - 1)$, indicando que este termo só existe para a última rodada do torneio.

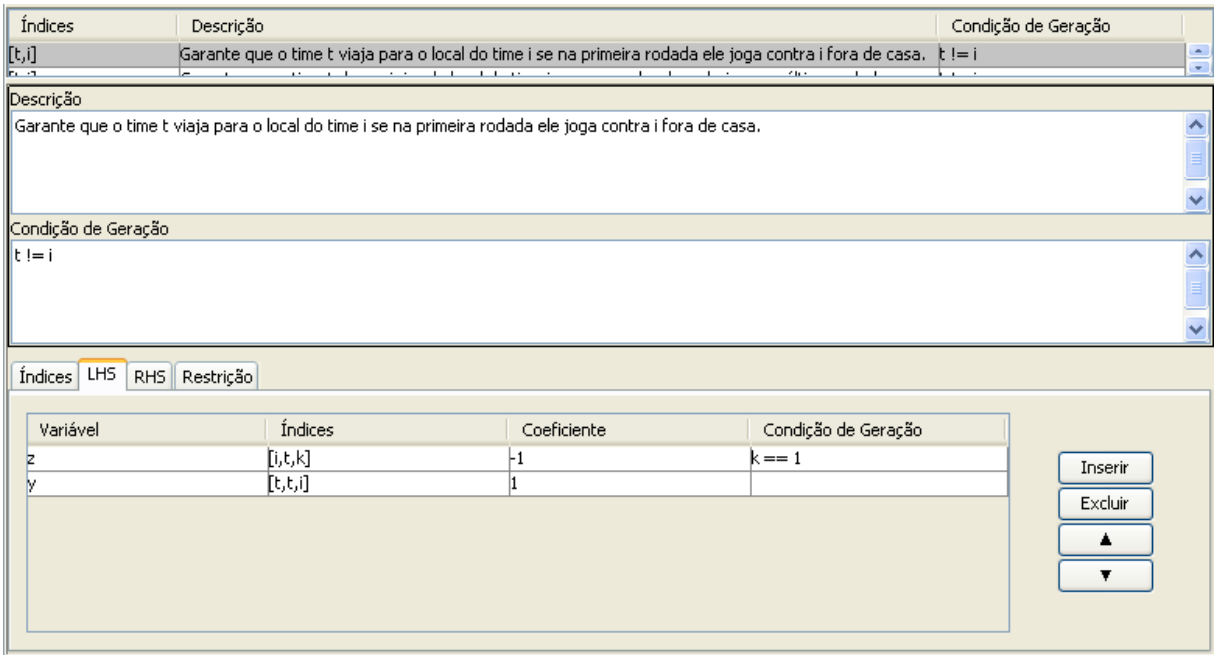


Figura 5.13: Restrição (19) *do modelo TTPPV com* $O(n^3)$ *variáveis*

A restrição (21) existe para todo time *t* e para toda rodada *k*, exceto para as três últimas rodadas. Logo, ela é indexada por *t* e *k*, e sua condição de geração é $k \leq n - 4$, uma vez que o torneio possui *n-1* rodadas, como mostra a Figura 5.14. O equacionamento do LHS desta restrição é composto apenas pelo termo z_{jtq}, que é somado em *q* e em *j*. Na restrição (21), o somatório em *q* é limitado pelo índice *k*. Como visto anteriormente, nas condições de geração do GeMM é possível utilizar os valores dos índices, logo, para modelar este somatório basta incluir a condição de que $q \geq k \; E \; q \leq (k + 3)$ no termo z_{jtq} desta restrição. Assim, para cada rodada *k*, este termo é somado para a rodada atual e para as três próximas. Ainda existe o somatório em *j* que deve respeitar a condição $j \neq t$. Logo, na notação do GeMM, a condição de

geração do termo z_{jtq} na restrição (21) é dada por *j != t && q >= k && q <= (k + 3)*, como exibe a Figura 5.14. O RHS da desigualdade (21) é igual a *3*.

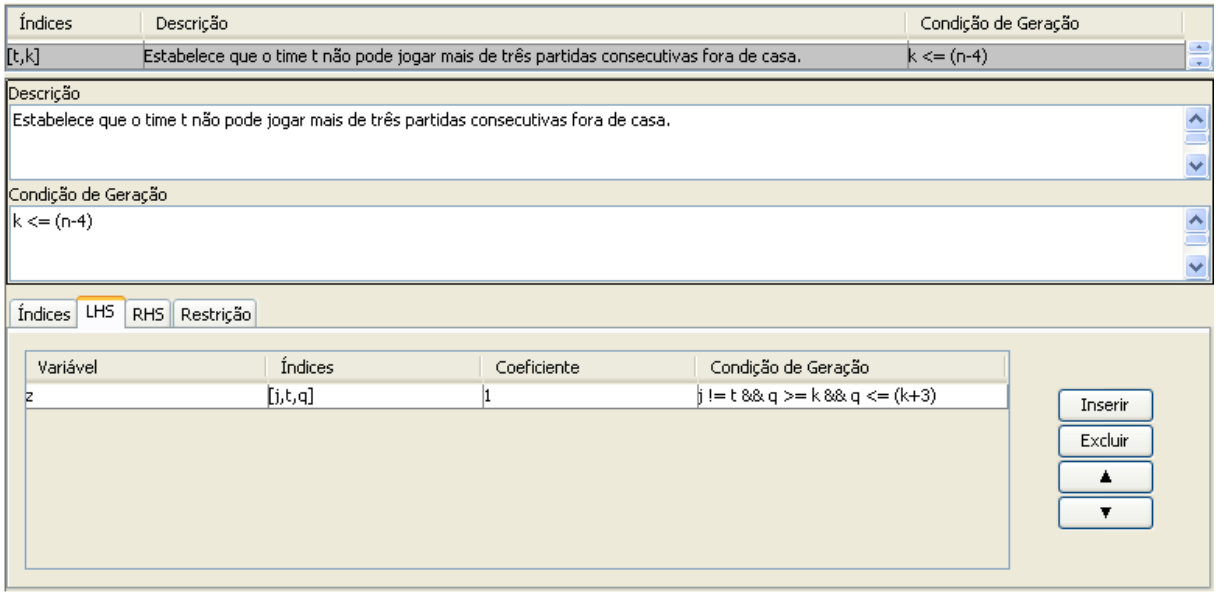


Figura 5.14: Restrição (21) *do modelo TTPPV com $O(n^3)$ variáveis*

A Figura 5.15 apresenta a imagem da restrição (21) gerada pelo GeMM, passando por sua reescrita em LaTeX. A restrição (22) pode ser implementada de forma análoga à restrição (21). Assim, o modelo do TTPPV para $O(n^3)$ variáveis foi completamente modelado no GeMM.

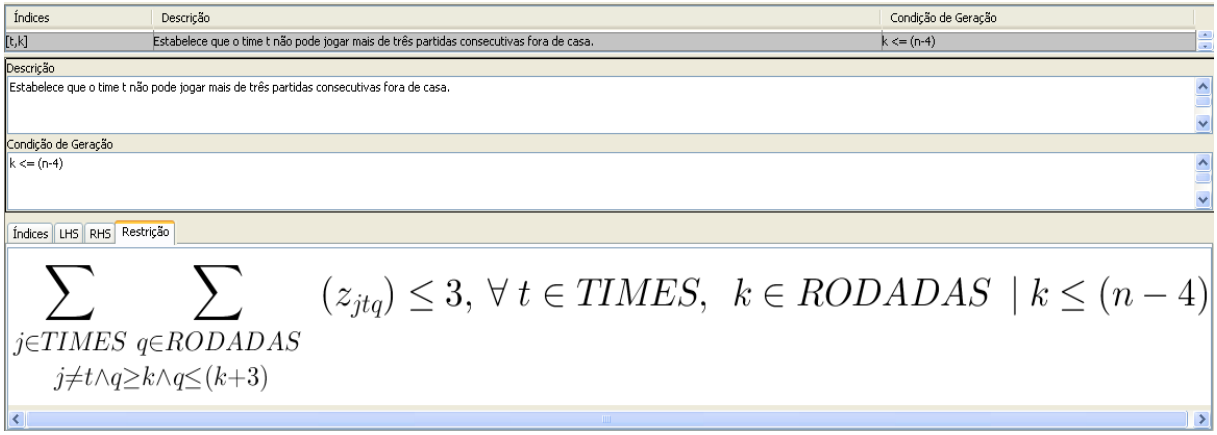


Figura 5.15: Expressão da restrição (21) *do modelo TTPPV com $O(n^3)$ variáveis*

5.2.3 MODELO TTPPV – $O(n^5)$ VARIÁVEIS

No modelo matemático para o TTPPV com $O(n^5)$ variáveis, proposto por MELO *et al.* (2009), as variáveis representam as viagens completas dos times que saem de sua cidade sede

para jogar uma, duas ou três partidas fora de casa, e então retornam para sua cidade. Assim, neste modelo são definidas três variáveis binárias diferentes: w^1_{tik}, que é igual a *1* se o time *t* faz uma viagem para jogar fora de casa com o time *i* na rodada *k* e retorna para casa na rodada *k+1*, senão é igual a *0*; w^2_{tijk}, que é igual a *1* se o time *t* inicia uma viagem para jogar fora de casa contra o time *i* na rodada *k*, em seguida viaja da cidade de *i* para jogar contra o time *j* fora de casa na rodada *k+1* e retorna para casa na rodada *k+2*, senão é igual a *0*; e, por fim, w^3_{tijlk}, que é igual a *1* se o time *t* inicia uma viagem para jogar fora de casa contra o time *i* na rodada *k*, em seguida viaja da cidade de *i* para jogar contra o time *j* fora de casa na rodada *k+1*, na sequência viaja da cidade de *j* para jogar contra o time *l* também fora de casa na rodada *k+2* e retorna para casa na rodada *k+3*, senão é igual a *0*. Nessa formulação é necessário que existam duas rodadas fictícias, a rodada *-1* e a rodada *0*. Logo, as variáveis que indicam viagens que comecem nessas rodadas devem receber valor *0*.

Para calcular a distância total percorrida pelos times são usados os parâmetros c_{ij}, c_{ijm} e c_{ijml}, que representam as distâncias das viagens completas de tamanho um, dois e três, respectivamente:

$$c_{ij} = d_{ij} + d_{ji} \tag{25}$$

$$c_{ijm} = d_{ij} + d_{jm} + d_{mi} \tag{26}$$

$$c_{ijml} = d_{ij} + d_{jm} + d_{ml} + d_{li} \tag{27}$$

A seguir é apresentado o modelo com $O(n^5)$ variáveis para o TTPPV, onde *n* é o número de times participantes do torneio:

$$Minimizar \sum_{k=1}^{n-1} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ (j,i)\in G}}^{n} \left[c_{ij} . w^1_{ijk} + \sum_{\substack{m=1 \\ (m,i)\in G \\ m\neq j}}^{n} \left(c_{ijm} . w^2_{ijmk} + \sum_{\substack{l=1 \\ (l,i)\in G \\ l\neq j\neq m}}^{n} c_{ijml} . w^3_{ijmlk} \right) \right] \tag{28}$$

Sujeito a

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ (j,i)\in G}}^{n} \left[\sum_{k\in\{-1,0\}} w^1_{ijk} + \sum_{\substack{m=1 \\ (m,i)\in G \\ m\neq j}}^{n} \left(\sum_{k\in\{-1,0,n-1\}} w^2_{ijmk} + \sum_{\substack{l=1 \\ (l,i)\in G \\ l\neq j\neq m}}^{n} \sum_{k\in\{-1,0,n-2,n-1\}} w^3_{ijmlk} \right) \right] = 0 \tag{29}$$

$$\sum_{k=1}^{n-1}\left\{w_{ijk}^{1}+\sum_{\substack{m=1\\(m,i)\in G\\m\neq j}}^{n}\left[\left(w_{ijmk}^{2}+w_{imjk}^{2}\right)+\sum_{\substack{l=1\\(l,i)\in G\\l\neq j\neq m}}^{n}\left(w_{ijmlk}^{3}+w_{imjlk}^{3}+w_{imljk}^{3}\right)\right]\right\}=1,\quad \forall(j,i)\in G \tag{30}$$

$$\sum_{\substack{j=1\\(j,i)\in G}}^{n}\left\{w_{ijk}^{1}+\sum_{\substack{m=1\\(m,i)\in G\\m\neq j}}^{n}\left[\left(w_{ijmk}^{2}+w_{ijm,k-1}^{2}\right)+\sum_{\substack{l=1\\(l,i)\in G\\l\neq j\neq m}}^{n}\left(w_{ijmlk}^{3}+w_{ijml,k-1}^{3}+w_{ijml,k-2}^{3}\right)\right]\right\}$$

$$+\sum_{\substack{j=1\\(i,j)\in G}}^{n}\left\{w_{jik}^{1}+\sum_{\substack{m=1\\(m,j)\in G\\m\neq i}}^{n}\left[\left(w_{jimk}^{2}+w_{jmi,k-1}^{2}\right)+\sum_{\substack{l=1\\(l,j)\in G\\l\neq i\neq m}}^{n}\left(w_{jimlk}^{3}+w_{jmil,k-1}^{3}+w_{jmli,k-2}^{3}\right)\right]\right\}=1,$$

$$\forall i=1,\ldots,n,\ \ k=1,\ldots,n-1 \tag{31}$$

$$\sum_{\substack{j=1\\(j,i)\in G}}^{n}\left\{+\sum_{\substack{m=1\\(m,i)\in G\\m\neq j}}^{n}\left[+\sum_{\substack{l=1\\(l,i)\in G\\l\neq j\neq m}}^{n}\begin{array}{c}w_{ijk}^{1}+w_{ij,k+1}^{1}\\ w_{ijm,k-1}^{2}+w_{ijmk}^{2}+w_{ijm,k+1}^{2}\\ \left(w_{ijml,k-2}^{3}+w_{ijml,k-1}^{3}+w_{ijmlk}^{3}+w_{ijml,k+1}^{3}\right)\end{array}\right]\right\}\leq 1$$

$$\forall i=1,\ldots,n,\ \ k=1,\ldots,n-2 \tag{32}$$

$$\sum_{q=k}^{k+3}\left\{\sum_{\substack{j=1\\(j,i)\in G}}^{n}\left[w_{ijq}^{1}+\sum_{\substack{m=1\\(m,i)\in G\\m\neq j}}^{n}\left(\left(w_{ijmq}^{2}+w_{ijm,q-1}^{2}\right)+\sum_{\substack{l=1\\(l,i)\in G\\l\neq j\neq m}}^{n}\left(w_{ijmlq}^{3}+w_{ijml,q-1}^{3}+w_{ijml,q-2}^{3}\right)\right)\right]\right\}\geq 1,$$

$$\forall i=1,\ldots,n,\ \ k=1,\ldots,n-4 \tag{33}$$

$$w_{ijk}^{1}\in\{0,1\},\quad i,j=1,\ldots,n,\ \ com\ \ i\neq j,\ \ k=-1,\ldots,n-1 \tag{34}$$

$$w_{ijmk}^{2}\in\{0,1\},\quad i,j,m=1,\ldots,n,\ \ com\ \ i\neq j\neq m,\ \ k=-1,\ldots,n-1 \tag{35}$$

$$w_{ijmlk}^{3}\in\{0,1\},\quad i,j,m,l=1,\ldots,n,\ \ com\ \ i\neq j\neq m\neq l,\ \ k=-1,\ldots,n-1 \tag{36}$$

A função objetivo (28) busca minimizar a distância percorrida pelos times durante o torneio. Nesta formulação, a função objetivo é escrita usando os parâmetros c_{ij}, c_{ijm} e c_{ijml}, calculados a partir do parâmetro d_{ij}, de acordo com as expressões (25), (26) e (27), respectivamente. O parâmetro d_{ij} é a distância da cidade do time i para a cidade do time j. O

parâmetro c_{ij} é a distância percorrida pelo time *i* em uma viagem completa, saindo da sua cidade para a cidade sede de *j* e retornando para casa em seguida, e sua expressão de cálculo é dada pela expressão (25). Os parâmetros c_{ijm} e c_{ijml} são calculados de forma análoga pelas expressões (26) e (27), respectivamente. A restrição (29) atribui zero para as variáveis associadas às viagens que iniciam nas rodadas fictícias *-1* e *0*, às viagens de tamanho dois e três que começam na última rodada e às viagens de tamanho três que começam na penúltima rodada. A restrição (30) garante que cada jogo ocorre exatamente uma vez. Ela estabelece que cada jogo *(j, i)* em *G* deve ser jogado em uma viagem do time *i* para um, dois ou três jogos fora de casa. A restrição (31) garante que o time *i* ou está jogando fora de casa ou tem outro time que o está visitando em cada rodada, garantindo que os times joguem em todas as rodadas. Essa restrição é construída definindo *1* para a soma de todas as variáveis associadas com as viagens do time *i* na rodada *k* com as viagens dos outros times visitando *i* na rodada *k*. A restrição (32) proíbe que o time *i* esteja envolvido em simultâneas ou consecutivas viagens na rodada *k*, isto é, sem retornar para sua cidade sede. A restrição (33) determina que o time *i* deve estar fora de casa para jogar pelo menos uma vez a cada quatro rodadas consecutivas. As restrições (34), (35) e (36) são as restrições de integralidade das variáveis binárias *w* e definem as condições em que elas devem existir no modelo.

Esta formulação para o problema do TTPPV apresenta expressões mais complexas do que a formulação com $O(n^3)$ variáveis. Porém, a maior parte das funcionalidades do GeMM, necessárias para implementar esse modelo, já foram apresentadas. As principais novidades desse modelo, em relação aos modelos matemáticos já descritos neste trabalho, são os parâmetros calculados a partir de parâmetros primários e a utilização de condições de geração para as variáveis. Nesta formulação, todas as expressões das restrições já estão escritas com as variáveis no LHS e apenas constantes no RHS.

5.2.4 IMPLEMENTAÇÃO DO MODELO TTPPV – $O(n^5)$ VARIÁVEIS

Para modelar este problema no GeMM foram usados os mesmos dois conjuntos da formulação anterior, *TIMES* e *RODADAS*. A diferença é que o conjunto das rodadas do torneio deve ter dois elementos adicionais, as rodadas fictícias *-1* e *0*. Logo, este conjunto tem *n + 1* elementos. O conjunto *TIMES*, além dos índices *t*, *i* e *j*, passa a ter os índices *l* e *m* para auxiliar a modelagem. O conjunto *RODADAS* continua com os mesmos índices *k* e *q*. Os parâmetros primários utilizados neste modelo são os mesmos da formulação anterior: *n*, número de times, *G*, que representa os jogos a serem programados e seus locais de realização, e d_{ij}, matriz de distâncias entre as cidades. Esses parâmetros representam os dados de entrada

do problema e seus valores devem ser informados para que o problema seja otimizado por um resolvedor.

