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

Fernando Lourenço Pinho Costa
Universidade Federal Fluminense – Instituto de Computação
Rua Passo da Pátria 156 - Bloco E - 3º andar - Niterói, RJ 24210-240
flpcosta@ic.uff.br

Leonardo Gresta Paulino Murta
Universidade Federal Fluminense – Instituto de Computação
Rua Passo da Pátria 156 - Bloco E - 3º andar - Niterói, RJ 24210-240
leomurta@ic.uff.br

Celso Carneiro Ribeiro
Universidade Federal Fluminense – Instituto de Computação
Rua Passo da Pátria 156 - Bloco E - 3º andar - Niterói, RJ 24210-240
celso@ic.uff.br

RESUMO

Este trabalho 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 (hardware e software) das que geram e manipulam os modelos e seus dados.

PALAVRAS CHAVE. Modelagem Matemática, Programação Linear, Controle de Versão.

Área principal: Otimização, Gerência de Configuração.

ABSTRACT

This paper addresses characteristics of software environments for mathematical modeling and proposes a system for modeling and management models of linear and integer programming problems. The main features of this modeling environment are: version control of the evolution 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 (hardware and software).

KEYWORDS. Mathematical Modeling, Linear Programming, Version Control.

Main area: Optimization, Configuration Management.

1. Introdução

Um ambiente de modelagem é um software que permite descrever matematicamente um determinado problema real a fim de resolvê-lo usando técnicas de programação linear e inteira, por exemplo. A principal função dos ambientes de modelagem é fornecer aos usuários uma ferramenta para que o problema possa ser modelado da melhor forma possível, o mais próximo da realidade. Além disso, este tipo de sistema deve ser capaz de converter o modelo e os dados do problema para o formato que os resolvedores possam entender e gerar soluções.

Este trabalho trata da elaboração de um sistema para gerenciar e desenvolver modelos de programação linear e inteira. Diferentemente da maioria dos softwares de modelagem, ele se propõe não apenas a fornecer uma ferramenta para a escrita de equações matemáticas, mas também controlar o ciclo de vida de um modelo, deste a sua elaboração até a sua utilização pelos tomadores de decisão. A motivação principal da elaboração deste sistema vem da associação do processo de desenvolvimento de um modelo matemático, que será utilizado dentro da cadeia de tomada de decisão de uma organização, com o processo de desenvolvimento de software. A área da engenharia de software está bastante evoluída e possui conceitos que podem ser aplicados também na construção e manutenção de modelos matemáticos e sistemas de apoio à decisão.

O processo de desenvolvimento de software é um conjunto de tarefas que devem ser executadas para a construção de um software que atenda às necessidades definidas pelos seus futuros usuários. Pressman (2006) cita, em um modelo de desenvolvimento em cascata, cinco fases, com diferentes tarefas em cada uma delas: 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 cujas principais contribuições são o controle de versão, o controle de mudança e a auditoria na elaboração do software. 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 acima podem ser observadas. Atividades como levantamento de requisitos junto aos usuários (neste caso, os tomadores de decisão), o projeto e a implementação da solução, o teste, a implantação e a manutenção também podem ser encontradas na elaboração de sistemas de apoio a decisão.

Como motivação, pode-se utilizar como exemplo a área de Pesquisa Operacional de uma grande companhia, que desenvolve diversos modelos para resolver problemas de otimização nas diversas áreas de negócio. Além do grande desafio de modelar o problema real e achar boas soluções, existem desafios da área de desenvolvimento de software que vão desde o suporte à solução de Pesquisa Operacional até a entrega de um produto de software para o usuário, que passa a utilizá-lo no dia-a-dia de tomada de decisão da área de negócio. O produto entregue deve ter uma interface amigável, que permita criar diversos cenários e obter os resultados facilmente. A qualidade de uma solução de Pesquisa Operacional será medida principalmente pelos resultados obtidos, mas também pela facilidade de interação entre o usuário, a aplicação e os dados necessários. Assim, alguns desafios do desenvolvimento de software identificados para sistemas de Pesquisa Operacional são: gerenciamento dos dados necessários para a modelagem do problema e dos diferentes cenários; a arquitetura da aplicação; interface homem/máquina (IHM); e a obtenção e o tratamento dos dados.

Na próxima seção desse artigo, é apresentada a revisão da literatura com os principais trabalhos que serviram de base ou que podem complementar este ambiente de modelagem. Na Seção 3 é descrita a arquitetura geral do sistema. Na Seção 4 são descritos os elementos e o mecanismo de modelagem matemática. Na Seção 5 é apresentado o tratamento dos dados dos modelos. Na Seção 6 é feito o detalhamento da estratégia de controle de evolução dos modelos e dos dados. Finalmente, na Seção 7 são apresentadas as conclusões e delineados trabalhos futuros.

