Atividades de Estágio
Agosto/2014 a Janeiro/2015

Uma Introdução ao IBM ILOG OPL

Estagiário

Gabriel Stefanini Vicente
Universidade Estadual de Campinas

Orientador

Prof. Dr. Lúcio Tunes dos Santos
Universidade Estadual de Campinas

Co-orientador

Me. Luís Franco de Campos Pinto
UniSoma Computação Ltda.

Campinas, 5 de Janeiro de 2015

Escrito com a distribuição TEXLive 2014

Agradecimentos

Eu expresso minha sincera gratidão ao professor Lúcio como orientador por ter tornado possível que as experiências fora da universidade componham minha avaliação acadêmica.

Eu também gostaria de agradecer o privilégio de fazer parte da pioneira em Pesquisa Operacional no Brasil, a UniSoma, e de estudar e de aprender ao lado de grandes analistas que formam seu singular corpo técnico.

Ao Volnei e à Bruna, o meu profundo agradecimento pelas linhas e linhas de códigos e horas e horas de idéias.

Resumo

A demanda por soluções integradas para problemas de tomada de decisão alavancaram soluções analíticas avançadas e novas abordagens para a otimização de decisão com o intuito de transformar dados em massa cada vez maiores em estratégias baseadas em fatos.
A última aposta da IBM para esse frutífero mercado é o *Decision Optimization Center* (DOC). Como produto, consiste em uma suíte de desenvolvimento aliada a uma linguagem de modelagem algébrica (AML), a *Otimization Programming Language* (OPL). A plataforma, porém, é insignificante sem o respaldo da Pesquisa Operacional. Desse modo, a modelagem matemática e o *deployment* são realizados por parceiras da corporação.
A UniSoma Computação Ltda. tem sido reconhecida com grande prestígio em seus projetos conjuntos, tendo recebido o prêmio em *Smarter Commerce*.
Este trabalho é fruto da oportunidade que a Matemática Aplicada proporciou ao estagiário de fazer parte dos últimos desenvolvimentos dentro da empresa e de elaborar um manual de referência (preliminar) para o novo paradigma.

Palavras-chave:

Modelagem Matemática, Programação Linear, CPLEX, OPL, IBM Decision Optimization Center

Conteúdo

Introdução

A complexidade e dimensionalidade dos problemas operacionais se impuseram a diversos campos ecônomicos, considerando o crescente acúmulo de informações e pluralidade de agentes. Em 2014, a Pesquisa Operacional foi escalada em segunda posição entre as mais promissoras carreiras pelo jornal U.S.News[1]. É nesse contexto que se torna evidente que as soluções mais desejadas são aquelas firmemente baseadas em conceitos e técnicas matemáticas e não meramente empíricas ou heurísticas. Desde o pós-guerra, a modelagem matemática e otimização têm se tornado cada vez mais comuns e confiáveis em incontáveis áreas de atuação, tais como as tradicionais manufaturas e processos logísticos até as mais inéditas como finanças e governos.

A gigante IBM já apostou nessa direção. Como uma das líderes globais em prover soluções, a corporação tem consolidado espaço em casos de sucesso e novos contratos e tecnologias. Desde a aquisição da ILOG Technlogies em 2009, a IBM tem promovido melhorias na plataforma a que se deu o nome de IBM *Decision Otimization Center*. Já foi conhecida por CPLEX Studio e IBM ODM. Trata-se de uma extensão do Eclipse, projeto já mantido pelo próprio IBM, para integrar ferramentas de modelagem matemática. Uma linguagem nascida com o propósito de facilitar esse processo surgiu durante esse esforço. Trata-se da *Optimization Programming Language* (OPL), que traz grandes vantagens sobre linguagens convenciais, além de fornecer bibliotecas e interconectividade. Juntamente com parceiras, foram fechados incontáveis projetos sobre a bandeira de *Smarter Commerce*. Um de seus casos de sucesso é uma brasileira, a UniSoma Computação Ltda.

Nessa sucinta obra, foram descritas as atividades de estágio na UniSoma e elaborado um preliminar manual para a linguagem OPL.