Além dos parâmetros primários, essa formulação utiliza os parâmetros calculados c_{ij}, c_{ijm} e c_{ijml}. A Figura 5.16 apresenta a definição do parâmetro c_{ijml} e a sua expressão de cálculo. Esse parâmetro é indexado pelos índices i, j, m e l do conjunto *TIMES* e a sua expressão de cálculo no GeMM é dada por *d[i,j] + d[j,m] + d[m,l] + d[l,i]*, como na expressão (27). Os parâmetros c_{ij} e c_{ijm} são calculados de forma análoga. Quando for criado um cenário de dados, antes de gerar uma instância do problema para a otimização em um resolvedor, o GeMM avalia as expressões dos parâmetros calculados para obter seus valores.

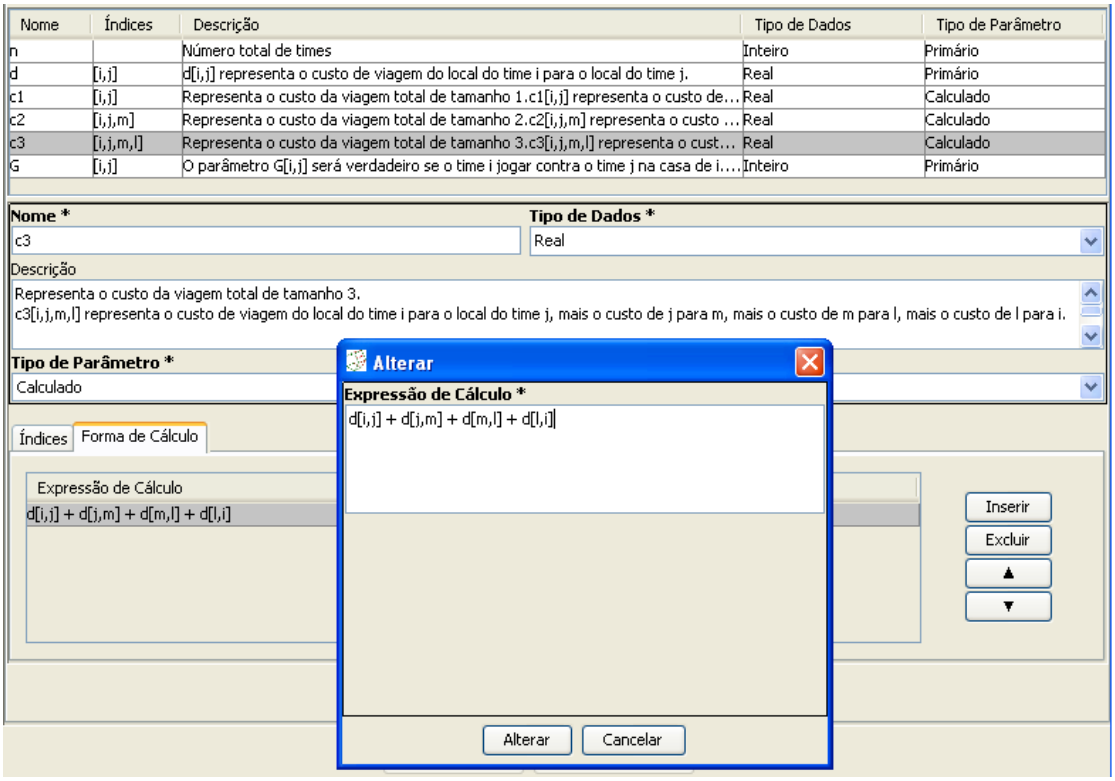


Figura 5.16: Parâmetros do modelo TTPPV com O(n^5) variáveis

A Figura 5.17 mostra a implementação no GeMM da variável w^3_{ijmlk} do modelo matemático. Ela é indexada por i, j, m e l, do conjunto *TIMES*, e k, do conjunto *RODADAS*. Diferentemente das variáveis implementadas no GeMM anteriormente, esta possui uma condição de geração. Conforme a restrição (36), w^3_{ijmlk} só deve existir quando $i \neq j \neq m \neq l$, ou seja, os times associados aos índices i, j, m e l devem ser todos diferentes. A Figura 5.17 ainda mostra a definição das variáveis w^2_{ijmk} e w^1_{ijk}.

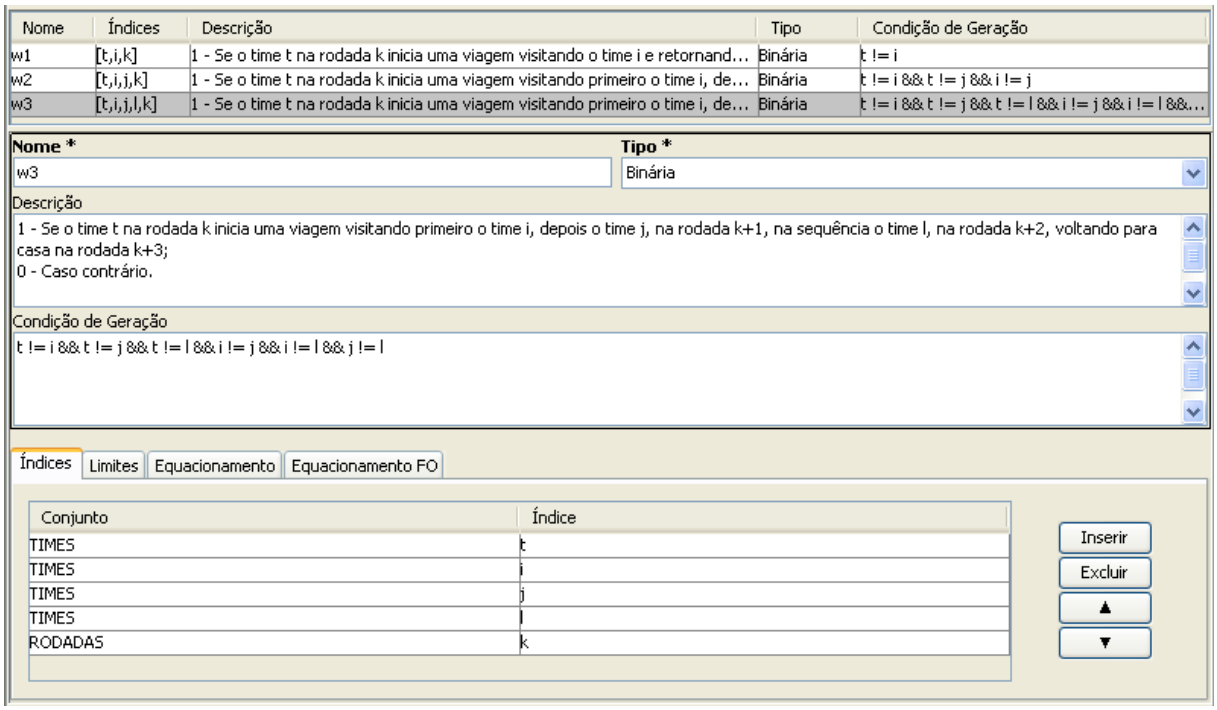


Figura 5.17: Variáveis do modelo TTPPV com $O(n^5)$ variáveis

Uma vez definidos os parâmetros e as variáveis, a função objetivo (28) pode ser implementada no GeMM, como exibe a Figura 5.18. O primeiro termo é o da variável w^1_{ijk}, seu coeficiente é c_{ij} e ele deve ser somado sempre que o jogo entre o time *j* e o time *i* na casa do time *j* existir, ou seja, $(j,i) \in G$, ou na notação do GeMM, *G[j,i] == 1*, para todo *i* e *j*. A soma em *k* deve ser feita apenas quando $k \geq 1$, uma vez que as rodadas *0* e *-1* são fictícias. O segundo termo da função objetivo, com a variável w^2_{ijmk}, é semelhante, exceto pela presença do índice *m*. Seu coeficiente é c_{ijm} e a soma deve ser feita sempre que existir o jogo entre *j* e *i* na casa do time *j* e existir o jogo entre *m* e *i* na casa do time *m*, ou seja, na notação do GeMM, *G[j,i] == 1 && G[m,i] == 1*. Por fim, o termo contendo a variável w^3_{ijmlk}, cujo coeficiente é c_{ijml}. Além de ser somado sobre os índices *k*, *i*, *j* e *m*, deve ser somado também sobre o índice *l*. Assim, além dos jogos do time *i* contra os times *j* e *m*, ambos fora de casa, deve também existir o jogo entre *l* e *i*. A condição deste termo no GeMM é dada por *G[j,i] == 1 && G[m,i] == 1 && G[l,i] == 1*.

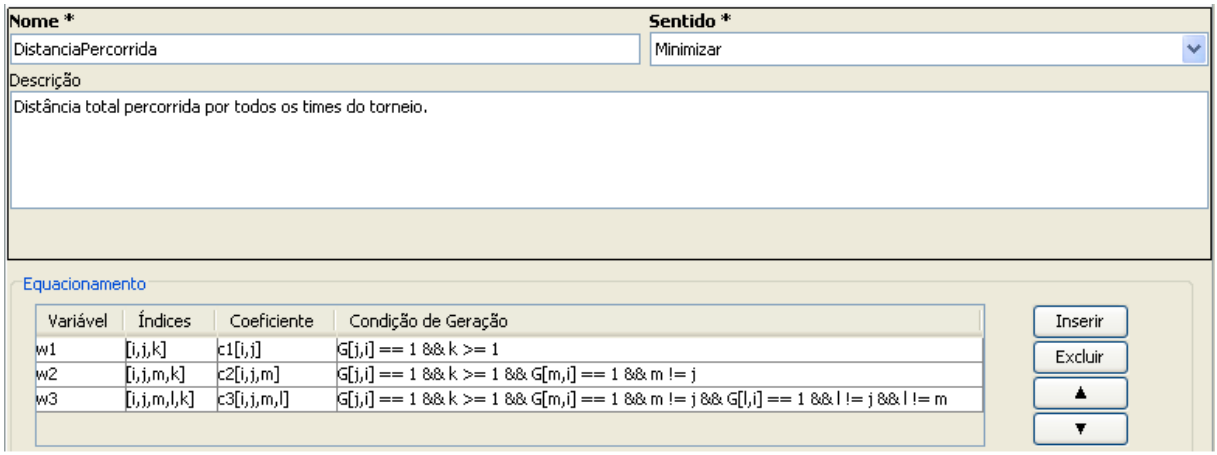


Figura 5.18: Função objetivo do modelo TTPPV com $O(n^5)$ variáveis

A Figura 5.19 mostra a modelagem da restrição (29). Nesta restrição existem três termos: um para cada variável, w^1_{ijk}, w^2_{ijmk} e w^3_{ijmlk}, todas com o coeficiente *1*. O termo da variável w^1_{ijk} deve ser somado sobre os índices *i* e *j*, respeitando a condição de que o jogo entre os times *j* e *i* ocorre na casa do time *j*, e sobre o índice *k*, desde que $k = -1$ e $k = 0$. O termo da variável w^2_{ijmk} deve ser somado sobre os índices *i* e *j*, respeitando a mesma condição anterior, sobre o índice *m*, desde que o jogo entre os times *m* e *i* ocorra na casa do time *m*, e sobre o índice *k*, apenas para as rodadas $k = -1$, $k = 0$ e $k = n - 1$. Por último, o termo da variável w^3_{ijmlk} deve ser somado sobre os mesmos índices da variável w^2_{ijmk}, acrescentando o índice *l*, desde que o jogo entre os times *l* e *i* ocorra na casa de *l*, e o índice *k* deve ser somado para as rodadas -1, 0, $n - 1$ e $n - 2$.

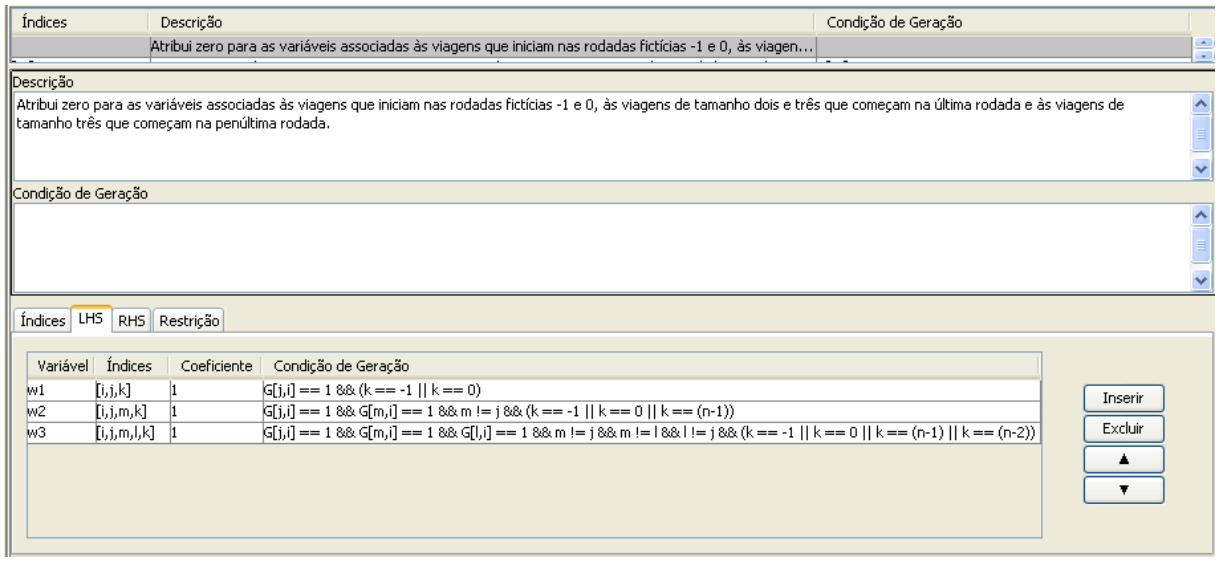


Figura 5.19: Restrição (29) *do modelo TTPPV com $O(n^5)$ variáveis*

O RHS desta restrição de igualdade é zero, uma vez que as viagens associadas às rodadas fictícias *-1* e *0* não devem existir, as viagens de tamanho dois não podem iniciar na última rodada e as viagens de tamanho três não podem iniciar nem na última, nem na penúltima rodada. A Figura 5.20 exibe a imagem da expressão da restrição (29) gerada pelo GeMM.

$$\sum_{\substack{i\in TIMES \\ G_{ji}=1\wedge(k=-1\vee k=0)}} \sum_{j\in TIMES} \sum_{k\in RODADAS} (w^1{}_{ijk})$$
$$+ \sum_{\substack{i\in TIMES \\ G_{ji}=1\wedge G_{mi}=1\wedge m\neq j\wedge(k=-1\vee k=0\vee k=(n-1))}} \sum_{j\in TIMES} \sum_{m\in TIMES} \sum_{k\in RODADAS} (w^2{}_{ijmk})$$
$$+ \sum_{\substack{i\in TIMES \\ G_{ji}=1\wedge G_{mi}=1\wedge G_{li}=1\wedge m\neq j\wedge m\neq l\wedge l\neq j\wedge(k=-1\vee k=0\vee k=(n-1)\vee k=(n-2))}} \sum_{j\in TIMES} \sum_{m\in TIMES} \sum_{l\in TIMES} \sum_{k\in RODADAS} (w^3{}_{ijmlk}) = 0$$

Figura 5.20: Expressão da restrição (29) *do modelo TTPPV com O(n^5) variáveis*

Neste modelo do TTPPV com *O(n^5)* variáveis foi utilizada apenas a restrição (29) para ilustrar a implementação das restrições no GeMM, uma vez que todas as outras podem ser modeladas de forma análoga. Todas as características do GeMM necessárias para modelar as restrições (30), (31), (32) e (33) já foram apresentadas neste capítulo ou no capítulo anterior.

5.2.5 CENÁRIO DE DADOS

Para testar a correta implementação no GeMM dos modelos descritos para o TTPPV, elaborou-se um cenário de teste de um torneio com seis times. Este número justifica-se por ser o menor número de *n* para que todas as restrições sejam válidas, uma vez que um torneio com quatro times possui apenas três rodadas. Logo, as restrições para garantir que o mesmo time não jogue mais de três vezes em casa ou fora de casa consecutivamente não existem. Na Tabela 5.1 estão os times participantes deste torneio de teste.

Tabela 5.1: Times do cenário de teste do TTPPV

Time	Cidade
Flamengo	Rio de Janeiro
São Paulo	São Paulo
Grêmio	Porto Alegre
Cruzeiro	Belo Horizonte
Guarani	Campinas
Vitória	Salvador

Na Tabela 5.2 estão as distâncias entre as cidades dos times participantes do torneio, de acordo com DNIT (2011), e na Tabela 5.3 estão os jogos e os locais onde devem ocorrer. No TTPPV os jogos dos times e os locais onde eles ocorrem são dados de entrada do problema. O problema de otimização deve decidir em quais rodadas cada jogo deve acontecer para que a distância total viajada por todos os times do torneio seja minimizada.

Tabela 5.2: Matriz de distância entre as cidades dos times do cenário de teste do TTPPV

(km)	**Rio de Janeiro**	**São Paulo**	**Porto Alegre**	**Belo Horizonte**	**Campinas**	**Salvador**
Rio de Janeiro	0	429	1553	434	511	1649
São Paulo	429	0	1109	586	99	1962
Porto Alegre	1553	1109	0	1712	1177	3127
Belo Horizonte	434	586	1712	0	601	1372
Campinas	511	99	1177	601	0	1982
Salvador	1649	1962	3127	1372	1982	0

Uma vez definidos os modelos matemáticos, o GeMM gera as telas para entrada dos dados e obtenção dos resultados dos cenários. Como as definições dos conjuntos e parâmetros são as mesmas para os dois modelos matemáticos apresentados para o TTPPV, as telas dos dados de entrada no GeMM também são semelhantes. A Figura 5.21 exibe a tela do GeMM para controle de versão dos cenários de dados do modelo matemático do TTPPV com $O(n^5)$ variáveis. Nesta tela é possível ver que este modelo possui um cenário de dados para o torneio com seis times, cujos dados da versão um estão apresentados nas Tabelas 5.1, 5.2 e 5.3.