2. Revisão da literatura

Geoffrion (1989) descreve cinco características desejáveis para ambientes de modelagem usando computador para suportar 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 solução do problema; permitir o fácil gerenciamento de recursos usados no processo de modelagem, como os dados, os modelos e os resolvedores, por exemplo. É possível observar que a maior parte dessas características descritas pode ser obtida aproveitando os conceitos da Engenharia de Software e, mais especificamente, de gerência de configuração.

O objetivo principal do ambiente apresentado neste trabalho é a modelagem de problemas de programação linear e inteira. Murphy, Stohr e Asthana (1992) apresentam uma série de esquemas para representação de problemas de programação linear. Dentre essas representações é possível destacar os geradores de matrizes, as representações algébricas, a modelagem estruturada e o uso de esquemas de banco de dados. Os geradores de matrizes foram os primeiros modeladores, como por exemplo, o OMNI (Haverly, 2001). Eles fornecem uma linguagem que permite gerar o formato MPS (IBM, 1975), que representa uma instância do problema de programação linear ou inteira. A representação algébrica, onde os modelos são descritos na forma de expressões matemáticas, usada em sistemas atualmente conhecidos, como GAMS (Bisschop e Meeraus, 1982) e o AMPL (Fourer, Gay e Kernighan, 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 banco de dados. Existem dois requisitos distintos nesta representação, mas que devem estar relacionados: a necessidade de registrar informações sobre a estrutura do modelo matemático e a necessidade de armazenar e manipular os dados do problema e os resultados que são obtidos a partir do otimizador. Além dessas representações, ainda existem as formas gráficas que usam grafos ou esquemas de blocos para representar os problemas.

Fourer (1997) apresenta uma solução de modelagem usando estrutura de bancos de dados. Apesar de ser uma solução específica para um problema de programação matemática, mostra que é uma abordagem interessante para problemas de grande escala. Atualmente, os sistemas gerenciadores de banco de dados (SGDBs) são amplamente utilizados nas organizações, gerindo e centralizando as principais informações das companhias. Logo, os sistemas de apoio à decisão de alguma forma precisam estar integrados com estes repositórios de informações.

Muitas tecnologias atuais também podem auxiliar na construção de sistemas de apoio a decisão, além dos sistemas de banco de dados. Fourer (1998) já trata do uso de tecnologia *Web* em sistemas de otimização e Fourer e Ma (2010) propõem um servidor de otimização utilizando *Web Services* e o formato *Extensible 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. O conceito desse tipo de serviço de otimização também está presente no sistema apresentado neste trabalho.

3. Arquitetura do sistema

Para a definição da arquitetura do sistema, foram identificados dois papéis distintos e essenciais no desenvolvimento de projetos de solução em Pesquisa Operacional, utilizando modelagem matemática. 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 em 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 utilizando uma arquitetura

cliente-servidor. Com essa arquitetura, é possível ter módulos para atender diferentes funções que se interligam através da aplicação central e da base de dados única. Além disso, permite que seja um sistema multiusuário, com principalmente dois tipos de perfis: modeladores e tomadores de decisão. Na Figura 1 pode ser vista a arquitetura geral do sistema. Basicamente, os modeladores interagem com a aplicação criando e mantendo os modelos matemáticos para resolver os problemas de otimização. Os tomadores de decisão manipulam os dados, com possibilidade de criação de diversos cenários para um mesmo problema, e os resultados obtidos pelos resolvedores. Com essa estrutura, cada usuário do sistema tem um ambiente adequado para realizar o seu trabalho, ou de modelagem ou de uso do modelo, mas esses ambientes podem ser compartilhados e interligados pela base de dados única.