[1] `http://money.usnews.com/careers/best-jobs/operations-research-analyst`

1 Programação Linear

Programação Linear lida com problemas de maximizar ou minimizar uma função linear de várias variáveis sujeita a restrições de igualdade e/ou desigualdades lineares. Consideremos o Problema de Programação Linear na Forma Padrão (PPLFP):

$$\begin{gathered} \text{minimizar } b^T x \\ \text{sujeito a } Ax \geq c, \\ x \geq 0 \end{gathered}$$

onde $A \in \mathbb{R}^{m \times n}$ é uma matriz de posto completo m, x é o vetor das variáveis de decisão, b e c são vetores-coluna dos coeficientes. Associado a esse problema temos o problema dual (PPLFD):

$$\begin{gathered} \text{maximizar } c^T y \\ \text{sujeito a } A^T y \leq b, \\ y \geq 0 \end{gathered}$$

onde $A \in \mathbb{R}^{m \times n}$ é uma matriz de posto completo m, y é o vetor das variáveis duais, b e c são vetores-coluna dos coeficientes.

Esse gênero de problema deu origem à Pesquisa Operacional, um conjunto de métodos analíticos para melhorar a eficácia de operações. Durante a Segunda Guerra Mundial, testemunhou-se um gigantesco salto no campo por parte de estrategistas militares. Em seguida, organizações começaram a adotar as técnicas e modelos. Atualmente, analistas de Pesquisa Operacional estão presentes virtualmente em todas indústrias, de manufaturas e finanças a órgãos governamentais.

Alguns dos problemas mais tradicionais:

- Alocação
- Designação
- Dieta
- Mistura
- Produção
- Produção e Distribuição

2 UniSoma Computação Ltda.

O estagiário foi selecionado para integrar o corpo técnico da UniSoma Computação em Março de 2014. O primeiro tópico do programa de estágio esteve pautado pelo guia para iniciantes, carinhosamente apelidado de "Kit Bixo". Trata-se de uma apresentação dos aspectos fundamentais relacionados à empresa. Tópicos abrangem tanto procedimentos gerais de ordem mais prática quanto a visão e os valores da organização.

Desde então, teve a oportunidade de fazer parte ativa do desenvolvimento de dois projetos tanto nas tarefas de otimização quanto de tecnologia de informação. Também vale ressaltar que são projetos sobre plataformas distintas.

2.1 Histórico

Sendo a pioneira em Pesquisa Operacional no Brasil, a UniSoma Matemática para Produtividade S.A., agora UniSoma Computação Ltda., foi concebida em 1984 pelo Prof. Dr. Miguel Taube Netto, ex-professor do Instituto de Matemática, Estatística e Computação Científica (IMECC) da Universidade Estadual de Campinas (UNICAMP).

Uma de suas primeiras soluções foi o desenvolvimento e a implantação do Sistema de Planejamento Florestal de Longo Prazo (PLANFLOR). Baseado em um modelo de programação matemática e desenvolvido para mainframe, viabilizou a geração automática e otimizada de planos de manejo florestal de empresas como a CVRD e a Belgo Mineira. Ainda na década de 1980, a UniSoma concebeu o PLANCORTE, modelo para programação de cortes em bobinas de papel. Por meio de técnicas de otimição combinatória, o PLANCORTE propiciou reduções de cerca de 50% nas perdas devidas aos agendamentos de atendimento de carteira de pedidos. Alguns contratos de licenciamento do PLANCORTE, como com a Ripasa e o Grupo Simão, foram feitos sob o modelo de risco, pelos quais a remuneração da UniSoma resultou dos ganhos efetivamente percebidos por essas empresas. Não por acaso, os mais rentáveis para a pioneira.

Em 1989, a UniSoma deu início a uma parceria duradoura e de sucesso com a SADIA. O principal resultado deste esforço conjunto foi o Sistema de Planejamento Integrado da Produção Avícola. O PIPA consiste em um conjunto de módulos de planejamento das atividades relacionadas à produção de frangos de corte, dos níveis táticos aos operacionais. A ferramenta deu suporte à integração das diversas áreas que fazem parte da cadeia de fornecimento avícola: Fomento Agropecuário, Frigoríficocos, Logística e Vendas.

A SADIA contabilizou e atribuiu ao PIPA um benefício da ordem de 50 milhões de dólares no período de 1992 a 1994. O êxito na aplicação prática das técnicas matemáticas contidas no PIPA resultou em reconhecimento internacional: em 1995, a UniSoma recebeu o Franz Edelman Award for Management Science Achievement. O premiação é realizada por duas organizações norte-americanas: o INFORMS (Institute for Operations Research and the Management Science) e a CPMS (The College for the Practice of the Management Sciences). O Franz Edelman é concedido anualmente às organizações públicas e privadas que se destacam na implantação inovadora e economicamente significativa de técnicas de Gestão Científica e Pesquisa Operacional.