Tabela 5.3: Jogos pré-definidos do cenário de teste do TTPPV

Jogo	Local
Flamengo x Cruzeiro	Rio de Janeiro
São Paulo x Cruzeiro	São Paulo
Guarani x Vitória	Campinas
Grêmio x Flamengo	Porto Alegre
Grêmio x Guarani	Porto Alegre
Vitória x Cruzeiro	Salvador
Flamengo x São Paulo	Rio de Janeiro
Flamengo x Guarani	Rio de Janeiro
Vitória x São Paulo	Salvador
Cruzeiro x Grêmio	Belo Horizonte
Flamengo x Vitória	Rio de Janeiro
Cruzeiro x Guarani	Belo Horizonte
São Paulo x Grêmio	São Paulo
São Paulo x Guarani	São Paulo
Grêmio x Vitória	Porto Alegre

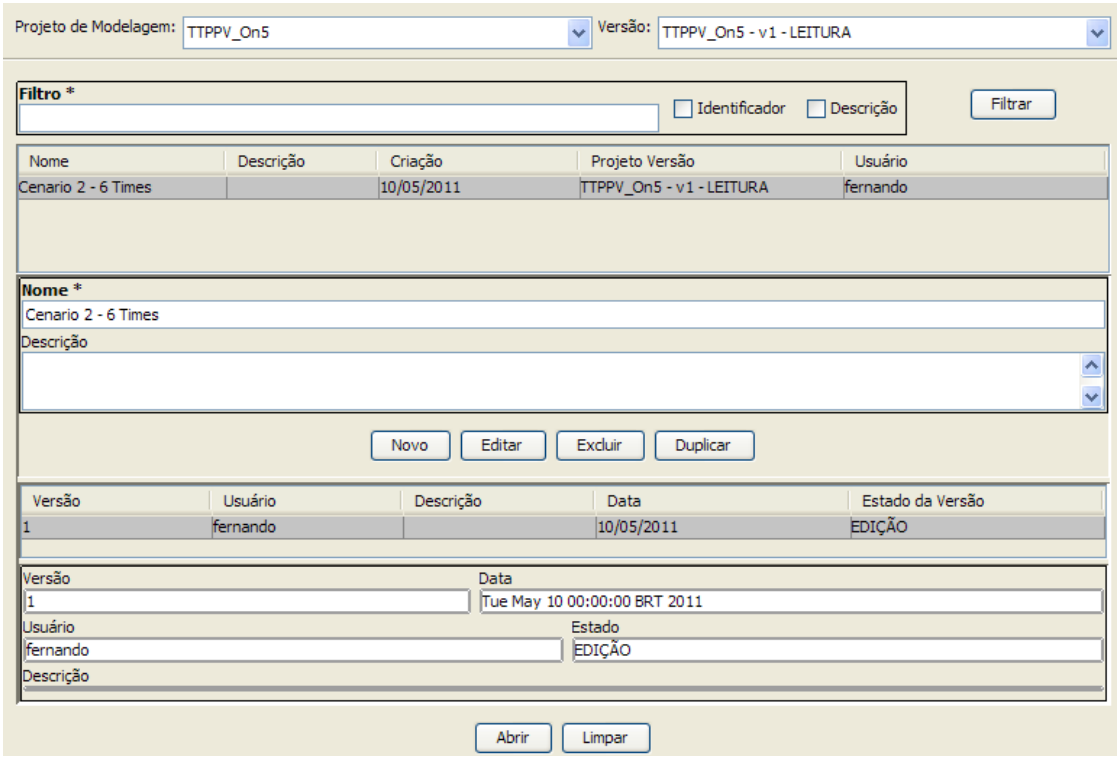


Figura 5.21: Controle dos cenários de dados do TTPPV

A Figura 5.22 apresenta os elementos do conjunto *RODADAS* na tela do GeMM desta versão do cenário de dados. Como são seis times, devem existir cinco rodadas neste torneio,

porém na modelagem do TTPPV com $O(n^5)$ variáveis, além das cinco rodadas, devem existir as rodadas fictícias *-1* e *0*. A Figura 5.23 mostra os elementos do conjunto *TIMES*.

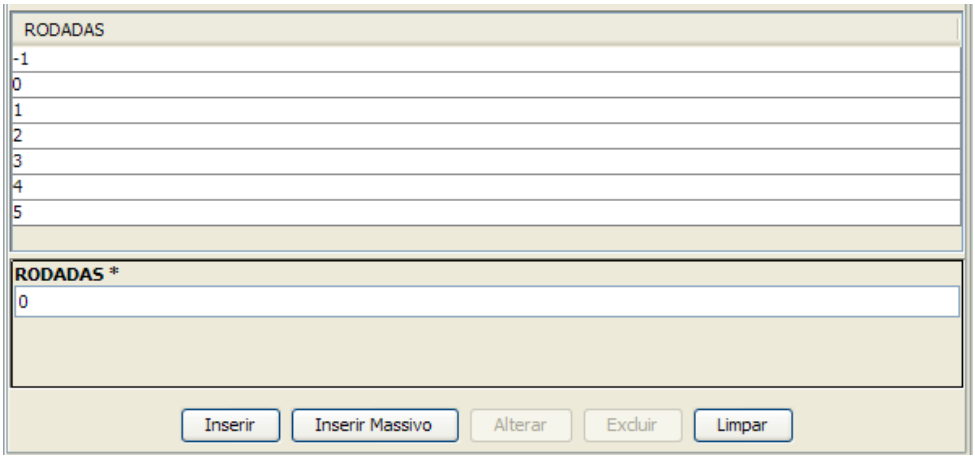


Figura 5.22: Rodadas do cenário de dados do TTPPV

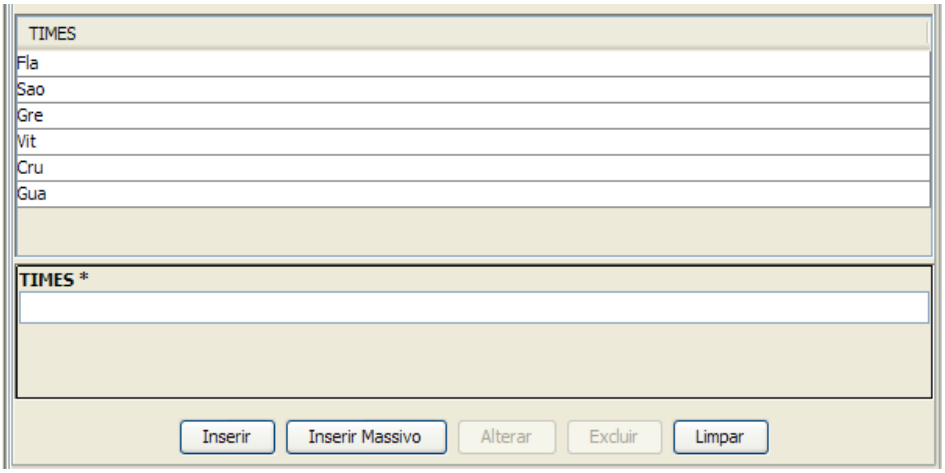


Figura 5.23: Times do cenário de dados do TTPPV

A Figura 5.24 apresenta a tela do GeMM com os valores dos parâmetros primários do cenário de dados do modelo do TTPPV, ou seja, as informações que estão na Tabela 5.2 e na Tabela 5.3. Para cada par de times é definido se existe ou não o jogo entre eles na cidade do primeiro time, representado pelo parâmetro *G[i,j]*, e a matriz de distância entre as cidades dos times, representada pelo parâmetro *d[i,j]*.

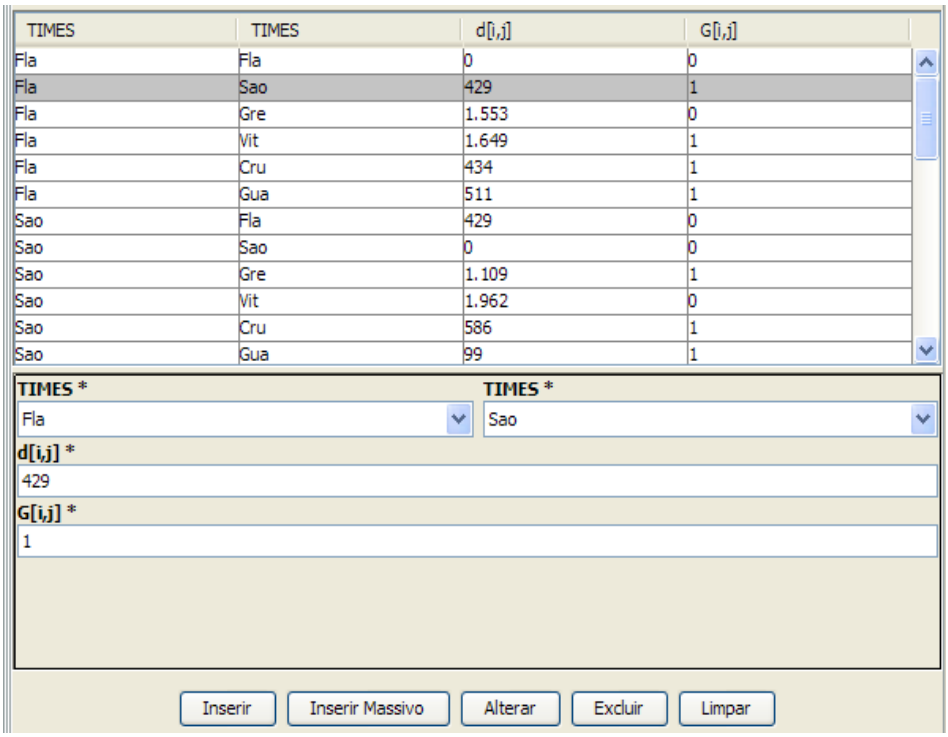


Figura 5.24: Parâmetros do cenário de dados do TTPPV

Uma vez definidos os dados de entrada, o cenário pode ser otimizado. Como mostrado no Capítulo 4, o GeMM gera uma instância de um PPLI a partir de uma versão do modelo matemático e de uma versão de um cenário de dados, passa essa instância automaticamente para o resolvedor e obtém os resultados. Executando o problema de otimização através do resolvedor CPLEX versão 12.2 obteve-se o valor de função objetivo de 25.173, que representa a distância total viajada por todos os times em quilômetros, conforme a tela de resumo do resultado exibida na Figura 5.25. Este resultado foi obtido para ambos os modelos matemáticos com os dados apresentados. A Figura 5.26 mostra uma tela com o resultado das variáveis do problema, neste caso, a variável w^2 do modelo do TTPPV com $O(n^5)$ variáveis.

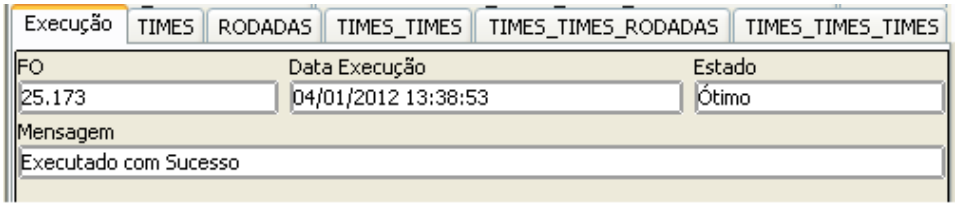
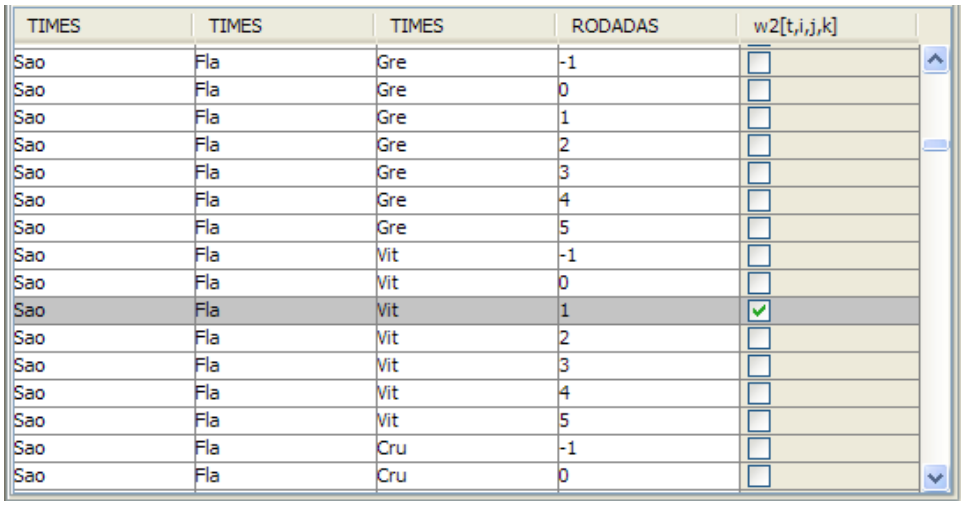


Figura 5.25: Resumo do resultado do cenário de dados do TTPPV



TIMES	TIMES	TIMES	RODADAS	w2[t,i,j,k]
Sao	Fla	Gre	-1	☐
Sao	Fla	Gre	0	☐
Sao	Fla	Gre	1	☐
Sao	Fla	Gre	2	☐
Sao	Fla	Gre	3	☐
Sao	Fla	Gre	4	☐
Sao	Fla	Gre	5	☐
Sao	Fla	Vit	-1	☐
Sao	Fla	Vit	0	☐
Sao	Fla	Vit	1	☑
Sao	Fla	Vit	2	☐
Sao	Fla	Vit	3	☐
Sao	Fla	Vit	4	☐
Sao	Fla	Vit	5	☐
Sao	Fla	Cru	-1	☐
Sao	Fla	Cru	0	☐

Figura 5.26: Valores das variáveis do cenário de dados do TTPPV

Este mesmo cenário de dados foi otimizado no GeMM para o modelo do TTPPV com $O(n^3)$ variáveis e o modelo com $O(n^5)$ variáveis. A Tabela 5.4 e a Tabela 5.5 apresentam os resumos das instâncias geradas pelo GeMM e os tempos de otimização. O tempo total é o tempo gasto para realizar todas as tarefas envolvidas na otimização de um cenário, que são: leitura das informações do modelo e do cenário do banco de dados; geração da instância do problema e sua passagem para o resolvedor; execução da otimização pelo resolvedor; e leitura dos resultados e gravação destes no banco de dados. O tempo utilizado apenas pelo GeMM é contabilizado pelo tempo total subtraído do tempo gasto pelo resolvedor.

Tabela 5.4: Resumo da execução do modelo TTPPV com $O(n^3)$ variáveis

Total de variáveis geradas	396
Total de restrições geradas	864
Estado	Ótimo
Valor FO	25.173
Tempo total de otimização (segundos)	2,370
Tempo utilizado pelo GeMM (segundos)	1,154 (48,69%)

Tabela 5.5: Resumo da execução do modelo TTPPV com $O(n^5)$ variáveis

Total de variáveis geradas	3.570
Total de restrições geradas	82
Estado	Ótimo
Valor FO	25.173
Tempo total de otimização (segundos)	6,578
Tempo utilizado pelo GeMM (segundos)	6,485 (98,59%)

A Tabela 5.6 e a Tabela 5.7 exibem os valores das variáveis *z* e *y* do modelo TTPPV com $O(n^3)$ variáveis. A variável *z* representa a própria tabela do torneio, resultado da otimização desse modelo matemático. Já a *y* representa as viagens feitas pelos times. Por exemplo, ao se observar os jogos do Flamengo nas três primeiras rodadas, na Tabela 5.6 é possível ver que ele joga em casa contra o Cruzeiro, fora de casa contra o Grêmio e volta a jogar em casa contra o Vitória. Nas duas primeiras linhas da Tabela 5.7 estão representadas as viagens que o Flamengo deve fazer para jogar essas três rodadas. A primeira linha indica que este deve viajar da sua cidade para a cidade do time do Grêmio. E a segunda linha indica que o Flamengo deve viajar da cidade do time do Grêmio para a sua cidade.

Tabela 5.6: Resultado da variável *z* do modelo TTPPV com $O(n^3)$ variáveis

z	***t***	***j***	***k***
1	Flamengo	Cruzeiro	1
1	São Paulo	Guarani	1
1	Grêmio	Vitória	1
1	São Paulo	Cruzeiro	2
1	Grêmio	Flamengo	2
1	Guarani	Vitória	2
1	Flamengo	Vitória	3
1	São Paulo	Grêmio	3
1	Cruzeiro	Guarani	3
1	Flamengo	Guarani	4
1	Vitória	São Paulo	4
1	Cruzeiro	Grêmio	4
1	Flamengo	São Paulo	5
1	Grêmio	Guarani	5
1	Vitória	Cruzeiro	5

Tabela 5.7: Resultado da variável *y* do modelo TTPPV com $O(n^3)$ variáveis

y	***t***	***i***	***j***
1	Flamengo	Flamengo	Grêmio
1	Flamengo	Grêmio	Flamengo
1	São Paulo	Flamengo	São Paulo
1	São Paulo	São Paulo	Vitória
1	São Paulo	Vitória	Flamengo
1	Grêmio	São Paulo	Cruzeiro
1	Grêmio	Grêmio	São Paulo
1	Grêmio	Cruzeiro	Grêmio
1	Vitória	Flamengo	Vitória

y	*t*	*i*	*j*
1	Vitória	Grêmio	Guarani
1	Vitória	Vitória	Grêmio
1	Vitória	Guarani	Flamengo
1	Cruzeiro	Flamengo	São Paulo
1	Cruzeiro	São Paulo	Cruzeiro
1	Cruzeiro	Vitória	Cruzeiro
1	Cruzeiro	Cruzeiro	Flamengo
1	Cruzeiro	Cruzeiro	Vitória
1	Guarani	Flamengo	Grêmio
1	Guarani	São Paulo	Guarani
1	Guarani	Grêmio	Guarani
1	Guarani	Cruzeiro	Flamengo
1	Guarani	Guarani	São Paulo
1	Guarani	Guarani	Cruzeiro

Nas Tabelas 5.8, 5.9 e 5.10 estão os valores das variáveis w^1, w^2 e w^3, respectivamente. Elas são o resultado do modelo TTPPV com $O(n^5)$ variáveis para o cenário de dados apresentado. Como elas representam as viagens de tamanho um, dois e três, a tabela do torneio é uma informação que deve ser obtida através da combinação dos valores dessas variáveis. Assim, na Tabela 5.11 está a tabela do torneio obtida como resultado deste modelo e deste cenário de dados.