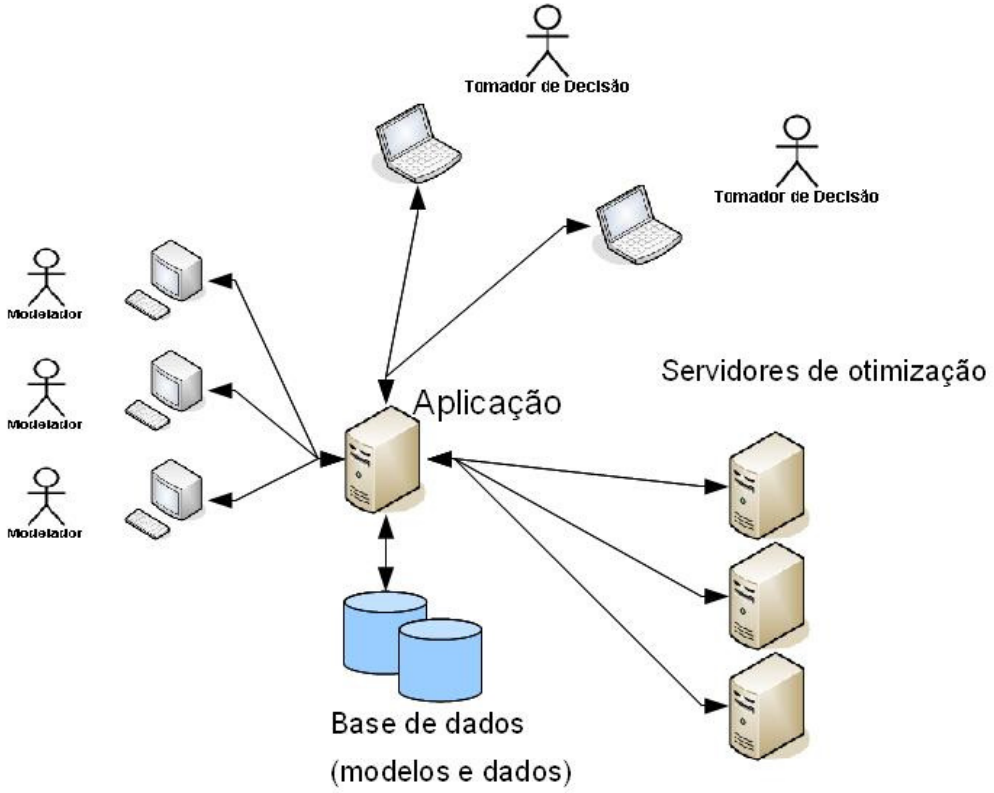

Figura 1. Arquitetura básica do sistema

Outro elemento importante na arquitetura do sistema são os servidores de otimização. Para problemas de grande porte, que precisam de um grande poder de processamento para encontrar soluções ótimas em um tempo adequado, normalmente são utilizadas máquinas com hardware e software dedicados para a execução dos processos de otimização. Além disso, é bastante desejável que essas máquinas possam ser compartilhadas para resolver diferentes tipos de problema, em momentos distintos. Muitas vezes, o computador de mesa não tem a capacidade adequada para executar o processamento necessário. Desta forma, aparece na arquitetura do sistema a figura dos servidores de otimização. Neste ponto, este trabalho pode ser complementado com a proposta de Fourer e Ma (2010), que permite a execução remota de problemas de otimização utilizando *Web Services* para passar os dados de um problema via XML para ser resolvido remotamente. O servidor da aplicação também é responsável por gerenciar as execuções das otimizações dos usuários, também chamadas de "corridas", controlando a distribuição entre os servidores específicos para otimização, a concorrência das corridas e até a prioridade de um determinado usuário ou projeto sobre outros. Os servidores de otimização devem se comunicar com os resolvedores. A versão atual do sistema proposto faz uso dos resolvedores CPLEX (IBM, 2009) e GUROBI (2011).

Outra característica importante do sistema é a utilização de controle de versão, tanto

para os modelos matemáticos quanto para os dados associados. Com esse controle, todos os elementos do sistema podem ser rastreados e as mudanças podem ser identificadas. Na Seção 6, são apresentados mais detalhes sobre o controle de versão desenvolvido.

4. Modelagem matemática para problemas de programação linear e inteira

Apesar do sistema descrito neste trabalho utilizar banco de dados para armazenar os modelos e os dados dos problemas de programação linear e inteira, os elementos de modelagem foram projetados observando as representações algébricas, principalmente da linguagem AMPL. 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 problemas de otimização apenas alterando os dados de entrada.

Para a modelagem dessa classe de problemas, são utilizados cinco elementos principais: conjuntos, parâmetros, variáveis, restrições e função objetivo. Para identificar esses elementos em um problema de modelagem, é utilizado como exemplo um modelo de programação linear bastante simples, o problema de transporte proposto por Dantzig (1963), que pode ser descrito da seguinte forma:

$$Min \sum_{j \in J} \sum_{i \in I} c_{ij} . x_{ij} \quad (1)$$

Sujeito a:

$$\forall i \in I : \quad \sum_{j \in J} x_{ij} \leq a_i \quad (2)$$

$$\forall \ j \in J : \quad \sum_{i \in I} x_{ij} \geq b_j \quad (3)$$

$$x_{ij} \geq 0 \quad (4)$$

Este problema considera produtores e mercados consumidores de um produto específico, definidos respectivamente pelos conjuntos *I* e *J*. Seu objetivo consiste em minimizar o custo de transporte entre os produtores até os mercados consumidores (1), de forma que a capacidade máxima de produção de cada produtor (2) e a demanda mínima de cada mercado consumidor (3) sejam atendidas. Os dados de entrada para esse problema são: o conjunto *I* de produtores, o conjunto *J* de mercados consumidores, os limites máximo de produção (a_i) de cada produtor, os limites mínimo de demanda (b_j) a ser atendida em cada mercado consumidos e o custo unitário (c_{ij}) de transporte de cada produtor para cada mercado consumidor. Especificamente no exemplo descrito nesse trabalho, o valor de c_{ij} é dado pela distância (d_{ij}) entre cada produtor e mercado consumidor multiplicada por um fator (*f*) que representa um custo constante por unidade de distância. O resultado procurado é a quantidade (x_{ij}) que deve ser produzida e transportada de cada produtor *i* para cada mercado consumidor *j*.