A UniSoma celebrou, em agosto de 2004, uma parceria de prestação de serviços com a Paragon Decision Technologies, empresa holandesa fornecedora do software AIMMS. O Advanced Integrated Multidimensional Modeling Software é uma linguagem de modelagem algébrica e uma ferramenta de desenvolvimento de sistemas de suporte à decisão e aplicações de planejamento avançado. É utilizado por várias empresas nas mais diversas áreas como gerenciamento da cadeia de fornecimento, planejamento de produção, logística, planejamento florestal e gerenciamento de risco, retorno e ativos. O AIMMS é integrado com vários solvers comerciais como, por exemplo, o CPLEX da ILOG Technologies.

Parceira da UniSoma desde 2003, a ILOG Technologies é uma das principais fornecedoras mundiais de ferramentas para o desenvolvimento de aplicações de planejamento

e programação baseadas em técnicas de programação matemática e de busca baseada em restrições. Destacam-se em seu portfolio de produtos o CPLEX (solver de alto desempenho para problemas de grande porte), o CP Optimizer e o ambiente de desenvolvimento OPL. No início de 2009 a ILOG foi adquirida pela IBM, uma das líderes globais nos mercados de *Service-Oriented Architecture* (SOA) e *Business Process Management* (BPM).

2.2 Projetos

O ritmo de desenvolvimento na UniSoma é frenético. É comum que cada analista esteja envolvimento ou na manutenção ou desenvolvimento de mais de um projeto. UniSoma adota o *Scrum* como processo, o que leva a entregáveis em dinâmica iterativa e incremental e à agilidade no gerenciamento do projeto. Os projetos dos quais o estagiário é desenvolvedor ainda estão sob cláusula de sigilo, portanto, serão referenciados por codinomes e terão apenas uma breve descrição.

2.2.1 SCP

O estagiário teve a singular oportunidade de envolvimento na manuteção de um projeto recém-terminado pela UniSoma. Foi o primeiro contato com modelagem matemática de um problema fisicamente tangível e complexo. Até então, na universidade, todos os aspectos para a resolução de exercícios já eram ou conhecidos ou facilmente manipuláveis. Essa tomada de consciência provocou um grande salto de aprendizado.

Projeto SCP traz a solução de um plano de gerenciamento de *Supply Chain*, na forma um modelo de programação matemática com robustez e de grande generalidade. Para a implementação, foram desenhados dois módulos em sintonia sobre a plataforma AIMMS. O estagiário teve a proveitosa chance de desenvolver manutenção evolutiva e se familiarizar progressivamente com o modelo de dados, das modelagens estatística e matemática. Também teve a chance de conhecer os conceitos relacionados a *Supply Chain*, como recursos, insumos, listas técnicas, processos e *lanes*.

Além do escopo concentrado modelagem matemática, foram realizadas tarefas relacionadas à computação. Essa abertura vem a contribuir com a ênfase em Matemática Computacional de sua graduação.

2.2.2 PPLAN

Projeto PPLAN inaugurou a parceria da UniSoma com a IBM e introduziu a OPL ao ambiente de desenvolvimento. A aplicação foi construída sobre a plataforma IBM DOC juntamente com personalizações de interface gráfica e integração de dados. Todo módulo adjunto é inteiramente na linguagem Java. O processamento propriamente dito ocorre em um servidor remoto de otimização licenciado pela IBM. O estagiário pôde promover inúmeras melhorias e novas implementações.

A UniSoma já conta com uma experiência de longa data com a otimização de produção e distriuição de *commodities* e problemas de *blending*. Nesse projeto, foi apresentada a maior solução para o primeiro. Trata-se de uma grande cooperativa agroindustrial, cujos principais insumos considerados são soja, café, trigo e outras mais *commodities*.

A otimização do planejamento da produção se deu em quatro diferentes linhas para horizontes diários, semanais e mensais. Foram consideradas incontáveis tabelas de dados de entrada que já estavam presentes nos processos internos do cliente. É uma particulari-

dade da UniSoma a singular atenção para respostas *tailor-made*, isto é, uma preocupação de que a implementação se adapte inteiramente aos processos da produção.

Desde a serialização das ordens de produção, pedidos transferências e exportações são decisões do modelo matemático. Devida à considerável dimensionalidade, tem se tornado um gigantesco PPLFP cujo principal desafio é a factibilidade. O IBM DOC permite uma grande flexibilidade no controle das restrições, de modo que ao usuário é aberta a possibilidade de ajustar parâmetros e de também comparar cenários *what-if*.

No último quartil, em fase final de projeto, a otimição passou a abranger os processos de exportação. O estagiário pôde participar na modelagem matemática e na implementação. Graças a generalidade do modelo, foi possível tratar as exportações como pedidos para *lanes* específicas e com certas características, como o controle de carga de navio e capacidade.