Tabela 5.8: Resultado da variável w^1 do modelo TTPPV com $O(n^5)$ variáveis

w^1	*t*	*i*	*k*
1	Flamengo	Grêmio	4
1	Cruzeiro	Vitória	1
1	Guarani	São Paulo	5

Tabela 5.9: Resultado da variável w^2 do modelo TTPPV com $O(n^5)$ variáveis

w^2	*t*	*i*	*j*	*k*
1	São Paulo	Flamengo	Vitória	1
1	Grêmio	Cruzeiro	São Paulo	2
1	Cruzeiro	São Paulo	Flamengo	4

Tabela 5.10: Resultado da variável w^3 do modelo TTPPV com $O(n^5)$ variáveis

w^3	*t*	*i*	*j*	*l*	*k*
1	Vitória	Flamengo	Guarani	Grêmio	3
1	Guarani	Grêmio	Flamengo	Cruzeiro	1

Tabela 5.11: Tabela do torneio montada a partir dos valores das variáveis w^1, w^2 e w^3

Time Casa	Time Visitante	Rodada
Flamengo	São Paulo	1
Vitória	Cruzeiro	1
Grêmio	Guarani	1
Flamengo	Guarani	2
Vitória	São Paulo	2
Cruzeiro	Grêmio	2
Flamengo	Vitória	3
São Paulo	Grêmio	3
Cruzeiro	Guarani	3
Grêmio	Flamengo	4
São Paulo	Cruzeiro	4
Guarani	Vitória	4
Flamengo	Cruzeiro	5
São Paulo	Guarani	5
Grêmio	Vitória	5

Ao se comparar a tabela do torneio obtida pelo modelo TTPPV com $O(n^5)$ variáveis, mostrada na Tabela 5.11, com a tabela do torneio retornada pelo modelo TTPPV com $O(n^3)$ variáveis, mostrada na Tabela 5.6, é possível observar que os modelos chegaram a soluções diferentes com o mesmo valor de função objetivo. Isso ocorre pois neste problema, os times começam em suas cidades e retornam para elas ao final da última rodada e a matriz de distância é simétrica, ou seja, $d_{ij} = d_{ji}$. Assim, se trocarmos a última rodada pela primeira, a penúltima pela segunda, e assim por diante, sempre é obtida uma solução com o mesmo custo.

Para validar que os dois modelos matemáticos foram implementados de forma correta e que o GeMM gerou corretamente a instância do PPLI equivalente, ambos os modelos foram implementados para este mesmo cenário de dados no AIMMS e também otimizados com o CPLEX. Os resultados obtidos foram os mesmo apresentados anteriormente, tanto o valor da função objetivo quanto os valores das variáveis.

5.3 PROBLEMA DE PROGRAMAÇÃO DE ENTREGA DE PEDIDOS

Outro problema de otimização utilizado na validação do GeMM foi o problema de programação de entrega de pedidos de produtos derivados de petróleo. Uma subsidiária da Petrobras é responsável por fazer a entrega de produtos para seus clientes e garantir que a demanda seja atendida. A entrega dos produtos é feita por meio rodoviário e o objetivo deste problema é programar as entregas respeitando a demanda e os limites de estoque dos clientes,

e os limites de capacidade de transporte, de forma a minimizar os custos envolvidos nessas entregas. Ele deve ser resolvido para um determinado horizonte de tempo.

O modelo matemático deste problema foi desenvolvido com base no sistema de gerenciamento de estoque apresentado por BERTAZZI *et al.* (2005), cuja principal característica é o controle dos estoques dos clientes sendo feito pelo centro de distribuição dos produtos. Nesta política de gerenciamento de estoque, o centro de distribuição dos produtos conhece os níveis de estoque e as demandas dos seus clientes. Assim, ele pode determinar a política de transporte de forma a atender às restrições de estoque e demanda dos seus clientes, buscando minimizar os custos totais de transporte. BERTAZZI *et al.* (2005) mostram que esta política de gerenciamento de estoque reduz significantemente o custo de transporte se comparado ao modelo tradicional, no qual cada cliente gerencia seu estoque isoladamente.

O problema de programação de entrega de pedidos é um problema de programação linear e inteira. Em sua modelagem, dividiu-se o horizonte de tempo em períodos discretos, no qual cada um deles é referenciado pelo índice *t* do conjunto *PERÍODOS*. Na prática, o horizonte que se quer programar vai de 7 a 30 dias e cada período é de um dia. Outros índices utilizados são *c*, que representa um cliente do conjunto *CLIENTES,* e *p*, que representa um produto do conjunto PRODUTOS. As principais variáveis de decisão são $Entrega_{pct}$, que indica a quantidade em quilogramas (Kg) que deve ser entregue do produto *p* para o cliente *c* no período *t;* $Estoque_{pct}$, que indica a quantidade (Kg) em estoque do produto *p* no cliente *c* no período *t*; e $SeEntrega_{pct}$, variável binária para indicar se existe ou não entrega do produto *p* para o cliente *c* no período *t*. Além dessas variáveis, são definidas as variáveis *MaiorEntrega* e *MaiorNúmeroEntregas* para auxiliar na modelagem da função objetivo. *MaiorEntrega* é a variável que indica a maior quantidade (Kg) entregue em um mesmo período, ao considerar todos os produtos e clientes. *MaiorNúmeroEntregas* é uma variável inteira para indicar o maior número de entregas feito em um único período entre todos os períodos.

Os dados de entrada para este problema são fornecidos pelos seguintes parâmetros: $Demanda_{pct}$, demanda de cada produto para cada cliente em cada período; $CapacidadeTransporte_t$, quantidade máxima que pode ser transportada por período; $EstoqueInicial_{pc}$, estoque inicial de cada produto em cada cliente; $EstoqueMínimo_{pc}$ e $EstoqueMáximo_{pc}$, que são os limites de estoque de cada produto em cada cliente. Ainda existe o parâmetro booleano *EntregaMáxima* que influencia o cálculo da variável $Entrega_{pct}$. Se ele for *1*, sempre que houver uma entrega para um cliente, ela deve encher seu estoque até o

máximo, se ele for *0*, essa restrição não existe. Assim, o modelo matemático do PPEP pode ser visto a seguir:

$$\text{Minimizar} \quad \begin{array}{c} PesoMaiorEntrega \cdot MaiorEntrega \\ + PesoMaiorNumeroEntregas \cdot MaiorNumeroEntregas \\ + \sum_{p} \sum_{c} \sum_{t} PesoNumeroEntregas \cdot SeEntrega_{pct} \end{array} \tag{37}$$

Sujeito a

$$Estoque_{pct} - Entrega_{pct} = EstoqueInicial_{pc} - Demanda_{pct}, \quad \forall p, c, t \mid t = 1 \tag{38}$$

$$Entrega_{pct} + Estoque_{pc,t-1} - Estoque_{pct} = Demanda_{pct}, \quad \forall p, c, t \mid t \geq 2 \tag{39}$$

$$Estoque_{pct} - EstoqueMaximo_{pc} \cdot EntregaMaxima \cdot SeEntregra_{pct} \geq 0, \quad \forall p, c, t \tag{40}$$

$$Estoque_{pct} \geq EstoqueMinimo_{pc}, \quad \forall p, c, t \tag{41}$$

$$Estoque_{pct} \leq EstoqueMaximo_{pc}, \quad \forall p, c, t \tag{42}$$

$$MaiorEntrega - \sum_{p} \sum_{c} Entrega_{pct} \geq 0, \quad \forall t \tag{43}$$

$$MaiorNumeroEntregas - \sum_{p} \sum_{c} SeEntrega_{pct} \geq 0, \quad \forall t \tag{44}$$

$$Entrega_{pct} - 10 \cdot EstoqueMaximo_{pc} \cdot SeEntregra_{pct} \leq 0, \quad \forall p, c, t \tag{45}$$

$$\sum_{p} \sum_{c} Entrega_{pct} \leq CapacidadeTransporte_{t}, \quad \forall t \tag{46}$$

$$Entrega_{pct} \geq 0, \quad \forall p, c, t \tag{47}$$

$$MaiorEntrega \geq 0 \tag{48}$$

$$Estoque_{pct} \geq 0, \quad \forall p, c, t \tag{49}$$

$$SeEntrega_{pct} \in \{0,1\}, \quad \forall p, c, t \tag{50}$$

$$MaiorNumeroEntregas \in \mathbb{N} \tag{51}$$

A função objetivo (37) trata de três objetivos que são ponderados por pesos: minimizar o tamanho da maior entrega, minimizar a maior quantidade de entregas feitas em um mesmo período e minimizar o número total de entregas realizadas. O principal objetivo deste modelo é reduzir o custo de transporte, que é representado pela parcela da função objetivo que minimiza o total de entregas. Porém, esse objetivo isoladamente pode fornecer soluções que

não são boas operacionalmente, como, por exemplo, uma solução com muitas entregas no mesmo período ou entregas muito grandes, destoando dos demais períodos. Assim, as outras duas parcelas da função objetivo visam distribuir melhor as entregas pelos períodos, melhorando a operacionalização das soluções geradas, uma vez que este é um modelo de programação das entregas e não contém todas as restrições operacionais.

A restrição (38) faz o balanço de estoque para o primeiro período, considerando a demanda, as entregas e o estoque inicial. A restrição (39) faz o balanço de estoque para os demais períodos, levando-se em consideração o estoque do período anterior. A restrição (40) garante, em conjunto com as restrições (38), (39) e (42), que a variável $Entrega_{pct}$ atinja um determinado valor que faça a variável $Estoque_{pct}$ ser igual ao parâmetro $EstoqueMáximo_{pc}$ se a variável $SeEntrega_{pct}$ e o parâmetro *EntregaMáxima* forem iguais a *1*. Esta restrição só é ativada quando o parâmetro *EntregaMáxima* for igual a *1*, logo, alterando o valor deste parâmetro o usuário altera o funcionamento do modelo matemático sem alterar o seu equacionamento. A restrição (41) garante que o nível de estoque de cada produto em cada cliente é sempre maior ou igual ao valor de estoque mínimo em todos os períodos. A restrição (42) garante que o nível de estoque de cada produto e para cada cliente é sempre menor ou igual ao valor de estoque máximo em todos os períodos. A restrição (43) calcula a maior quantidade entregue em um mesmo período. A restrição (44) calcula a maior quantidade de entregas feitas em um mesmo período entre todos os períodos. A restrição (45) garante que a variável $Entrega_{pct}$ só assume valor maior que zero se a variável $SeEntrega_{pct}$ for igual a *1* para cada produto, cliente e período. A restrição (46) garante que o total entregue em cada período não ultrapasse a capacidade de transporte. As restrições (47), (48) e (49) são as restrições de não negatividades das variáveis reais e as restrições (50) e (51) são as restrições de integralidade das variáveis inteiras.

5.3.1 IMPLEMENTAÇÃO DO MODELO PEPP

O primeiro passo para implementar esse modelo no GeMM é definir o projeto de modelagem PEPP e, em seguida, criar uma versão do projeto para a definição dos elementos de modelagem. Foram definidos no GeMM os três conjuntos do PPEP: *PRODUTOS*, *CLIENTES* e *PERÍODOS*, com os respectivos índices, *p*, *c* e *t*, como mostra a Figura 5.27. A Figura 5.28 apresenta os parâmetros necessários para a modelagem do PPEP. Todos os parâmetros são primários, representando os dados de entrada do modelo. Entre eles estão os pesos das variáveis na função objetivo, os dados de demanda dos clientes, a capacidade de

transporte por período, as informações do estoque dos clientes e o indicador booleano *EntregaMáxima*, utilizado para ativar e desativar a restrição (40).

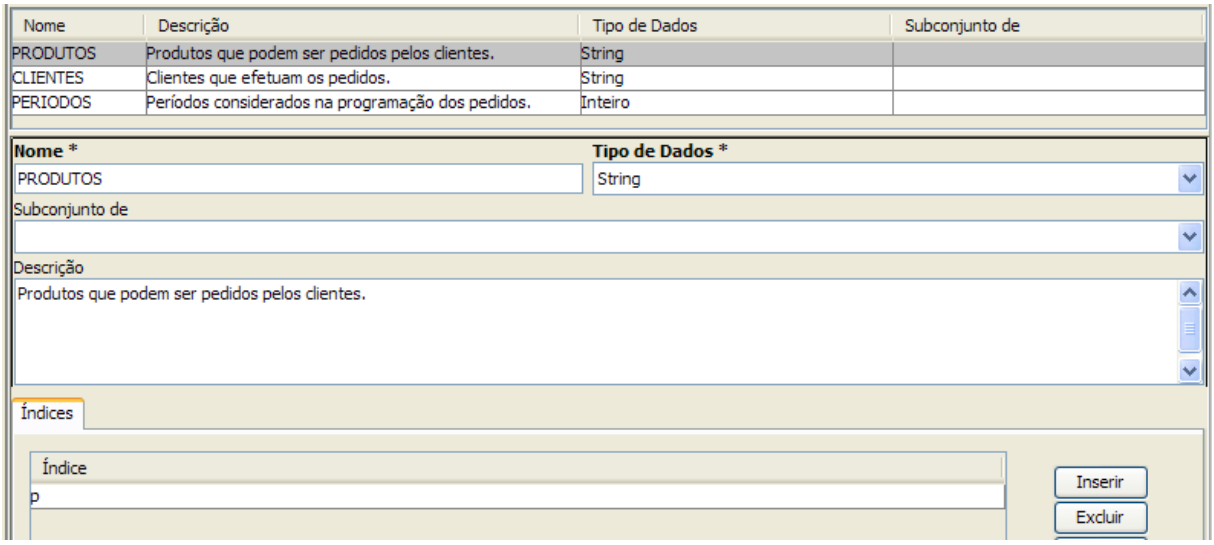


Figura 5.27: Conjuntos do modelo do PPEP

A Figura 5.29 mostra a tela do GeMM com as variáveis do modelo. Este modelo envolve os três tipos de variáveis: binária ($SeEntrega_{pct}$), inteira (*MaiorNumeroEntregas)* e real (*MaiorEntrega*, $Entrega_{pct}$ e $Estoque_{pct}$).

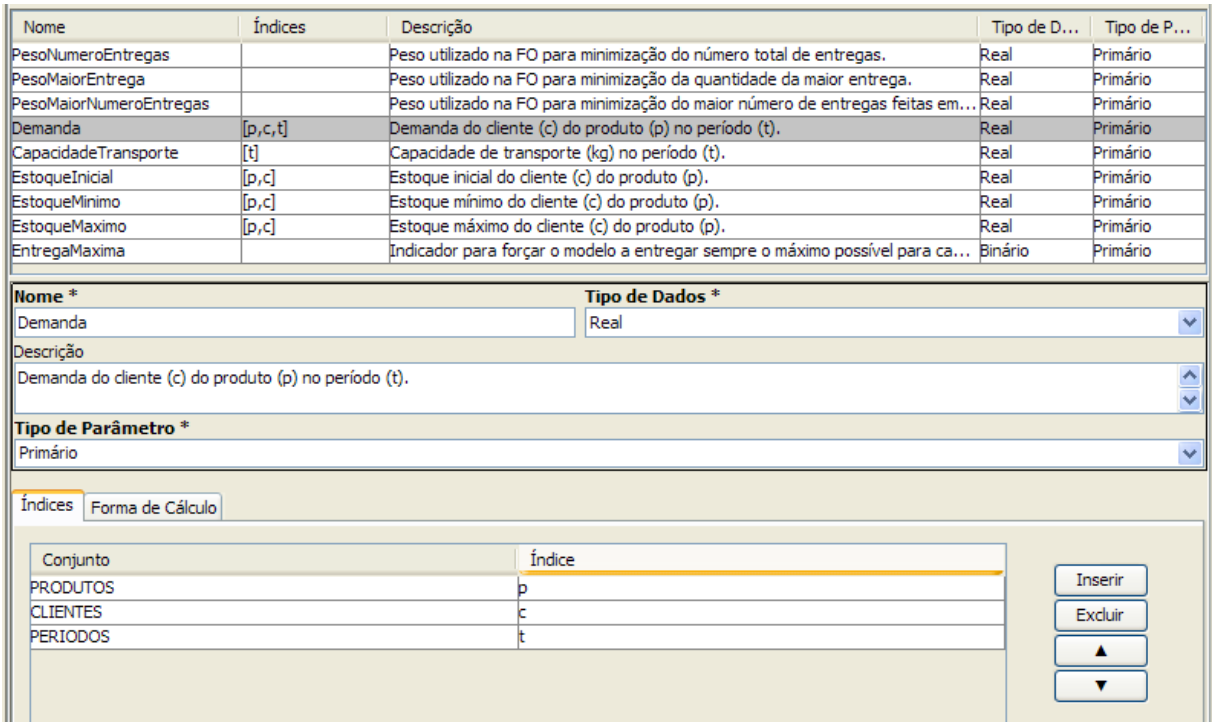


Figura 5.28: Parâmetros do modelo do PPEP

A Figura 5.30 apresenta a restrição (38) modelada no GeMM. Ela é indexada por *p*, *c* e *t*, e só deve existir para o primeiro período, logo possui a condição de geração $t == 1$, na

notação do GeMM. O LHS é dado pela expressão $Estoque_{pct}$ - $Entrega_{pct}$, cuja modelagem pode ser vista na Figura 5.30, e o RHS é dado pela expressão $EstoqueInicial_{pc}$ - $Demanda_{pct}$, mostrada na Figura 5.31. As outras restrições podem ser modeladas no GeMM de forma análoga, pois todas as funcionalidades necessárias já foram apresentadas.