Os conjuntos são coleções de objetos pertinentes ao problema. Eles representam as entidades principais dos problemas e, normalmente, os outros elementos de modelagem se referem aos objetos pertencentes aos conjuntos. Os conjuntos possuem índices associados. O índice permite referenciar um elemento genérico dentro do conjunto. Eles são essenciais na descrição dos problemas de programação linear e inteira. Um tipo bastante comum de conjunto, por exemplo, é uma sequência numérica. É possível também a criação de subconjuntos. No exemplo do problema de transporte, dois conjuntos são usados na modelagem: o conjunto *I* de produtores e o conjunto *J* de mercados consumidores. Os índices *i* e *j* estão associados respectivamente a esses conjuntos.

Os parâmetros no modelo matemático são invariantes que são usados na modelagem. Normalmente representam os dados necessários para gerar o modelo. Eles podem ser valores que devem ser informados diretamente pelo usuário (dados de entrada) ou podem ser calculados antes da geração do modelo a partir de uma fórmula. Os parâmetros podem ser valores independentes, que não dependem de nenhum outro objeto, ou podem ser indexados pelos elementos de um ou mais conjuntos. Neste caso, os parâmetros funcionam como vetores ou matrizes cujos índices são os elementos dos conjuntos.

Um exemplo de parâmetro deste problema é o custo de transporte do produtor i para o mercado consumidor j, dado por c_{ij}. Este parâmetro deve ser indexado pelos elementos do conjunto de produtores I e de mercados consumidores J, uma vez que cada par (i,j) possui um valor de custo associado. Porém, esse parâmetro não é dado diretamente no caso específico do problema tratado nesse exemplo. A informação disponível é a distância entre cada local, dado por d_{ij}, e o fator de custo unitário por distância, dado por f. Logo, o parâmetro c_{ij} deve ser calculado pela expressão "$c_{ij} = f * d_{ij}$". O sistema permite parâmetros calculados a partir de expressões contendo constantes e outros parâmetros. Os operadores aritméticos possíveis são mostrados na Tabela 1.

Tabela 1. Operadores aritméticos

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

As variáveis são os elementos de modelagem que devem ser calculados pelo processo de otimização. Elas são definidas no sistema da mesma forma que os parâmetros. As variáveis também podem ser indexadas pelos objetos dos conjuntos. No exemplo, as variáveis são as quantidades que devem ser transportadas entre cada produtor e mercado consumidor. Logo, elas podem ser representadas como um elemento único que possui dois índices, i e j. Uma variável pode ser do tipo real, inteiro ou binário, uma vez que o sistema permite a modelagem de problemas de programação linear e inteira. Além do seu tipo, uma variável deve ter definido os seus valores limites, ou seja, sua faixa de validade.

As restrições são representadas por equações ou inequações lineares. No exemplo apresentado as restrições podem ser vistas nas inequações (2) e (3). Elas são responsáveis por limitar o problema. A função objetivo é uma expressão linear que o problema de otimização deve procurar maximizar ou minimizar, no exemplo apresentado ela pode ser vista na expressão (1).

Tabela 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		

Para permitir a modelagem de problemas mais complexos do que o exemplo apresentado, o ambiente de modelagem proposto neste trabalho trata as condições de geração de variáveis e restrições. As condições de geração são expressões booleanas que podem usar constantes, valores de parâmetros e valores de índices para determinar a condição em que as

variáveis e restrições devem ser geradas no problema que é passado para o resolvedor. São permitidos nessas expressões operadores aritméticos que podem ser vistos na Tabela 1 e operadores lógicos e de comparação, que podem ser vistos na Tabela 2.

5. Tratamento dos dados

Os conjuntos, índices e parâmetros permitem generalizar a modelagem a fim de viabilizar a criação de diferentes instâncias do mesmo problema de otimização sem que haja necessidade de alterar a definição do modelo matemático. Um problema generalizado deve possuir em sua modelagem conjuntos, índices e parâmetros, e estes podem ser utilizados na definição das variáveis, restrições e função objetivo. Os elementos contidos nos conjuntos e os valores dos parâmetros representam os dados de entrada para o problema de otimização. Os valores das variáveis e o valor da função objetivo, representam o resultado do problema. Na abordagem utilizada neste trabalho, uma instância ou cenário de dados de um problema de programação linear ou inteira é o grupo de informações composto pelos dados de entrada e pelo resultado obtido, se o problema já foi otimizado. Logo, um cenário de dados são os elementos contidos nos conjuntos e os valores dos parâmetros, assim como o valor das variáveis e da função objetivo, se a otimização já foi executada. Desta forma, para o mesmo modelo é possível gerar diferentes problemas de programação linear ou inteira apenas alterando o cenário de dados, sem alterar a modelagem matemática do problema.