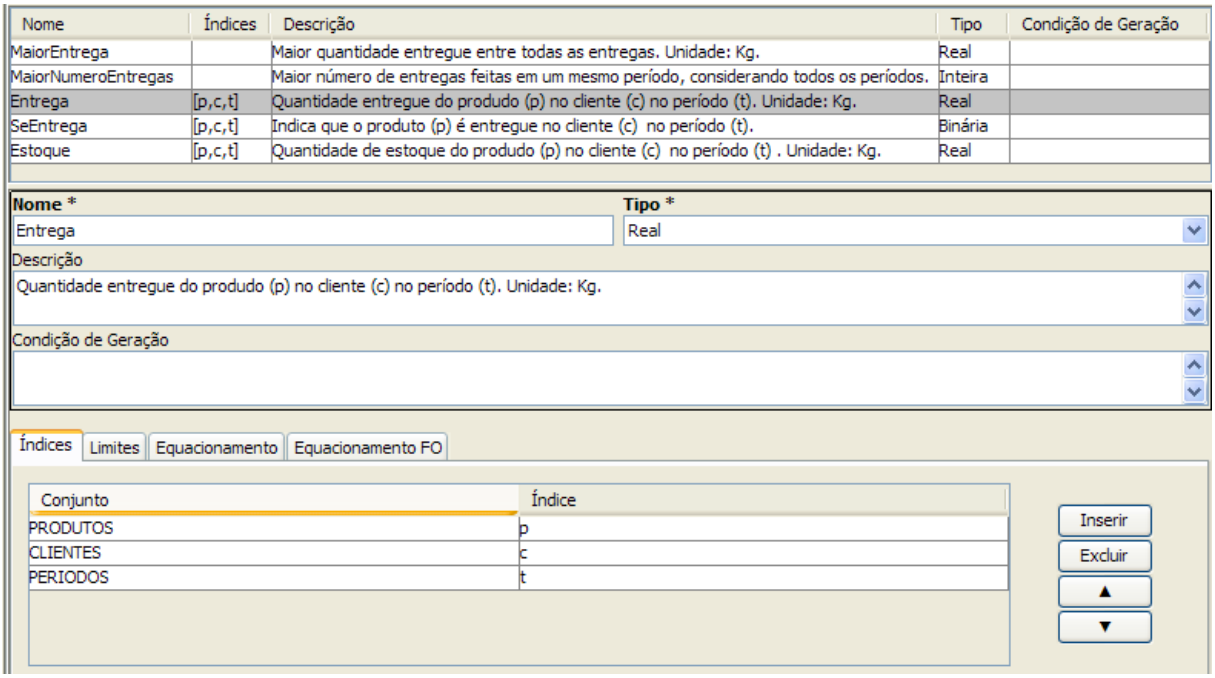


Figura 5.29: Variáveis do modelo do PPEP

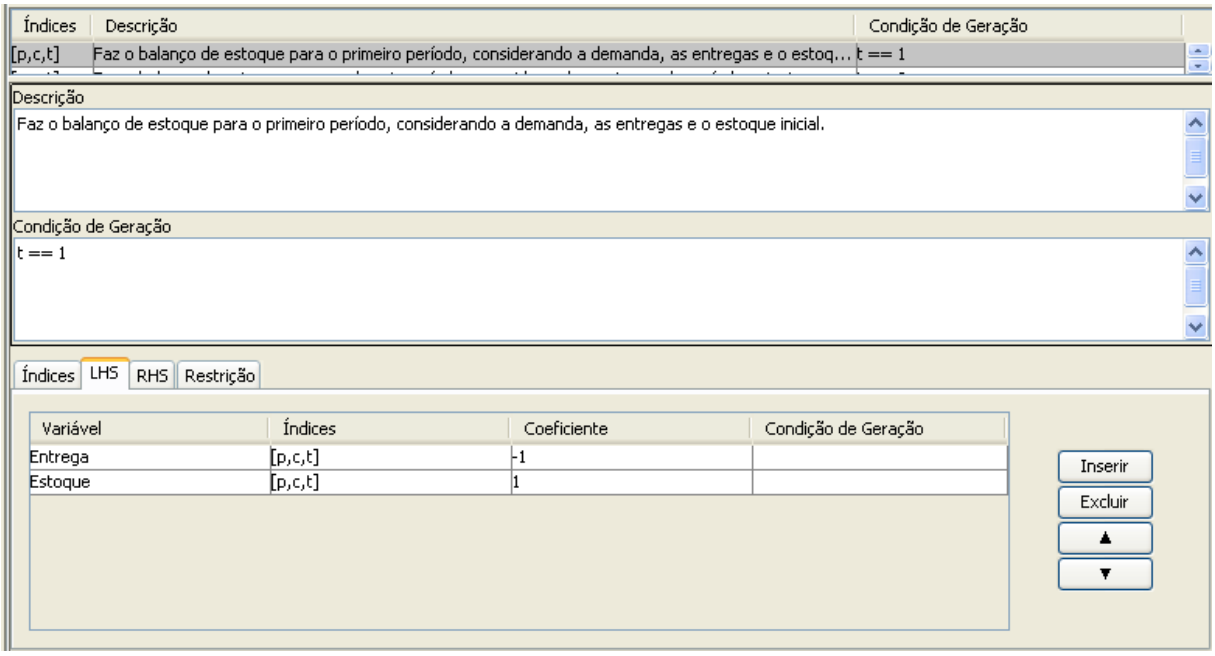


Figura 5.30: Restrição (38) *do modelo do PPEP*

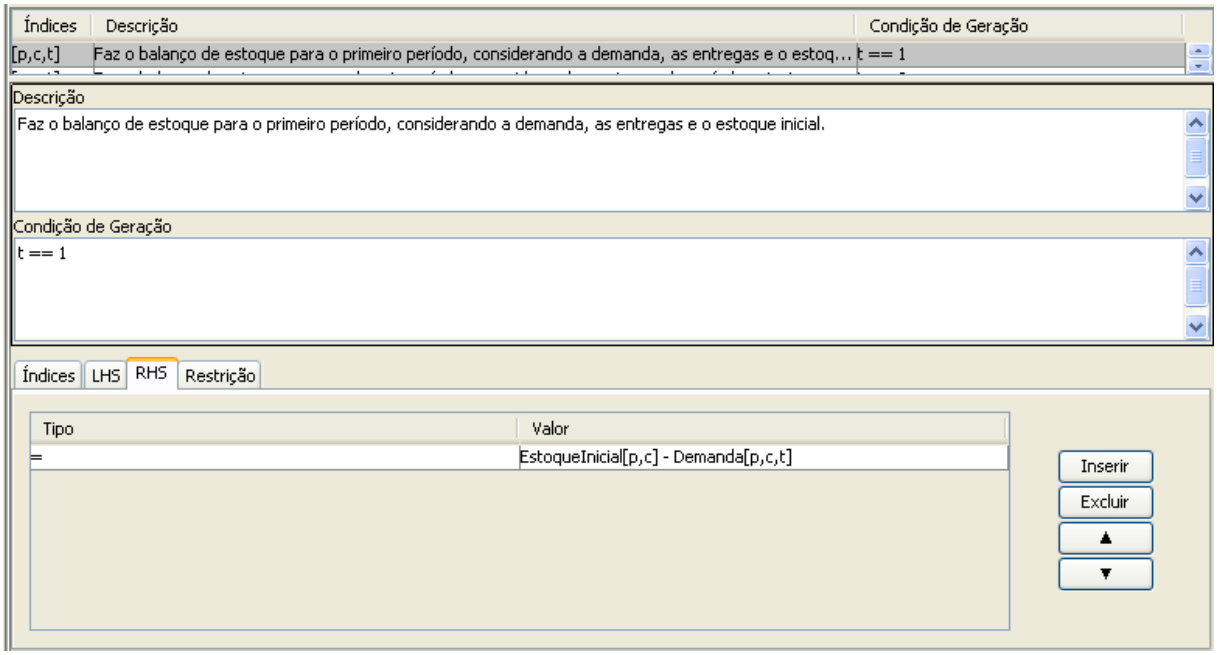


Figura 5.31: RHS da restrição (38) *do modelo do PPEP*

A Figura 5.32 mostra a modelagem da função objetivo. Ela é de minimização e os coeficientes das variáveis no modelo são justamente os parâmetros de peso para ponderar os três critérios usados na escolha da melhor solução: minimizar a maior quantidade entregue em um mesmo período, o maior número de entregas feitas em um mesmo período e o total de entregas feitas em todo o horizonte.

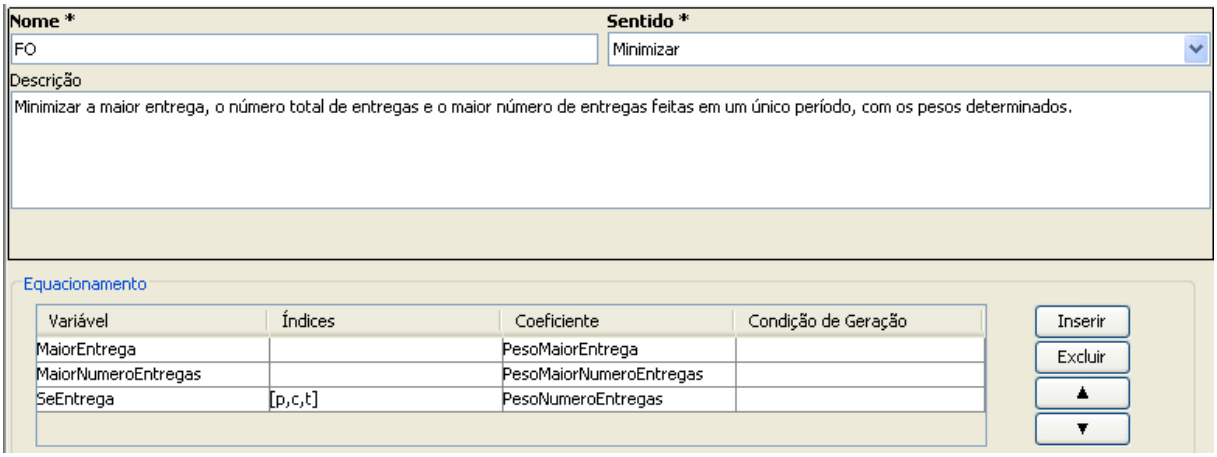


Figura 5.32: Função objetivo do modelo do PPEP

Uma vez implementado o modelo do PPEP no GeMM foi possível criar cenários de dados para que eles fossem otimizados através de um resolvedor.

5.3.2 CENÁRIO DE DADOS

Para validar o modelo implementado no GeMM foi criado um cenário de exemplo. Este cenário de dados para teste foi construído com três períodos, dois clientes e um produto. A capacidade de transporte utilizada é de 20.000 Kg nos três períodos e a demanda é de 1.000 Kg para todos os clientes nos três períodos. Os dados de estoque para os clientes podem ser vistos na Tabela 5.12.

Tabela 5.12: Dados de estoque para cenário de dados do PPEP

Cliente	Estoque Inicial (Kg)	Estoque Mínimo (Kg)	Estoque Máximo (Kg)
1	200	0	2.000
2	500	0	2.000

Para informar os dados apresentados no GeMM e otimizar este problema, é necessário criar uma versão de um cenário de dados como mostra a Figura 5.33. A partir do modelo matemático implementado, o GeMM gera as telas para a entrada das informações dos cenários de dados. A Figura 5.34 exibe a tela para edição dos parâmetros indexados por *PRODUTOS* e *CLIENTES*, que são os dados de estoque exibidos na Tabela 5.12.

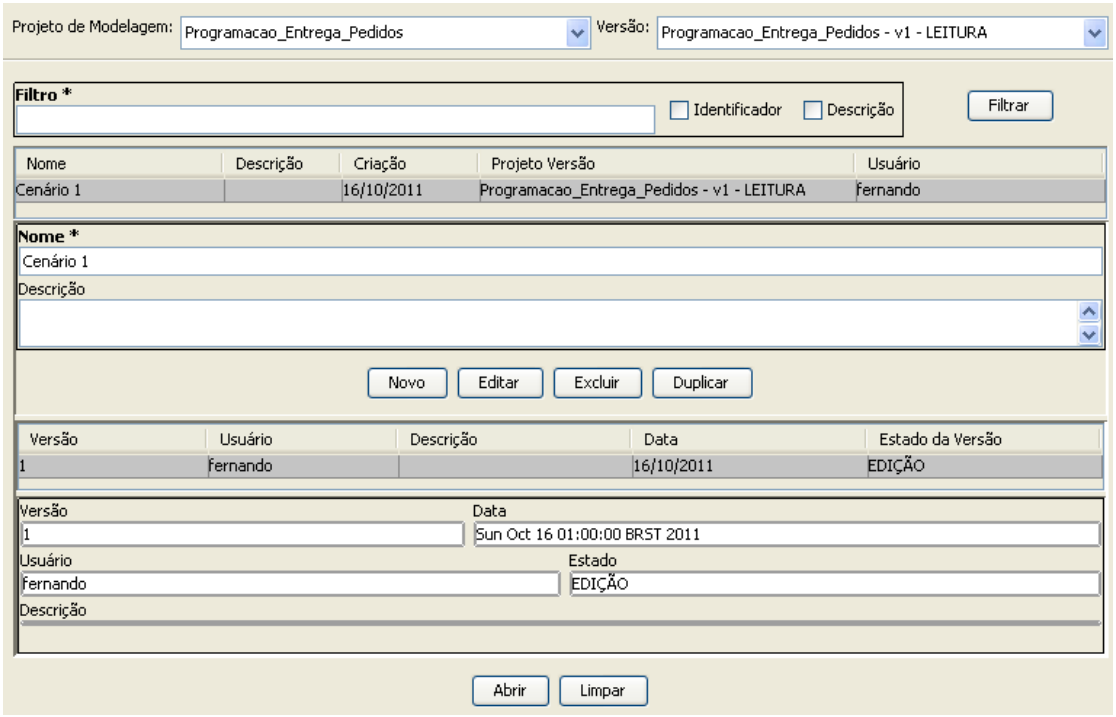


Figura 5.33: Controle dos cenários de dados do PPEP

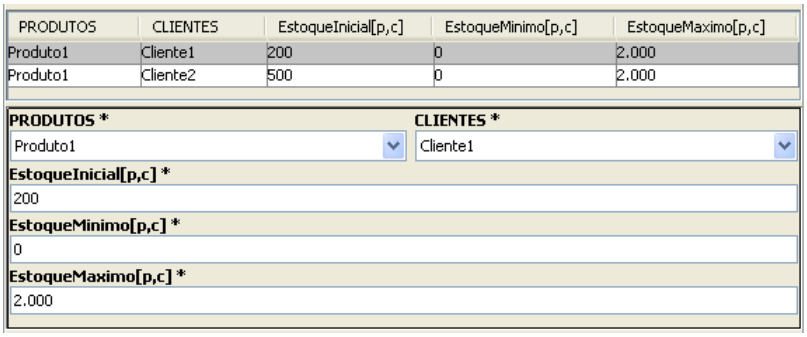


Figura 5.34: Parâmetros do cenário de dados do PPEP

A Figura 5.35 mostra a tela gerada pelo GeMM para edição dos valores dos parâmetros primários que não são indexados. Esses parâmetros são os pesos usados na função objetivo e o indicador *EntregaMáxima*.

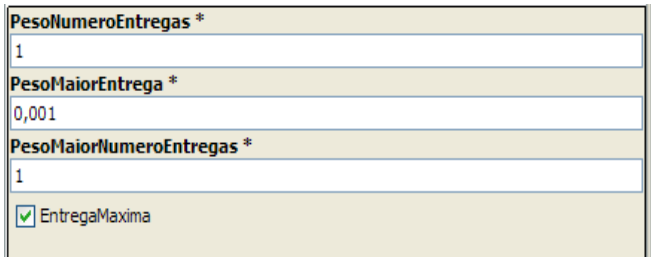


Figura 5.35: Parâmetros dos pesos da FO do cenário de dados do PPEP

A partir dos dados fornecidos, o GeMM gera uma instância de um PPLI e a otimiza utilizando um resolvedor. Na Tabela 5.13 está o resumo da otimização feita deste cenário de dados no GeMM. Seu resultado foi obtido com o CPLEX e foi validado com a implementação do mesmo modelo e dos mesmos dados no AIMMS. Tanto o valor da função objetivo, quanto os valores das variáveis foram iguais em ambos os ambientes de modelagem.

Tabela 5.13: Resumo da execução do cenário de teste do modelo PPEP

Total de variáveis geradas	20
Total de restrições geradas	39
Estado	Ótimo
Valor FO	9,3
Tempo total de otimização (segundos)	0,422
Tempo utilizado pelo GeMM (segundos)	0,391 (92,65%)

A Figura 5.36 mostra a tela com o resumo do resultado e o valor da função objetivo. Como mostra a Figura 5.35, o indicador *EntregaMáxima* é igual a *1*, assim, sempre que ocorre uma entrega, ela deve encher o estoque do cliente. Na Figura 5.37 estão os valores das variáveis, ou seja, a solução da programação de entrega dos pedidos. Para atender a demanda

diária de 1.000 Kg para os dois clientes é feita apenas uma entrega para cada um deles no primeiro período. Esta entrega, já descontada a demanda do período, deixa o nível de estoque no máximo, que é de 2.000 Kg. Nos outros períodos não há necessidade de fazer entregas, uma vez que o estoque consegue atender à demanda até o final do horizonte.

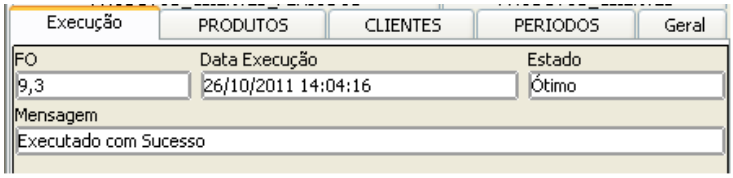


Figura 5.36: Resumo do resultado do cenário de dados do PPEP

PRODUTOS	CLIENTES	PERIODOS	SeEntrega[p,c,t]	Entrega[p,c,t]	Estoque[p,c,t]
Produto1	Cliente1	1	☑	2.800	2.000
Produto1	Cliente2	1	☑	2.500	2.000
Produto1	Cliente1	2	☐	0	1.000
Produto1	Cliente2	2	☐	0	1.000
Produto1	Cliente1	3	☐	0	0
Produto1	Cliente2	3	☐	0	0

Figura 5.37: Resultado das variáveis do cenário de dados do PPEP

Uma vez modelado este problema no GeMM, foi criado outro cenário de dados com informações reais desta subsidiária da Petrobras. Este cenário de dados teve por objetivo validar o GeMM na geração de grandes instâncias de PPLI. Nele são considerados 1.063 clientes, sete períodos (horizonte de uma semana) e um produto. A Tabela 5.14 mostra o resumo da instância do problema que foi gerada e otimizada pelo CPLEX versão 12.2.

Tabela 5.14: Resumo da execução do cenário do modelo PPEP

Total de variáveis geradas	22.325
Total de restrições geradas	37.226
Estado	Ótimo
Valor FO	808,31
Tempo total de otimização (segundos)	4702,967
Tempo utilizado pelo GeMM (segundos)	423,111 (9%)

A solução desta instância também foi validada com a implementação do modelo no AIMMS, que chegou ao mesmo resultado, tanto em termos do valor da função objetivo, como em termos dos valores das variáveis.

5.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou dois problemas de otimização, sendo que um deles com duas formulações matemáticas. O principal objetivo do GeMM é modelar PPLI's, assim os modelos matemáticos apresentados foram implementados a fim de avaliar a capacidade de modelagem do ambiente. Mostrou-se como o GeMM trata as diversas peculiaridades que podem existir na modelagem de problemas desse tipo. Para detalhar a implementação dos modelos matemáticos no GeMM foram utilizadas as próprias telas do ambiente.

Apresentou-se também cenários de dados para que o GeMM gerasse as instâncias dos PPLI's. Mostrou-se as telas geradas pelo GeMM a partir dos modelos matemáticos para tratar os respectivos cenários de dados. Os resultados obtidos foram comparados com os mesmos modelos e dados implementados no AIMMS, ambiente comercial para modelagem matemática.

Quanto ao tempo de execução gasto pelo GeMM para otimizar as instâncias apresentadas, pode-se observar nas Tabelas 5.4, 5.5 e 5.13 que para problemas menores o tempo de processamento do GeMM em relação ao tempo total é alto, pois a otimização dessas instâncias ocorre muito rápido e o tempo gasto pelo resolvedor é muito baixo. Para execução de instâncias maiores, como apresenta a Tabela 5.14, o tempo de processamento do GeMM é proporcionalmente baixo em relação ao tempo total, uma vez que a otimização desse problema é muito custosa computacionalmente. Desta forma, pode-se concluir que o tempo gasto pelo GeMM é proporcionalmente maior para instâncias pequenas dos problemas, mas para instâncias grandes, o tempo gasto pelo GeMM é muito baixo em relação ao tempo gasto pelo resolvedor. Em termos absolutos, o tempo gasto pelo GeMM foi aceitável para os problemas resolvidos.