Cada modelo matemático vai precisar dos dados em uma estrutura diferente. Em um determinado modelo, podem ser criados um ou mais conjuntos, que podem ser usados para indexar diversos parâmetros, variáveis e restrições, com diferentes combinações. A indexação dos elementos de modelagem se assemelha com as chaves primárias das entidades de banco de dados. Cada entidade possui campos que caracterizam a sua chave primária, ou seja, conjunto de um ou mais atributos cujos valores, considerando a combinação de valores de todos os atributos da chave primária, nunca se repetem. Associando com os modelos matemáticos, cada atributo na chave primária pode ser comparado com um índice e a chave primária com a combinação dos índices que indexam os elementos de modelagem, ou seja, variáveis, restrições e parâmetros.

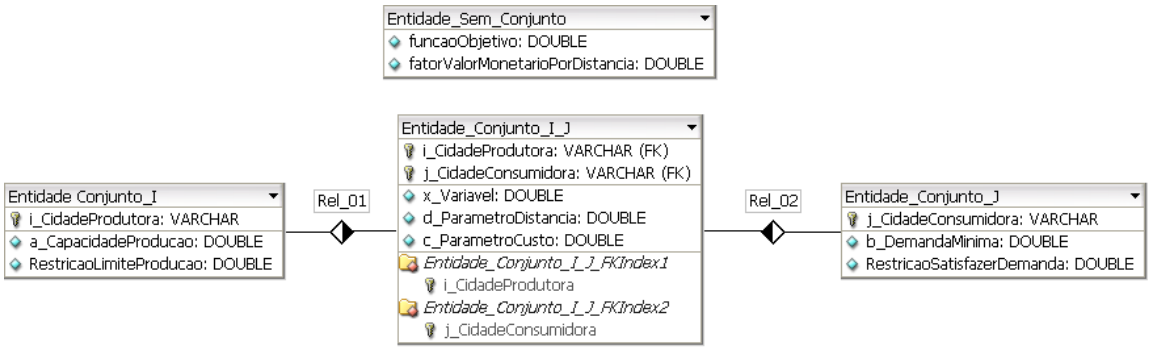

Figura 2. Diagrama de entidades e relacionamentos para o problema de transporte

Na abordagem deste trabalho, é possível afirmar que, para um dado modelo matemático, sempre vai existir uma entidade de banco de dados (tabela) para cada um dos conjuntos definidos e essa entidade deve conter um atributo onde são armazenados os valores dos elementos desse conjunto. Este atributo é a própria chave primária da entidade. Ainda pode ser afirmado que, para armazenar os valores dos parâmetros e variáveis, vai existir uma entidade de banco de dados para cada combinação diferente dos conjuntos associados aos índices dos parâmetros, restrições e variáveis. Cada uma dessas entidades deve possuir um atributo para cada índice, e esses atributos associados aos índices compõem a chave primária da entidade. Todas as variáveis, parâmetros e restrições que possuem a mesma combinação de índices, devem ter um atributo relacionado na mesma entidade de banco de dados, fora da chave primária, para armazenar o valor da respectiva variável, parâmetro ou restrição. Uma entidade isolada também

deve ser criada para conter todos os valores dos elementos que não são indexados, como o valor da função objetivo, por exemplo, ou parâmetros, variáveis e restrições que não possuem índices. Essa entidade isolada sempre vai possuir apenas um registro, uma vez que os valores não são indexados. Cada cenário de dados de um problema de programação linear e inteira vai possuir apenas um valor para os elementos não indexados.

No exemplo do problema de transporte existem elementos indexados pelas seguintes combinações de índices: sem índice, somente *i*, somente *j* e *i* com *j*. Na Figura 2 pode ser visto o diagrama de entidades e relacionamentos que representa a estrutura para armazenar os dados e os resultados deste problema. Dadas as combinações existentes dos índices, foram criadas quatro entidades: uma entidade isolada para o fator *f*, que não possui índice, e para a função objetivo; uma entidade que representa o conjunto *I*, com um atributo para os elementos do conjunto, que é a chave primária da entidade, um atributo para o parâmetro a_i e um para a restrição de limite de produção (2), que é indexada por *i*; uma entidade que representa o conjunto *J*, com um atributo para os elementos do conjunto (chave primária), um atributo para o parâmetro b_j e um atributo para a restrição de demanda mínima (3), que é indexada por *j*; e uma entidade que representa os elementos de modelagem indexados por índices dos conjuntos *I* e *J*, com dois atributos na chave primária que indicam os valores dos índices *i* e *j*, e um atributo para cada elemento de modelagem indexado por *i* e *j*, variável x_{ij} e os parâmetros d_{ij} e c_{ij}. Nesta última entidade, os atributos com os valores de *i* e *j*, além de comporem a chave primária, são chaves estrangeiras do relacionamento desta entidade com as entidades que representam os conjuntos *I* e *J*, respectivamente (relacionamentos *Rel_01* e *Rel_02* da Figura 2).