CAPÍTULO 6 - CONCLUSÃO

6.1 CONTRIBUIÇÕES

Conforme definido no Capítulo 1, o objetivo principal dessa dissertação é desenvolver um novo ambiente de software para a modelagem de problemas de programação linear e inteira, chamado de GeMM, Gerenciador de Modelos Matemáticos. Ele tem o propósito de não apenas tratar a atividade de modelagem, mas também apoiar a gerência do ciclo de vida dos modelos matemáticos e dos dados associados.

O Capítulo 2 apresentou um estudo baseado em revisão sistemática da literatura, feito para identificar trabalhos relacionados com a modelagem de PPLI e uma pesquisa de opinião com profissionais e estudantes de Pesquisa Operacional. Esses estudos buscaram identificar quais os requisitos mais importantes devem ser atendidos por um ambiente de modelagem matemática, tanto para apoiar os desenvolvedores dos modelos, quanto para atender aos tomadores de decisão. Esses estudos orientaram a definição do escopo e da implementação do GeMM. Como destaca MAKOWSKI (2005), características como o controle de versão e a gerência de configuração dos modelos matemáticos são pouco exploradas e estudadas, apesar de bastante relevantes. A partir das pesquisas feitas também foi possível concluir que características como o tratamento dos dados separados do modelo, a utilização de bancos de dados, integração com outros softwares, o uso de interfaces gráficas para a entrada de dados e análise dos resultados pelos tomadores de decisão e comunicação eficiente com os resolvedores são críticas no desenvolvimento deste tipo de aplicação.

O Capítulo 3 mostrou os principais conceitos utilizados para a elaboração do GeMM, enquanto o Capítulo 4 detalhou as características de implementação da abordagem. As principais diferenças desta proposta, se comparada com as ferramentas de modelagem existentes, são a sua arquitetura na estrutura cliente-servidor, que permite o compartilhamento de informações e a interação entre os modeladores e os tomadores de decisão, e a sua capacidade de controlar a evolução dos modelos matemáticos e dos dados, através do controle de versão. Por ter esses dois aspectos, o GeMM passa a ter um diferencial em especial: ser multiusuário, no qual as pessoas podem trabalhar sobre diferentes versões acessando o ambiente pela rede, graças à arquitetura cliente-servidor, e não tendo que se preocupar com as alterações simultâneas, graças ao controle de versão. Outras ferramentas de modelagem não apoiam essas atividades, exigindo que seus usuários façam esse controle externamente.

A utilização de controle de versão traz conceitos da Engenharia de Software para a modelagem matemática. Permite o controle do ciclo de vida dos modelos e dos dados utilizados para obter os resultados, que podem ser usados no processo decisório de uma organização. Tal controle permite, inclusive, auditorias dos resultados obtidos, uma vez que estes estão associados a uma determinada versão de modelo e a uma determinada versão de cenário de dados. Para o tratamento dos cenários de dados, o GeMM gera automaticamente uma aplicação a partir de um modelo matemático para a entrada de dados e análise dos resultados. Na maior parte das aplicações práticas, alguns dados do problema não são conhecidos e, portanto, precisam ser estimados da melhor forma possível (BAZARAA, 1977). A análise de sensibilidade dos problema permite identificar o que ocorre com os resultados se um determinado parâmetro possuir outro valor ou se uma determinada restrição não existir. O tratamento adequando de cenários de dados e suas versões permite aos usuários do GeMM realizarem diversas análises sobre os modelos sem modificar sua definição.

Esta arquitetura também torna independentes as máquinas dos usuários daquelas que executam os problemas de otimização, onde estão instalados os resolvedores, além de separar as informações dos modelos matemáticos dos dados do problema. Esta separação dos dados, dos modelos e dos resolvedores é preconizada por RAMIREZ *et al.* (1993). A utilização dos conceitos da metamodelagem para armazenar em bancos de dados informações sobre os modelos matemáticos e sobre os dados associados também traz contribuições para o tratamento de forma geral da persistência dos PPLI's. Em abordagens como as de FOURER (1997) e DUTTA e FOURER (2008), a utilização de banco de dados relacional é feita para armazenar problemas específicos, com a construção de estruturas que atendam a um determinado problema ou a uma classe de problemas. Nestes casos, para cada modelo elaborado, um esquema de banco de dados deve ser criado. O GeMM utiliza o mesmo esquema de banco de dados para armazenar diferentes modelos e seus dados, que apesar de estarem na mesma estruturas, estão logicamente separados.

Uma contribuição importante deste trabalho é a proposta de uma estrutura, que a Figura 4.13 apresenta, para tratar os modelos dos problemas de programação linear e inteira. Esta estrutura permitiu uma representação geral dos PPLI's e foi a base do tratamento dos dados através da metamodelagem e da aplicação do versionamento na modelagem matemática. Através dessa estrutura de dados, o GeMM torna a modelagem dos problemas independente da interface com os usuários. Na implementação do ambiente utilizou-se formulários para que os usuários modelassem os problemas, porém esta interface pode ser

substituída sem que as funcionalidades propostas sejam perdidas, uma vez que a representação do problema é feita pela estrutura da Figura 4.13.

Outra característica do GeMM é a geração da documentação dos modelos matemáticos. A geração de imagens das expressões das funções objetivo e das restrições, passando por sua reescrita em LaTeX, se mostrou de grande utilidade para compreender e documentar os modelos. Em conjunto com os campos para descrição dos elementos de modelagem nos formulários do GeMM é possível gerar um documento em LaTeX e, consequentemente, um documento em PDF, que descreve completamente o modelo matemático implementado.

Conforme definido no Capítulo 1, o principal objetivo do GeMM é modelar problemas de programação linear e inteira. Assim, o Capítulo 5 apresentou dois problemas de otimização, sendo que um deles com duas formulações matemáticas, a fim de avaliar a capacidade de modelagem do ambiente. Os modelos apresentados foram escolhidos por possuírem características comuns a outros PPLI's e por terem peculiaridades que exigem diferentes recursos dos ambientes de modelagem. Apresentou-se também cenários de dados para que o GeMM gerasse as instâncias dos problemas e obtivesse os resultados através de um resolvedor, a fim de serem comparados com os resultados obtidos por meio da modelagem do mesmo problema em um ambiente de modelagem comercial. O Capítulo 5 mostrou que o GeMM é capaz de modelar problemas complexos e tratar seus diferentes cenários de dados.

6.2 LIMITAÇÕES

Uma limitação identificada é o controle de concorrência pessimista, através da utilização de bloqueio das versões que estão em edição. O controle de concorrência pessimista não permite edição em paralelo dos objetos controlados, obrigando que as alterações sejam feitas de forma sequencial. O GeMM não permite que diferentes usuários editem modelos matemáticos ou cenários de dados simultaneamente, assim, o projeto de modelagem ou um cenário de dados fica bloqueado para alteração por um único usuário. Também não permite a mesclagem de diferentes versões do mesmo modelo matemático para a criação de um novo.

Outra limitação do GeMM é não escrever as expressões das funções objetivo e das restrições em LaTeX da maneira mais concisa possível, exatamente como uma pessoa escreveria. Por exemplo, como o GeMM trata termo por termo dessas expressões, quando mais de um deles é somado sobre o mesmos índices e nas mesmas condições, o símbolo do somatório não precisaria ser repetido. Esta limitação pode ser percebida na comparação da

expressão da restrição (15) do modelo TTPPV, apresentada no Capítulo 5, com a imagem da Figura 5.9, referente a modelagem desta mesma restrição no GeMM.

O GeMM também necessita que o usuário escreva as restrições de forma que o LHS e o RHS estejam separados, no qual o LHS envolve apenas termos com variáveis e o RHS seja definido por uma expressão invariante. As linguagens algébricas, como, por exemplo, o AMPL e o GAMS, permitem que as expressões das restrições sejam escritas textualmente e cabe ao ambiente de modelagem interpretá-las.

6.3 TRABALHOS FUTUROS

Durante a elaboração deste trabalho, desde a definição do seu objetivo até a implementação do GeMM, surgiram diversas possibilidades de pesquisas e novos desenvolvimentos que não entraram no seu escopo, principalmente em função do tempo limitado. A seguir, apresentam-se alguns temas relacionados com essa dissertação que podem gerar novos trabalhos e eventuais desdobramentos.

O estudo sobre a modelagem de outros tipos de problemas de otimização pode estender o propósito do GeMM. O tratamento da modelagem de problemas não lineares, como problemas quadráticos, por exemplo, permitiria ao GeMM abranger uma gama maior de problemas encontrados em diversas áreas de aplicação de Pesquisa Operacional. A modelagem e as estruturas de dados para representação dos modelos de programação não linear são alguns assuntos a serem explorados nesta linha.

Outro trabalho futuro poderia ser a geração automática de cenários a partir da variação dos valores de um ou mais parâmetros, possibilitando analisar as possíveis alternativas do problema modelado. Por exemplo, no problema da dieta apresentado no Capítulo 4, chegou-se a uma solução ótima dado o preço dos alimentos, que é um dos parâmetros do modelo. A ideia deste trabalho futuro é, por exemplo, entender o comportamento deste modelo caso ocorra variação nos preços dos alimentos. O ambiente de modelagem pode apoiar essa atividade, gerando de forma otimizada diversas instâncias que diferem apenas nos valores de um ou mais parâmetros e fazendo a distribuição de carga entre os servidores de otimização.

A geração automática de testes também poderia ser explorada, no qual cada restrição ou conjunto de restrições possuiriam cenários de testes específicos, a fim de verificar a coerência do modelo. Outras funcionalidades que também poderiam melhorar o ambiente de modelagem são a análise da sintaxe, a verificação de defeitos e inviabilidades em tempo de

modelagem, semelhante ao que é feito por IDE's de desenvolvimento de software. Essas funções podem trazer ganhos na produtividade de construção de modelos matemáticos.

Diferentes formas de entrada de dados e visualização dos resultados também poderiam ser implementadas para auxiliar o processo de tomada de decisão. Ferramentas como tabelas dinâmicas, gráficos e relatórios automáticos trariam ao GeMM formas poderosas de análise e edição dos dados. Sua usabilidade poderia ser aprimorada com a utilização de gráficos de Gantt para problemas de programação e edição visual de grafos para problemas de fluxo em rede, por exemplo.

Trabalhos futuros envolvendo a gerência de configuração podem ser elaborados no intuito de controlar a razão das mudanças e apoiar a rastreabilidade dos dados. Por exemplo, para uma dada alteração no modelo ou nos dados de entrada, quais cenários de dados precisariam ser recalculados em função das mudanças afetarem o resultado ou não? Assim como permitiria a associação de diferentes modificações nos modelos e nos dados que possuem o mesmo motivo. A gestão de mudança poderia ser explorada neste contexto da modelagem matemática.

Pode-se ainda realizar estudos sobre a mesclagem de diferentes versões dos modelos matemáticos e seus cenários de dados. A mesclagem se aplica de várias formas: entre versões de modelos, entre versões de cenários da mesma versão do modelo e, principalmente, na migração de cenários de dados em função de evoluções dos modelos matemáticos.

REFERÊNCIAS

ADOBE SYSTEMS. **Portable Document Format (PDF)**. Disponível em: <http://www.adobe.com/br/products/acrobat/adobepdf.html>. Último acesso em 30 de dezembro de 2011.

BASSI, L. J. The diet problem revisited. **The American Economist**, v. 20, p. 35-39, 1976.

BAZARAA, M. **Linear programming and network flows**. New York: Wiley, 1977.

BERTAZZI, L.; PALETTA, G.; SPERANZA, M. G. Minimizing the total cost in an integrated vendor—Managed inventory system. **Journal of Heuristics**, v. 11, p. 393-419, 2005.

BISSCHOP, J.; MEERAUS, A. On the development of a general algebraic modeling system in a strategic planning environment. **Mathematical Programming Studies**, v. 20, p. 1-29, 1982.

BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. **UML guia do usuário**. Rio de Janeiro: Elsevier-Campus, 2006.

BROOK, A.; KENDRICK, D.; MEERAUS, ALEXANDER. GAMS, a user's guide. **ACM SIGNUM Newsletter**, v. 23, p. 10-11, 1988.

CHANGE VISION. **Astah Community**. Disponível em: <http://astah.net/>. Último acesso em 30 de dezembro de 2011.

CHARI, K.; SEN, T. K. An implementation of a graph-based modeling system for structured modeling (GBMS/SM). **Decision Support Systems**, v. 22, p. 103-120, 1998.

COLLAUD, G.; PASQUIER-BOLTUCK, J. gLPS: A graphical tool for the definition and manipulation of linear problems. **European Journal of Operational Research**, v. 72, p. 277-286, 1994.

CONRADI, R.; WESTFECHTEL, B. Version models for software configuration management. **ACM Computing Surveys**, v. 30, p. 232-282, 1998.

DNIT. **Distância entre cidades**. Disponível em: <http://www1.dnit.gov.br/rodovias/distancias/distancias.asp>. Último acesso em 30 de dezembro de 2011.

DUTTA, G.; FOURER, R. Database structure for a class of multi-period mathematical programming models. **Decision Support Systems**, v. 45, p. 870-883, 2008.

ECLIPSE FOUNDATION. **EclipseLink JPA**. Disponível em: <http://www.eclipse.org/eclipselink/jpa.php>. Último acesso em 30 de dezembro de 2011.

ESTUBLIER, J. Software configuration management: a roadmap. **Proceedings of 22nd International Conference on Software Engineering, The Future of Software Engineering**, Limerick, p. 279-289, 2000.

ESTUBLIER, J. Objects control for software configuration management. **Advanced Information Systems Engineering: 13th International Conference, CAiSE 2001**, Interlaken, p. 359-373, 2001.

FABFORCE.NET. **DBDesigner**. Disponível em: <http://www.fabforce.net/dbdesigner4/>. Último acesso em 30 de dezembro de 2011.

FORSTER, M.; MEVERT, P. A tool for network modeling. **European Journal of Operational Research**, v. 72, p. 287-299, 1994.

FOURER, R. Database structures for mathematical programming models. **Decision Support Systems**, v. 20, p. 317-344, 1997.

FOURER, R. Predictions for web technologies in optimization. **INFORMS Journal on Computing**, v. 10, p. 388-389, 1998.

FOURER, R. Software survey: Linear programming. **OR/MS TODAY**, v. 38, p. 60-69, 2011.

FOURER, R.; GAY, D. M.; KERNIGHAN, B. W. A modeling language for mathematical programming. **Management Science**, v. 36, p. 519-554, 1990.

FOURER, R.; MA, J.; MARTIN, K. Optimization Services: a framework for distributed optimization. **Operations Research**, v. 58, p. 1624-1636, 2010.

GEOFFRION, A. M. An introduction to structured modeling. **Management Science**, v. 33, p. 547-588, 1987.

GEOFFRION, A. M. Computer-based modeling environments. **European Journal of Operational Research**, v. 41, p. 33-43, 1989.

GREENBERG, H. J. A natural language discourse model to explain linear programming models and solutions. **Decision Support Systems**, v. 3, p. 333-342, 1987.

GREENBERG, H. J. The ANALYZE rulebase for supporting LP analysis. **Annals of Operations Research**, v. 65, p. 91-126, 1996.

GREENBERG, H. J.; MURPHY, F. H. Views of mathematical programming models and their instances. **Decision Support Systems**, v. 13, p. 3-34, 1995.

GUERRA, E. **SwingBean Framework**. Disponível em: <http://swingbean.sourceforge.net/>. Último acesso em 30 de dezembro de 2011.

GUROBI. **Gurobi Optimizer**. Disponível em: <http://www.gurobi.com/>. Último acesso em 30 de dezembro de 2011.

HAN THE THANH. **pdfTeX**. Disponível em: <http://www.tug.org/applications/pdftex/>. Último acesso em 30 de dezembro de 2011.

HAVERLY, C. A. OMNI model management system. **Annals of Operations Research**, v. 104, p. 127–140, 2001.

HSQLDB. **HSQLDB**. Disponível em: <http://hsqldb.org/>. Último acesso em 30 de dezembro de 2011.

HUH, S.; KIM, H. A real-time synchronization mechanism for collaborative model management. **Decision Support Systems**, v. 37, p. 315-330, 2004.

IBM CORPORATION. **IBM mathematical programming language extended 370 (MPSX / 370)**. Program Reference Manual, 1975.

IBM CORPORATION. **CPLEX**. Disponível em: <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>. Último acesso em 30 de dezembro de 2011.

JEP JAVA. **Jep Java - Math Expression Parser Open Source**. Disponível em: <http://sourceforge.net/projects/jep/>. Último acesso em 30 de dezembro de 2011.

JEUSFELD, M. A.; JOHNEN, U. A. An executable meta model for re-engineering of database schemas. **Proceedings of the 13th International Conference on the Entity-Relationship Approach**, Manchester, p. 533-547, 1994.

JLATEXMATH. **JLaTeXMath - A Java API to render LaTeX**. Disponível em: <http://forge.scilab.org/index.php/p/jlatexmath/>. Último acesso em 15 de janeiro de 2012.

KITCHENHAM, B. **Procedures for performing systematic reviews**. Technical Report, Keele University, 2004.

KWON, O. B.; PARK, S. J. RMT: A modeling support system for model reuse. **Decision Support Systems**, v. 16, p. 131-153, 1996.

LAMPORT, L. **LaTeX**. Disponível em: <http://www.latex-project.org/>. Último acesso em 30 de dezembro de 2011.

LEE, J. S. A model base for identifying mathematical programming structures. **Decision Support Systems**, v. 7, p. 99-105, 1991.

LENARD, M. An object-oriented approach to model management. **Decision Support Systems**, v. 9, p. 67-73, 1993.

MA, J. Type and inheritance theory for model management. **Decision Support Systems**, v. 19, p. 53-60, 1997.

MAKHORIN, A. **GLPK - GNU Linear Programming Kit**. Disponível em: <http://www.gnu.org/software/glpk/>. Último acesso em 15 de janeiro de 2012.

MAKOWSKI, M. A structured modeling technology. **European Journal of Operational Research**, v. 166, p. 615-648, 2005.

MAROS, I.; KHALIQ, M. H. Advances in design and implementation of optimization software. **European Journal of Operational Research**, v. 140, p. 322-337, 2002.

MATURANA, S. Issues in the design of modeling languages for mathematical programming. **European Journal of Operational Research**, v. 72, p. 243-261, 1994.