Um ponto a ser observado é que cada modelo matemático diferente possui um conjunto de entidades e relacionamentos distinto. Para que o sistema seja genérico e possa ter uma estrutura de banco de dados uniforme, que atenda a qualquer tipo de problema de programação linear ou inteira, a solução dada foi a utilização de meta-modelagem. Na meta-modelagem é possível criar entidades, relacionamentos e atributos de forma virtual em um banco de dados, ao invés de criar as tabelas fisicamente no esquema.

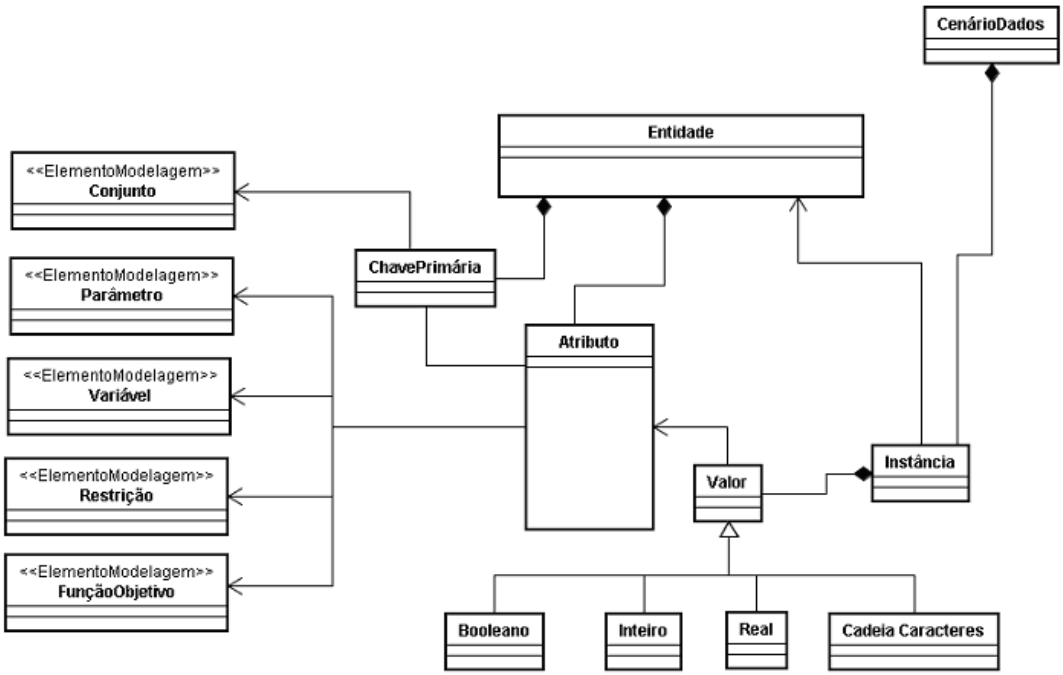

Figura 3. Meta-modelagem para armazenar os dados de um problema de programação linear

A meta-modelagem utiliza tabelas de uso geral para criar qualquer modelo de entidades e relacionamentos que represente qualquer problema de programação linear e inteira. Assim, toda vez que um modelo matemático estiver pronto, a estrutura para persistir seus dados é criada

através da meta-modelagem e o modelo fica disponível para ser utilizado. Na figura 3 pode ser vista a meta-modelagem feita para armazenar os dados de um problema de programação linear. Comparando com o diagrama de entidade e relacionamento da figura 2, pode ser observado que os elementos dos conjuntos estão associados com as chaves primárias das entidades e que as variáveis, restrições, parâmetros e função objetivo estão relacionados com os atributos.

Cada registro na tabela *Entidade* representa uma tabela do modelo de dados, cada coluna da tabela é um registro em *Atributo* e cada linha é um registro em *Instância*. Uma instância ou linha representa um conjunto de valores, um registro ou tupla da entidade. Ainda na figura 3, pode-se observar a tabela *CenárioDados* que agrupa as instâncias ou linhas. Desta forma, é possível ter vários cenários de dados diferentes para um mesmo modelo matemático. Cada modelo matemático diferente possuirá seus registros nas tabelas de uso geral *Entidade*, *ChavePrimária* e *Atributo*. E cada conjunto de dados de um modelo específico possuirá seus registros nas tabelas *CenárioDados, Instância* e *Valor*.