MATURANA, S.; FERRER, J. C.; BARAÑAO, F. Design and implementation of an optimization-based decision support system generator. **European Journal of Operational Research**, v. 154, p. 170-183, 2004.

MCALOON, K.; TRETKOFF, C. Logic, modeling, and programming. **Annals of Operations Research**, v. 71, p. 335-372, 1997.

MELO, R. A.; URRUTIA, S.; RIBEIRO, C. C. The traveling tournament problem with predefined venues. **Journal of Scheduling**, v. 12, p. 607-622, 2009.

MENS, T. A state-of-the-art survey on software merging. **IEEE Transactions on Software Engineering**, v. 28, p. 449-462, 2002.

MUHANNA, W. An object-oriented framework for model management and DSS development. **Decision Support Systems**, v. 9, p. 217-229, 1993.

MURPHY, F. H.; STOHR, E. A. An intelligent system for formulating linear programs. **Decision Support Systems**, v. 2, p. 39-47, 1986.

MURPHY, F. H.; STOHR, E. A.; ASTHANA, A. Representation schemes for linear programming models. **Management Science**, v. 38, p. 964-991, 1992.

ORACLE. **Oracle Database**. Disponível em: <http://www.oracle.com/br/products/database/index.html>. Último acesso em 30 de dezembro de 2011a.

ORACLE. **NetBeans IDE**. Disponível em: <http://netbeans.org/>. Último acesso em 30 de dezembro de 2011b.

PARAGON DECISION TECHNOLOGY. **AIMMS**. Disponível em: <http://www.aimms.com/>. Último acesso em 30 de dezembro de 2011.

PILLUTLA, S.; NAG, B. N. Object-oriented model construction in production scheduling decisions. **Decision Support Systems**, v. 18, p. 357-375, 1996.

PRESSMAN, R. **Engenharia de software**. São Paulo: McGraw-Hill, 2006.

RAMIREZ, R.; CHING, C.; ST. LOUIS, R. D. Independence and mappings in model-based decision support systems. **Decision Support Systems**, v. 10, p. 341-358, 1993.

SCOPUS. **Scopus – Document search**. Disponível em: <http://www.scopus.com/>. Último acesso em 30 de dezembro de 2011.

SINGULAR SYSTEMS. **Jep Java - Math Expression Parser**. Disponível em: <http://www.singularsys.com/jep/>. Último acesso em 30 de dezembro de 2011.

SRINIVASAN, A.; SUNDARAM, D. An object relational approach for the design of decision support systems. **European Journal of Operational Research**, v. 127, p. 594-610, 2000.

THE APACHE SOFTWARE FOUNDATION. **Subversion**. Disponível em: <http://subversion.apache.org/>. Último acesso em 30 de dezembro de 2011.

APÊNDICE A - PROTOCOLO DA REVISÃO SISTEMÁTICA DA LITERATURA

A.1 QUESTÃO DA PESQUISA

A questão principal que deve ser respondida por este estudo baseado em revisão sistemática da literatura é: quais os requisitos mais importantes que devem ser atendidos em um ambiente de modelagem matemática, tanto para os desenvolvedores dos modelos quanto para os seus usuários, os tomadores de decisão?

A.1.1 POPULAÇÃO

A população que está envolvida nessa pesquisa é composta pelos projetos de desenvolvimento de ambientes de modelagem matemática, pelas linguagens de modelagem e pelos estudos e pesquisas que envolvam modelagem matemática.

A.1.2 INTERVENÇÃO

Elementos da Engenharia de Software utilizados em sistemas e ambientes para modelagem matemática de problemas de programação linear e problemas de programação inteira.

A.1.3 RESULTADO

Características e requisitos dos ambientes de modelagem para problemas de programação linear e problemas de programação inteira.

A.2 ESTRATÉGIA PARA PESQUISA

Para realizar a pesquisa dos trabalhos, o primeiro passo foi identificar quais termos mais apropriados deveriam estar na consulta para que fossem encontrados os resultados esperados. De acordo com a estrutura da questão deste estudo, os termos foram divididos em três tópicos, que estão relacionados justamente com a população, a intervenção e o resultado desta pesquisa.

Desta forma, a consulta para busca nas bases de dados foi formada usando uma conjunção desses termos, ou seja, a consulta pode ser representada da seguinte forma: **POPULAÇÃO** *E* **INTERVENÇÃO** *E* **RESULTADO**.

A.2.1 TERMOS UTILIZADOS

Como a língua mais utilizada nas bases de dados eletrônicas é o inglês, os termos pesquisados estão todos nesta língua. Para a montagem do protocolo foram levantados os principais termos que devem ser combinados na *query* de busca. Esses termos devem ser

pesquisados principalmente nos títulos, nos resumos e nas palavras-chave dos trabalhos publicados. Os termos foram divididos nas seguintes categorias:

- Otimização (POPULAÇÃO):

  operational research, operations research, decision making, decision model, decision models, decision support, management science, optimisation, optimization, integer programming, linear programming, mathematical programming.

- Sistemas de informação e banco de dados (INTERVENÇÃO):

  software engineering, computer-assisted analysis, computer-based environment, computer-aided software engineering, large-scale systems, database, database-oriented languages, object-oriented framework, object-oriented systems.

- Sistemas de modelagem (RESULTADO):

  enterprise model management systems, graphical modeling, graphical modelling, integrated modeling environments, integrated modelling environments, matrix generators, matrix generator, matrix generation, model generation, model generator, model generators, model management, model-data separation, modeling environment, modeling language, modeling life cycle, modeling system, modeling systems, modelling environment, modelling language, modelling life cycle, modelling system, modelling systems, object oriented modeling, object oriented modelling, relational modeling, relational modelling, structured modeling, structured modelling.

A.2.2 CONSULTA

Uma vez levantados os termos relevantes, eles foram combinados para formar uma consulta para a realização da pesquisa. Seguindo o formato definido para o estudo baseado em revisão sistemática da literatura para a combinação dos termos, **POPULAÇÃO** *E* **INTERVENÇÃO** *E* **RESULTADO**, a seguinte consulta foi montada:

```
("operational research" OR "operations research" OR "decision making" OR "decision model" OR "decision models" OR "decision support" OR "management science" OR "optimisation" OR "optimization" OR "integer programming" OR "linear programming" OR "mathematical programming")

AND

("software engineering" OR "computer-assisted analysis" OR "computer-based environment" OR "computer-aided software engineering" OR "large-scale systems" OR "database" OR "database-oriented languages" OR "object-oriented framework" OR "object-oriented systems")

AND
```

("enterprise model management systems" OR "graphical modeling" OR "graphical modelling" OR "integrated modeling environments" OR "integrated modelling environments" OR "matrix generators" OR "matrix generator" OR "matrix generation" OR "model generation" OR "model generator" OR "model generators" OR "model management" OR "model-data separation" OR "modeling environment" OR "modeling language" OR "modeling life cycle" OR "modeling system" OR "modeling systems" OR "modelling environment" OR "modelling language" OR "modelling life cycle" OR "modelling system" OR "modelling systems" OR "object oriented modeling" OR "object oriented modelling" OR "relational modeling" OR "relational modelling" OR "structured modeling" OR "structured modelling")

A.2.3 FONTES

Para operacionalizar a pesquisa deste estudo baseado em revisão sistemática da literatura, buscou-se utilizar uma base de dados eletrônica capaz de pesquisar em diversas fontes simultaneamente.

Como cada base de dados eletrônica apresenta um formato específico para a execução da consulta, foi selecionada a base que apresentou as melhores fontes de consulta e a forma mais flexível para montagem da pesquisa. Assim, foi escolhida a base de dados eletrônica do SCOPUS (2011) para execução da pesquisa.

Nesta base, foram identificadas as principais fontes e publicações que poderiam trazer o melhor retorno para esta pesquisa. Assim, foram selecionadas as seguintes fontes para pesquisa dentro do SCOPUS:

- Decision Support Systems
- European Journal of Operational Research
- Annals of Operations Research
- Lecture Notes in Computer Science Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics
- Environmental Modelling and Software
- Journal of the Operational Research Society
- Informs Journal on Computing
- Computer Science in Economics and Management
- Journal of Management Information Systems

A.2.4 *QUERY* DE BUSCA

Uma vez levantados os termos para a consulta, o escopo, a base de dados e o respectivo formato para montagem da pesquisa, neste caso do SCOPUS, a *query* de busca ficou da seguinte forma:

(TITLE-ABS-KEY("operational research") OR TITLE-ABS-KEY("operations research") OR TITLE-ABS-KEY("decision making") OR TITLE-ABS-KEY("decision

model") OR TITLE-ABS-KEY("decision models") OR TITLE-ABS-KEY("decision support") OR TITLE-ABS-KEY("management science") OR TITLE-ABS-KEY("optimisation") OR TITLE-ABS-KEY("optimization") OR TITLE-ABS-KEY("integer programming") OR TITLE-ABS-KEY("linear programming") OR TITLE-ABS-KEY("mathematical programming"))

AND

(TITLE-ABS-KEY("software engineering") OR TITLE-ABS-KEY("computer-assisted analysis") OR TITLE-ABS-KEY("computer-based environment") OR TITLE-ABS-KEY("computer-aided software engineering") OR TITLE-ABS-KEY("large-scale systems") OR TITLE-ABS-KEY("database") OR TITLE-ABS-KEY("database-oriented languages") OR TITLE-ABS-KEY("object-oriented framework") OR TITLE-ABS-KEY("object-oriented systems"))

AND

(TITLE-ABS-KEY("enterprise model management systems") OR TITLE-ABS-KEY("graphical modeling") OR TITLE-ABS-KEY("graphical modelling") OR TITLE-ABS-KEY("integrated modeling environments") OR TITLE-ABS-KEY("integrated modelling environments") OR TITLE-ABS-KEY("matrix generators") OR TITLE-ABS-KEY("matrix generator") OR TITLE-ABS-KEY("matrix generation") OR TITLE-ABS-KEY("model generation") OR TITLE-ABS-KEY("model generator") OR TITLE-ABS-KEY("model generators") OR TITLE-ABS-KEY("model management") OR TITLE-ABS-KEY("model-data separation") OR TITLE-ABS-KEY("modeling environment") OR TITLE-ABS-KEY("modeling language") OR TITLE-ABS-KEY("modeling life cycle") OR TITLE-ABS-KEY("modeling system") OR TITLE-ABS-KEY("modeling systems") OR TITLE-ABS-KEY("modelling environment") OR TITLE-ABS-KEY("modelling language") OR TITLE-ABS-KEY("modelling life cycle") OR TITLE-ABS-KEY("modelling system") OR TITLE-ABS-KEY("modelling systems") OR TITLE-ABS-KEY("object oriented modeling") OR TITLE-ABS-KEY("object oriented modelling") OR TITLE-ABS-KEY("relational modeling") OR TITLE-ABS-KEY("relational modelling") OR TITLE-ABS-KEY("structured modeling") OR TITLE-ABS-KEY("structured modelling"))

AND

(LIMIT-TO(EXACTSRCTITLE, "Decision Support Systems") OR LIMIT-TO(EXACTSRCTITLE, "European Journal of Operational Research") OR LIMIT-TO(EXACTSRCTITLE, "Annals of Operations Research") OR LIMIT-TO(EXACTSRCTITLE, "Lecture Notes in Computer Science Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics") OR LIMIT-TO(EXACTSRCTITLE, "Environmental Modelling and Software") OR LIMIT-TO(EXACTSRCTITLE, "Journal of the Operational Research Society") OR LIMIT-TO(EXACTSRCTITLE, "Informs Journal on Computing") OR LIMIT-TO(EXACTSRCTITLE, "Computer Science in Economics and Management") OR LIMIT-TO(EXACTSRCTITLE, "Journal of Management Information Systems"))

A.3 CRITÉRIOS DE SELEÇÃO DE ESTUDOS

Os critérios de seleção de estudos são os critérios que definem quando os estudos devem ou não ser incluídos na análise para a obtenção dos resultados. Existem critérios de inclusão e de exclusão de estudos.

A.3.1 CRITÉRIOS PARA A INCLUSÃO DE ESTUDOS

São incluídos os estudos sobre sistemas de informação, ambientes e linguagens de modelagem para a modelagem matemática de problemas de programação linear e

programação inteira. Também são incluídos trabalhos sobre a modelagem geral de problemas de apoio à decisão e de Pesquisa Operacional.

A.3.2 CRITÉRIOS PARA EXCLUSÃO DE ESTUDOS

Na *query* de busca não é possível filtrar exatamente todos os trabalhos de interesse de forma automática. Assim, foram elaborados alguns critérios de exclusão para que trabalhos, que não estivessem relacionados exatamente com o tema da pesquisa, pudessem ser excluídos de forma sistemática.

Seguem os critérios elaborados:

- **CE01**: Uso de modelagem matemática para a resolução de um problema específico
- **CE02**: Análise do processo de tomada de decisão, sem uso de ambiente de modelagem
- **CE03**: Modelagem e resolução de problemas usando técnicas de simulação
- **CE04**: Análise e modelagem de problemas estocásticos e não determinísticos
- **CE05**: Modelagem de software que não seja de Pesquisa Operacional
- **CE06**: Análise e modelagem de problemas dinâmicos
- **CE07**: Análise e modelagem de problemas não lineares
- **CE08**: Problemas específicos de software, como ferramentas de uso específico ou metodologias sem relação com problemas de Pesquisa Operacional

A.3.3 ESTRATÉGIA PARA EXCLUSÃO DE ESTUDOS

Os trabalhos retornados pela pesquisa foram analisados inicialmente pelo título, depois pelo resumo e então foram lidos de forma completa. Nenhum trabalho foi excluído apenas pelo título, pois pelo menos os resumos também foram lidos. Se na leitura do resumo ficou evidente a aplicação de algum dos critérios de exclusão, o artigo foi então excluído do estudo. Caso o trabalho tenha se mostrado aderente ao assunto ou não tenha ficado clara a aplicação dos critérios de exclusão apenas com a leitura do resumo, então o artigo foi lido por completo. Se o artigo se mostrou pertinente, os resultados foram extraídos, senão, o trabalho foi excluído após a leitura completa segundo algum critério definido anteriormente.

A.4 EXECUÇÃO DA REVISÃO

A *query* de busca foi executada na base de dados eletrônica do SCOPUS em maio de 2010 e 62 trabalhos completos foram encontrados. O primeiro passo foi analisar os títulos e fazer a leitura de todos os resumos. Com a leitura dos resumos, foram excluídos 13 trabalhos por estarem de acordo com algum critério de exclusão. Esses trabalhos podem ser vistos na Tabela A.1.

Tabela A.1: Trabalhos excluídos pela leitura do resumo

Título	Autores	Publicação	Ano	Critério de Exclusão
Modelling and simulation of ecological propagation processes: Application to fire spread	Muzy, A., Innocenti, E., Santucci, J. F., Santoni, P. A., Hill, D. R. C.	Environmental Modelling and Software	2005	CE03
SISCO: An object-oriented supply chain simulation system	Chatfield, D. C., Harrison, T. P., Hayya, J. C.	Decision Support Systems	2006	CE03
Monitoring patterns through an integrated management and mining tool	Kotsifakos, E. E., Ntoutsi, I., Vrahoritis, Y., Theodoridis, Y.	Lecture Notes in Computer Science	2008	CE08
Reflections for the future of system dynamics	Richardson, G. P.	Journal of the Operational Research Society	1999	CE06
NED-2: An agent-based decision support system for forest ecosystem management	Nute, D., Potter, W. D., Maier, F., Wang, J., Twery, M., Rauscher, H. M., Knopp, P., Thomasma, S., Dass, M., Uchiyama, H., Glende, A.	Environmental Modelling and Software	2004	CE03
Keynote address II: Domain-specific modeling: No one size fits all	Tolvanen, J. P.	Lecture Notes in Computer Science	2005	CE05
Privacy in Statistical Databases: UNESCO Chair in Data Privacy International Conference, PSD 2008, Proceedings		Lecture Notes in Computer Science	2008	CE01
Model transformation using graph transactions	Ribeiro, L., Foss, L., Da Silva, B., Nunes, D.	Lecture Notes in Computer Science	2009	CE05
Efficient reasoning about finite satisfiability of UML class diagrams with constrained generalization sets	Maraee, A., Balaban, M.	Lecture Notes in Computer Science	2007	CE05
Empirical evaluation in software engineering: Role, strategy, and limitations	Briand, L.C.	Lecture Notes in Computer Science	2007	CE05
The assessment of emission-source contributions to air quality by using a coupled MM5-ARPS-CMAQ modeling system: A case study in the Beijing metropolitan region, China	Cheng, S., Chen, D., Li, J., Wang, H., Guo, X.	Environmental Modelling and Software	2007	CE03
Deterministic mathematical modelling for the spatial allocation of multi-categorical resources. With an application to real health data	Segall, Richard S.	Journal of the Operational Research Society	1992	CE01;CE07
SYMMS: A model management system that supports model reuse, sharing, and integration	Muhanna, W. A.	European Journal of Operational Research	1994	CE02

Dos trabalhos pesquisados, 22 foram excluídos durante a leitura completa, devido a algum critério de exclusão. Esses trabalhos podem ser vistos na Tabela A.2.