6. Controle de versão

No ambiente, o projeto de modelagem representa um problema de programação linear e inteira que deve ser modelado e resolvido. Ele agrupa os elementos de modelagem (conjuntos, parâmetros, variáveis, restrições e função objetivo). A evolução dos modelos ocorre com a criação de versões para o mesmo projeto. O conceito de uma versão do projeto de modelagem é o conjunto de versões dos elementos que o compõem. Desta forma, cada elemento possui a sua versão individualmente, porém seu ciclo de vida está sempre associado a um determinado projeto. Uma vez editado, o elemento passa ter a mesma versão do projeto. Por exemplo, como pode ser visto na Figura 4, o projeto está na sua terceira versão, porém a maioria dos elementos ainda está na primeira versão, e isso quer dizer que eles não foram editados posteriormente à sua criação. Já a restrição $AtendimentoDemanda_j$, em destaque na Figura 4, foi editada na versão atual. Com esse tipo de controle é possível identificar todas as alterações feitas em cada uma das versões e o respectivo usuário responsável.

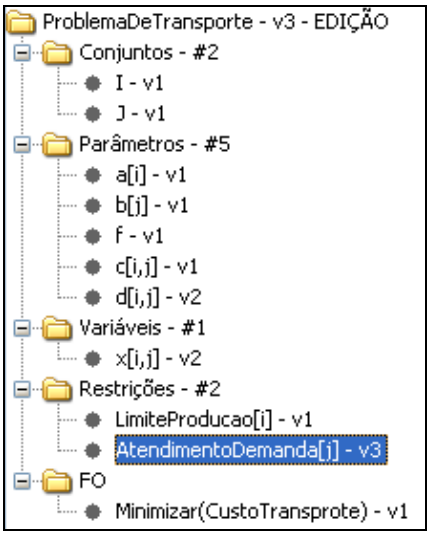

Figura 4. Versão dos elementos de modelagem

Pela característica de ser cliente-servidor e multiusuário, o sistema permite que mais de um usuário possa ver um projeto, desde que possua autorização para isso. Uma vez que alguém comece a editar a modelagem, o projeto inteiro fica bloqueado para a edição deste usuário. Assim que for terminada a modificação, o usuário pode liberar o projeto para que outro o edite, criando nova versão, ou para que essa versão do projeto seja utilizada, com criação dos cenários de dados pelos tomadores de decisão. Na Figura 5 pode ser visto o esquema de versionamento desenvolvido.

Quando a versão for liberada, ela não pode mais ser removida ou editada. Desta forma,

é garantindo que todo o ciclo de vida do modelo está registrado. Antes da liberação, o usuário que estiver editando a versão do projeto 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 nova versão.

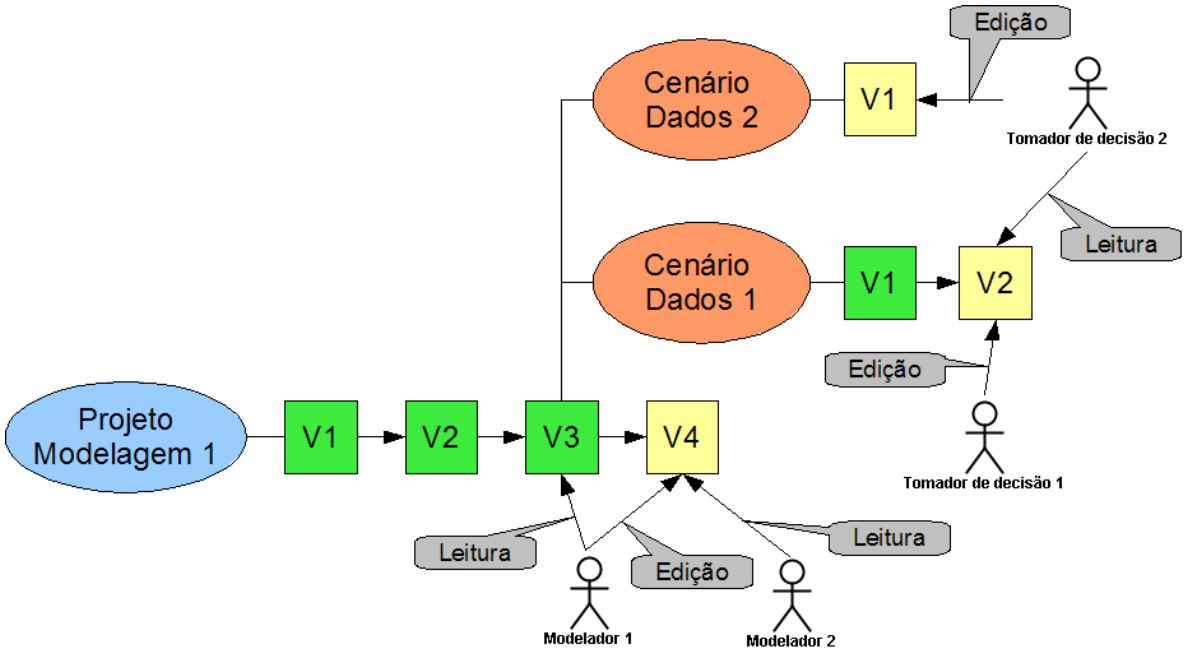

Figura 5. Versionamento do projeto de modelagem e dos dados associados

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. Cada cenário também possui as suas versões e o mesmo conceito de controle de modificações. Enquanto estiver em edição, outros usuários só podem visualizar o cenário. Quando o cenário for liberado, pode ser modificado com a criação de uma nova versão. Na Figura 5, também pode ser vista a associação dos cenários e suas versões com as versões dos projetos.

7. Conclusões

A abordagem descrita neste trabalho tem o objetivo de não apenas tratar a atividade de modelagem para problemas de programação linear e inteira, mas também auxiliar em todo o ciclo de vida dos modelos matemáticos. As principais diferenças desta proposta 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, através do controle de versão. Esta arquitetura também torna independente das máquinas dos usuários aquelas que executam os problemas de otimização e a instalação dos resolvedores, além de separar as informações dos modelos matemáticos dos dados do problema.

A base dessa abordagem é permitir a modelagem e o gerenciamento de modelos de programação linear e inteira através de um ambiente de software. Assim, foi desenvolvida uma ferramenta de modelagem que permita o desenvolvimento de modelos e o tratamento dos dados para solucionar problemas de diversas áreas do conhecimento. Tem-se por objetivo atender tanto às necessidades dos tomadores de decisão quanto dos modeladores.

A versão atual do ambiente de modelagem teve a parte que trata especificamente da modelagem matemática validada com a implementação de três modelos propostos por Melo, Urrutia e Ribeiro (2009). Para implementá-los foi necessário o uso de condições de geração para

variáveis e restrições, assim como a utilização de parâmetros calculados a partir de outros parâmetros, dada uma fórmula de cálculo. Um dos modelos implementados no ambiente trata variáveis com cinco índices, a ferramenta não possui restrição quanto à quantidade de índices de uma variável ou parâmetro.

O uso de controle de versão intrínseco ao sistema traz conceitos da Engenharia de Software para a aplicação de 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 um determinado modelo e a um cenário de dados.

Os trabalhos futuros para a evolução desta proposta são principalmente outras formas de modelagem, como o tratamento de problemas não-lineares. Por outro lado, a utilização de gerência de configuração pode ser aprimorada para contribuir com a evolução do controle de versão, permitindo, por exemplo, uma estratégia otimista, onde as versões dos projetos de modelagem e dos cenários de dados possam ser editadas em paralelo por mais de um usuário e mescladas ao final. Neste caso, são necessárias ferramentas para mesclar versões e identificar e resolver possíveis conflitos. Na abordagem atual, uma estratégia pessimista para controle de concorrência é utilizada, onde a edição de uma versão bloqueia todo o projeto para um único usuário.

Referências

Bisschop, J. e Meeraus, A. (1982), On the Development of a General Algebraic Modeling System in a Strategic Planning Environment, *Mathematical Programming Studies,* 20, 1-29.

Dantzig, G. B. (1963), Linear Programming and Extensions, *Mathematical Princeton University Press*.

Fourer, R., Gay, D. M., e Kernighan, B. W. (1990), A Modeling Language for Mathematical Programming, *Management Science,* 36, 519–554.

Fourer, R. (1997), Database structures for mathematical programming models, *Decision Support Systems*, 20, 317-344.

Fourer, R. (1998), Predictions for Web Technologies in Optimization, *INFORMS Journal on Computing*, 10, 388–389.

Fourer, R., Ma, J. (2010), Optimization Services: A Framework for Distributed Optimization, *Operational Research*, 58, No. 6, 1624-1636.

Geoffrion, A. M. (1987), Introduction to Structured Modeling, *Management Science,* 33, 5, 547-588.

Geoffrion, A. M. (1989), Computer-based modeling environments, *European Journal of Operational Research,* 41, 33-43.

GUROBI (2011), Gurobi Optimizer, referência on-line em http://www.gurobi.com/, visitada pela última vez em 06/05/2011.

Haverly, C. A. (2001), OMNI Model Management System, *Annals of Operations Research*, 104, 127–140.

IBM Corporation (1975), IBM Mathematical Programming Language Extended 370 (MPSX / 370), *Program Reference Manual*, SH19-1095.

IBM Corporation (2009), CPLEX, referência on-line em http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, visitada pela última vez em 06/05/2011.

Melo, R. A., Urrutia, S. e Ribeiro, C. C. (2009), The traveling tournament problem with predefined venues, *Journal of Scheduling,* 12, 6, 607-622.

Murphy, F. H., Stohr, E. A. e Asthana, A. (1992), Representation Schemes for Linear Programming Models, *Management Science - INFORMS*, 38, 964-991.

Pressman, Roger S., *Engenharia de software*, McGraw-Hill, São Paulo, 2006.