Tabela A.2: Trabalhos excluídos após a leitura

Título	Autores	Publicação	Ano	Critério de Exclusão
Knowledge sharing and negotiation support in multiperson decision support systems	Jarke, M.	Decision Support Systems	1986	CE02
Model composition in a distributed environment	Chari, K.	Decision Support Systems	2003	CE02
A model management solution system for multicommodity network flows	Lari, A. R., Nag, B. N.	European Journal of Operational Research	1993	CE02
Using a mathematical programming modeling language for optimal CTA	Castro, J., Baena, D.	Lecture Notes in Computer Science	2008	CE01
An entity-relationship approach to model management	Blanning, R.W.	Decision Support Systems	1986	CE02
Model integration and a theory of models	Dolk, D. R., Kottemann, J. E.	Decision Support Systems	1993	CE02
Model management issues and directions	Chang, A. M., Holsapple, C. W., Whinston, A. B.	Decision Support Systems	1993	CE02
Meta-synthesis approach to complex system modeling	Gu, J., Tang, X.	European Journal of Operational Research	2005	CE05
Model management systems. An overview	Blanning, R. W.	Decision Support Systems	1993	CE02
The Conical Methodology and the evolution of simulation model development	Nance, R. E.	Annals of Operations Research	1994	CE03
Relationships between the decision support system subspecialties and reference disciplines: An empirical investigation	Eom, S. B.	European Journal of Operational Research	1998	CE02
Integrated model management in the data warehouse era	Dolk, D. R.	European Journal of Operational Research	2000	CE08
Mapping the intellectual structure of research in decision support systems through author cocitation analysis (1971-1993)	Eom, S. B.	Decision Support Systems	1996	
Modelling and analysis of multistage stochastic programming problems: A software environment	Messina, E., Mitra, G.	European Journal of Operational Research	1997	CE04
Supporting decision support: Where information on DSS is located	Abraham, T., Wankel, C.	Decision Support Systems	1995	CE02
The Theory of SML Schema-Directed Query	Tsai, Y. C.	Annals of Operations Research	2001	CE08

Título	Autores	Publicação	Ano	Critério de Exclusão
Database schema transformation optimisation techniques for the AutoMed system	Tong, N.	Lecture Notes in Computer Science	2003	CE08
An active modeling system for econometric analysis	Dolk, D. R., Kridel, D. J.	Decision Support Systems	1991	CE02
Model management system for IRT-based test construction decision support system	Wu, I. L.	Decision Support Systems	2000	CE02
Object-oriented modelling of military communications networks	Bailey, Michael, Kemple, William, West, Michael, Chase, Charles	Journal of the Operational Research Society	1994	CE01
Subscript-free modeling languages: A tool for facilitating the formulation and use of models	Lin, S. Y. E., Schuff, D., St. Louis, R. D.	European Journal of Operational Research	2000	CE08
MetaModelica: A unified equation-based semantical and mathematical modeling language	Pop, A., Fritzson, P.	Lecture Notes in Computer Science	2006	CE05

Dos 62 trabalhos pesquisados, 27 foram lidos e analisados em detalhe. A partir desses, que podem ser vistos na Tabela A.3 em ordem cronológica, foram extraídos os resultados esperados neste estudo baseado em revisão sistemática.

Tabela A.3: Trabalhos incluídos na revisão da literatura

Título	Autores	Publicação	Ano
An intelligent system for formulating linear programs	Murphy, F. H., Stohr, E. A.	Decision Support Systems	1986
A natural language discourse model to explain linear programming models and solutions	Greenberg, H. J.	Decision Support Systems	1987
A support system for optimization modelling	Singh, I. S., Sadagopan, S.	Decision Support Systems	1987
Computer-based modeling environments	Geoffrion, A. M.	European Journal of Operational Research	1989
A model base for identifying mathematical programming structures	Lee, J. S.	Decision Support Systems	1991
An object-oriented approach to model management	Lenard, M. L.	Decision Support Systems	1993
Independence and mappings in model-based decision support systems	Ramirez, R. G., Ching, C., St. Louis, R. D.	Decision Support Systems	1993
An object-oriented framework for model management and DSS development	Muhanna, W. A.	Decision Support Systems	1993
Towards a logical reconstruction of structured modeling	Chari, S., Krishnan, R.	Decision Support Systems	1993

Título	Autores	Publicação	Ano
Issues in the design of modeling languages for mathematical programming	Maturana, S. V.	European Journal of Operational Research	1994
gLPS: A graphical tool for the definition and manipulation of linear problems	Collaud, G., Pasquier-Boltuck, J.	European Journal of Operational Research	1994
A tool for network modeling	Forster, M., Mevert, P.	European Journal of Operational Research	1994
Views of mathematical programming models and their instances	Greenberg, H. J., Murphy, F. H.	Decision Support Systems	1995
An object-oriented framework for model management	Ma, J.	Decision Support Systems	1995
The ANALYZE rulebase for supporting LP analysis	Greenberg, H. J.	Annals of Operations Research	1996
RMT: A modeling support system for model reuse	Kwon, O. B., Park, S. J.	Decision Support Systems	1996
A structured modeling based methodology to design decision support systems	Raghunathan, S.	Decision Support Systems	1996
Object-oriented model construction in production scheduling decisions	Pillutla, S. N., Nag, B. N.	Decision Support Systems	1996
Type and inheritance theory for model management	Ma, J.	Decision Support Systems	1997
Logic, modeling, and programming	McAloon, K., Tretkoff, C.	Annals of Operations Research	1997
An implementation of a graph-based modeling system for structured modeling (GBMS/SM)	Chari, K., Sen, T. K.	Decision Support Systems	1998
Object relational approach for the design of decision support systems	Srinivasan, A., Sundaram, D.	European Journal of Operational Research	2000
OMNI Model Management System	Haverly, C. A.	Annals of Operations Research	2001
Advances in design and implementation of optimization software	Maros, I., Khaliq, M. H.	European Journal of Operational Research	2002
Design and implementation of an optimization-based decision support system generator	Maturana, S., Ferrer, J. C., Barañao, F.	European Journal of Operational Research	2004
A real-time synchronization mechanism for collaborative model management	Huh, S. Y., Kim, H. M.	Decision Support Systems	2004
A structured modeling technology	Makowski, M.	European Journal of Operational Research	2005

A.5 RESULTADOS OBTIDOS

Após a identificação das características mais importantes levantadas em cada um dos trabalhos, elas foram agrupadas em tópicos afins para possibilitar uma análise quantitativa dos resultados obtidos no estudo baseado em revisão sistemática da literatura. Essa análise, que pode ser vista na Tabela A.4, representa a resposta da questão proposta nesse estudo: quais os

requisitos mais importantes que devem ser atendidos em um ambiente de modelagem matemática, tanto para os desenvolvedores dos modelos quanto para os seus usuários, os tomadores de decisão?

Observando os resultados extraídos da bibliografia identificada, dois grupos principais de requisitos foram identificados: requisitos sobre a obtenção de dados de ferramentas externas e integração com sistemas gerenciadores de bancos de dados e requisitos sobre a modelagem utilizando elementos gráficos. Neste segundo grupo existe uma variedade de características como a modelagem através de desenhos de grafos, utilização de estruturas de blocos, uso de interfaces amigáveis para tratamento de dados e resultados através do clique e arraste do *mouse*. Cada um desses dois grupos de requisitos foram citados em praticamente metade dos trabalhos analisados.

Tabela A.4: Análise quantitativa das características encontradas nos trabalhos

Características	**% aproximada de trabalhos**
Modelagem com uso de gráficos, utilização de estruturas gráficas, interfaces amigáveis para modelagem e análise dos resultados	50%
Integração com bancos de dados ou planilhas, aquisição automática dos dados, separação dos dados do modelo	46%
Modularização, reuso e aplicação de conceitos de orientação a objetos (herança, agregação etc.) na construção de modelos matemáticos	43%
Uso de linguagens algébricas e de lógica de programação para modelagem de problemas de pesquisa operacional	36%
Apoio à utilização de diferentes resolvedores e interfaces genéricas	18%
Validação pelos usuários e verificação do modelo	18%
Controle de versão e evolução dos modelos	14%
Documentação do modelo e automatização da documentação	11%
Análise de resultados e *debug* de inviabilidade	11%
Modelagem e uso em ambiente distribuído	7%

Muitos trabalhos citam a separação dos dados do modelo, principalmente através do uso de aplicações de banco de dados e planilhas eletrônicas para armazenar os dados. O tratamento dos dados se mostrou de grande relevância, tanto em trabalhos mais antigos quanto nos mais recentes. A integração com banco de dados é umas das características mais citadas. Na maior parte dos trabalhos, a separação do modelo dos dados de uma instância é tratada como fundamental, e o uso do banco de dados pode ser útil nesta tarefa, tanto para armazenamento dos dados, quanto dos modelos.

Em cerca da metade dos trabalhos, é possível observar características como o uso de conceitos da orientação a objetos na modelagem matemática, a divisão dos modelos em blocos ou módulos permitindo o reuso de componentes e a composição do modelo através de blocos de submodelos. Além dessas características, foi possível observar considerações sobre as linguagens de modelagem em cerca de um terço dos trabalhos. Nesses trabalhos são abordados principalmente as linguagens algébricas de modelagem e o uso de lógica de programação e elementos de programação procedural na modelagem matemática. Eles destacam que, por mais funcionalidades que um ambiente de modelagem possua, em algum nível uma linguagem algébrica e de programação deve ser utilizada na criação dos modelos.

Outra característica bastante citada e com bastante relevância é a independência das ferramentas utilizadas na resolução dos problemas de otimização. Assim, muitos trabalhos citam a flexibilidade do uso de uma vasta gama de opções de resolvedores matemáticos. O ambiente de modelagem deve ser o mais independente possível dos resolvedores, permitindo uma integração com as melhores opções disponíveis no mercado e na academia. Além desses requisitos citados, pode-se identificar características como a validação e verificação lógica dos modelos, permitindo ao modelador, em tempo de desenvolvimento, saber se o modelo está logicamente correto. A documentação e a escrita do modelo matemático em uma linguagem bem próxima de uma linguagem natural, facilitando assim o entendimento do modelo e do problema que ele se propõe a resolver, também são características levantadas. Além disso, a análise de resultados e análise de inviabilidade também são citadas em alguns trabalhos.

Características como versionamento, controle da evolução e controle do ciclo completo do modelo matemático (do desenvolvimento até a utilização) foram levantadas em alguns trabalhos, assim como o uso de modelos para a tomada de decisão em ambientes distribuídos, com pessoas envolvidas no processo em diferentes locais. Porém essas características se mostraram ainda pouco exploradas.

APÊNDICE B - QUESTIONÁRIO E SUAS RESPOSTAS

Total de participantes: 19

1. Formação acadêmica:

(a) Doutorado – **11%**

(b) Doutorando – **21%**

(c) Mestrado – **42 %**

(d) Mestrando – **5%**

(e) Graduação – **16%**

(f) Graduando – **5%**

2. Qual é sua experiência com desenvolvimento de modelos matemáticos? (pode marcar mais de uma opção)

(a) Nunca desenvolvi um modelo matemático, apenas conheço as técnicas – **0%**

(b) Já fiz um curso/disciplina sobre modelagem matemática e programação linear e/ou inteira – **30%**

(c) Desenvolvi modelos matemáticos usando técnicas de programação linear e/ou inteira para trabalhos acadêmicos (projetos finais, teses, artigos, …) – **24%**

(d) Desenvolvi modelos matemáticos usando técnicas de programação linear e/ou inteira para a indústria – **46%**

3. Os problemas modelados por você para a resolução usando métodos exatos normalmente são: (pode marcar mais de uma opção)

(a) Linear – **42%**

(b) Inteiro – **45%**

(c) Não linear – **13%**

4. Quais as linguagens de modelagem matemática que você usa com mais frequência? (pode marcar mais de uma opção)

(a) AIMMS – **41%**

(b) AMPL – **3%**

(c) GAMS – **13%**

(d) LINDO – **6%**

(e) SIGMO – **19%**

(f) Outras. – **18%**

Especificar:_____

Outras citadas:

- *Resolvedor do Excel c/ suas extensões desenvolvidas pela Frontline*
- *C++*
- *FLOPC++*
- *OpenCalc*
- *Específico em Access: Modelo da Petrobras*
- *XPRESS*

5. Quais as ferramentas usadas para implementar um modelo matemático que você usa com mais frequência? (pode marcar mais de uma opção)

(a) AIMMS – **38%**

(b) AMPL – **3%**

(c) GAMS – **3%**

(d) LINGO – **3%**

(e) SIGMO – **15%**

(f) Editores de texto – **21%**

(g) IDE's de desenvolvimento de software (Eclipse, Netbeans) – **3%**

(h) Outras. Especificar:_____ – **14%**

Outras citadas:

- *Concert*
- *MS-Excel, OpenCalc*

- *Access*
- *XPRESS-IVE (FICO)*

6. A elaboração e implementação de um modelo matemático normalmente é feito por quantos desenvolvedores?

(a) Somente por mim – **16%**

(b) Em dupla – **21%**

(c) Equipe com até 5 desenvolvedores – **58%**

(d) Equipe de com mais de 5 desenvolvedores – **5%**

7. Ao final, o modelo desenvolvido é usado no apoio à decisão por quem? (pode marcar mais de uma opção)

(a) Somente por mim – **10%**

(b) Por mim e por outros membros da minha equipe, que conhecem a modelagem – **7%**

(c) Por outra pessoa que conhece o problema, mas não conhece a modelagem – **21%**

(d) Por outra equipe de até 5 usuários que não conhecem a modelagem – **28%**

(e) Por outra equipe com mais de 5 usuários que não conhecem a modelagem – **34%**

8. Os modelos desenvolvidos são executados com qual frequência? (pode marcar mais de uma opção)

(a) Somente uma vez – **0%**

(b) Pouco frequente, sob demanda – **17%**

(c) Muito frequente, sem periodicidade definida – **61%**

(d) Com uma frequência bem definida (diariamente, semanalmente, …) – **22%**

9. Uma vez implementado o modelo, normalmente quem faz a sua manutenção?

(a) Somente eu – **21%**

(b) Eu e os outros membros da minha equipe, que participaram do desenvolvimento – **74%**

(c) É feita por outro profissional ou equipe que não participou do desenvolvimento – **5%**

10. Na maior parte das vezes os modelos desenvolvidos são executados nas mesmas máquinas em que são implementados?

(a) Sim – **11%**

(b) Não – **89%**

11. Como os dados necessários para a execução do modelo são armazenados?

(a) Junto com as informações do modelo – **0%**

(b) Separado das informações do modelo, mas ficam na própria ferramenta usada para a modelagem – **32%**

(c) Em fontes externas que deve ser importadas antes da execução do modelo (banco de dados, sistemas externos, arquivos texto, planilhas, …) – **68%**

12. Quem pode tratar os dados de entrada do modelo sem alterar a modelagem? (pode marcar mais de uma opção)

(a) Eu, que desenvolvi o modelo – **23%**

(b) Membros da minha equipe, que participaram do desenvolvimento – **19%**

(c) Outro profissional ou equipe que não participou do desenvolvimento do modelo – **58%**

13. Você usa controle de versão de desenvolvimento de modelos matemáticos?

(a) Nunca – **16%**

(b) Raramente – **21%**

(c) Na maioria das vezes – **42%**

(d) Sempre – **21%**

14. Nos modelos desenvolvidos é possível identificar em cada uma das versões: (pode marcar mais de uma opção)

(a) Quem realizou a alteração – **31%**

(b) Quem solicitou a alteração – **4%**

(c) Quando foi feita a alteração – **46%**

(d) Por que a alteração é necessária – **19%**

15. Os dados referentes ao modelo e os resultados estão de alguma forma associados a uma versão específica do modelo elaborado?

(a) Sim – **84%**

(b) Não – **16%**

16. Existem versões do mesmo modelo que devem ser mantidas (efetuar correções e melhorias) em paralelo, ou seja, as duas versões podem ser executadas para obtenção de resultados?

(a) Sim – **67%**

(b) Não – **33%**

17. O controle de versão é integrado à ferramenta usada para a implementação do modelo matemático?

(a) Sim – **5%**

(b) Não – **95%**

18. Você ou sua equipe são responsáveis pela manutenção dos dados usados na execução do modelo?

(a) Nunca – **28%**

(b) Raramente – **56%**

(c) Na maioria das vezes – **6%**

(d) Sempre – **10%**

19. Quando o modelo sofre uma alteração você precisa se preocupar com as modificações dos dados?

(a) Nunca – **11%**

(b) Raramente – **16%**

(c) Na maioria das vezes – **47%**

(d) Sempre – **26%**

20. Os dados usados em versões anteriores do modelo devem ser modificados para a versão mais atual?

(a) Nunca – **11%**

(b) Raramente – **32%**

(c) Na maioria das vezes – **53%**

(d) Sempre – **4%**

21. Quanto à documentação da formulação matemática de um modelo, normalmente:

(a) O próprio fonte do modelo matemático é suficiente para entender e explicar a formulação do problema **– 25%**

(b) A formulação do problema é documentada separada do modelo matemático, feita antes ou depois **– 63%**

(c) A partir de um documento com a formulação eu gero automaticamente o modelo matemático ou boa parte dele já implementado usando uma ferramenta. Qual? **– 0%**

(d) A partir do modelo matemático implementado eu gero automaticamente a documentação da formulação ou boa parte dela usando uma ferramenta. Qual? **R. SIGMO – 12%**

22. Qual a importância que você identifica em cada uma das características para um ambiente de modelagem matemática? (nota de 1 a 5)

(a) Performance na montagem do problema para passagem para o resolvedor
(b) Integração com a maior quantidade de resolvedores possível
(c) Facilidade com a linguagem de modelagem
(d) Facilidade de integração dos dados com o modelo
(e) Facilidade de tratamento dos dados do modelo
(f) Integração com sistemas externos
(g) Controle de versão no modelo matemático
(h) Geração de documentação automática
(i) Facilidade de separação das informações do modelo dos dados
(j) Facilidade na entrada de dados e na visualização dos resultados

Resultado:

NOTA	1	2	3	4	5
Performance na montagem do problema para passagem para o resolvedor	0%	0%	17%	28%	55%
Integração com a maior quantidade de resolvedores possíveis	0%	6%	33%	39%	22%
Facilidade com a linguagem de modelagem	0%	0%	17%	22%	61%
Facilidade de integração dos dados com o modelo	6%	0%	0%	33%	61%
Facilidade de tratamento dos dados do modelo	6%	0%	11%	39%	44%
Integração com sistemas externos	6%	6%	39%	33%	16%
Controle de versão no modelo matemático	11%	22%	39%	11%	17%
Geração de documentação automática	17%	39%	11%	17%	16%
Facilidade de separação das informações do modelo dos dados	0%	0%	24%	53%	23%
Facilidade na entrada de dados e na visualização dos resultados	6%	0%	17%	28%	49%

23. Quais funcionalidades não descritas acima você desejaria em um ambiente de modelagem matemática?

Respostas livres:

- *Tratamento integrado dos dados com o modelo e validação dos dados antes da otimização*
- *O uso de um modelador matemático que seja uma biblioteca ou um pacote para uma linguagem de programação daria grande flexibilidade no desenvolvimento de otimizadores modulares. Com esse conceito, só conheço o FlopC++ da Coin-OR.*
- *Seria interessante se o ambiente fosse capaz de, dado um modelo (que poderia ser escrito em uma linguagem como LaTeX), exportar este modelo com base em uma instância da base nos diversos formatos utilizados pelos resolvedores. Seria interessante que o usuário pudesse descrever textualmente as restrições para consulta posterior. O controle de versão também é muito interessante. Além das versões para um mesmo modelo, poderia haver também uma forma de agrupar os diferentes modelos desenvolvidos para o mesmo problema.*
- *Possibilidade de gerar (parte do) equacionamento e (parte da) interface via código; controle de unidades dos elementos presentes no equacionamento; servidor com serviço para gerenciamento de múltiplas versões/controle de acesso/gerenciamento de fila integrado; computação distribuída integrada (grid computing).*
- *Geração de uma DLL do modelo semelhante ao que faz o Flop C++. Acho que um bom modelador deveria ser amigável como o Aimms com possibilidade geração de uma DLL.*