3 IBM ILOG OPL

Com aquisição da francesa ILOG em 2009, a célebre implementação CPLEX do algoritmo *Simplex* de Dantzig passou a fazer parte dos portfólios comerciais da IBM. A *Optimization Programming Language* (OPL) protagoniza o ambiente de desenvolvimento IBM *Decision Opmization Center*, baseado em *plug-ins* para o Eclipse[2]. A linguagem garante uma descrição natural da programação matemática para os modelos de otimização, com uma sintaxe avançada e compacta. Trata-se de uma linguagem de modelagem álgebrica (AML) que vem substituir linguagens *general-purpose*, tais como C, Java e Python. De qualquer modo, foi tomado o cuidado de fazê-la interagir com essas plataformas mais genéricas por via de bibliotecas.

Vale notar que, apesar de vantajosa, continua uma plataforma em construção. Para o caso da UniSoma, graças à parceria com a IBM, tem-se acesso a versões ainda não publicadas.

3.1 Definições

Nesse segmento, a linguagem OPL é apresentada sucintamente na prática. As definições têm por objetivo fornecer um manual de rápida consulta. Aspectos mais detalhados talvez exijam um olhar mais atento para a documentação oficial.

3.1.1 Modelo

Um modelo OPL consiste em:

- Declarações
- Instruções de pré-processamentos (não-obrigatório)
- Modelo/Problema
- Instruções de pós-processamentos (não-obrigatório)

3.1.2 Tipos e Estruturas

Tipos básicos são abstratamente implementados no OPL com precisão computacional arbitrária:

- Inteiros (*int*)
- Pontos flutuantes (*float*)
- Strings (*string*)

codes/tipos_estruturas.mod

```
1 int n = 42;

3 float f = 3.2;

5 {string} string = "Optimization Programming Language"
```

[2]Eclipse é um Ambiente Integrado de Desenvolvimento (IDE) mantido por um consórcio do qual a IBM faz parte.

Também estão disponíveis estruturas de dados e alguns métodos:

- Ranges (*range*)
- Vetores (*array*)
- Tuplas (*tuple*)
- Conjuntos (*set*)

codes/tipos_estruturas.mod

```
/* Range */
range I = 1..4;

/* Vetores */
int A[1..4] = [10, 20, 30, 40];
float F[1..4] = [1.2, 3.2, 3.4, 4.5];
string S[1..2] = ["IBM", "UniSoma"];

/* Vetores multidimensionais */
int M[1..2] [1..2] = [ [10, 20, 30], [40, 50, 60]];

/* Tuplas */
tuple Ponto {
  int x;
  int y;
};
```

Na tabela abaixo estão listadas operações usuais de conjuntos. Essa é uma das vantagens na prática do OPL em relação a outras linguagens. Vale ressaltar que o índice tem início em 0.

Tabela 1: Funções de Conjuntos

Operação	Resultado
union, inter, diff	elem = função(c1, c2)
firt, last	elem = função(c1)
next, prev	elem = função(c1, elem, int)
ord	int = função(c1)

Podem ser definidos conjuntos para quaisquer tipo definidos por tuplas, como no exemplo abaixo de um conjunto de pontos.

codes/tipos_estruturas.mod

```
/* Conjuntos */
{int} C = {4,3,2,1}

/* Ordenacao de conjuntos */
sorted {int} Org = {4,3,2,1};  \\ organiza em ordem "natural"
ordered {int} Ord = {1,2,3,4}; \\ ordena em ordem crescente

/* Conjunto de tipo definido */
{Ponto} Pontos = {<1,2>, <1,3>, <3,4>};
```

3.1.3 Dados

Até o momento, foram apresentadas as principais características que compõem um modelo de otimização usando OPL. Contudo, resta talvez o mais fundamental: dados. No Eclipse, estão disponíveis inúmeras interfaces para acoplar o modelo matemático ao dados e espelhar as entradas e saída em banco de dados. Contudo, foge do escopo desse manual.

Os dados podem ser inicializados:

- Internamente
- Externamente

Todos os exemplos anteriores foram inicializados internamente. Agora, vejamos como importar dados de um arquivo externo.

codes/dados.mod

```
/* Inicializacao externa */
{string} Cidades = ...;

float Custo[Cidades] = ...;

tuple Lane {
      string orig;
      string dest;
}
```

Para o arquivo *dados.dat*:

codes/dados.dat

```
Produtos = { "Campinas", "Sao Paulo", "Rio de Janeiro" };
Custo = [ 20, 40, 60 ];

Lanes = [ <"Campinas", "Sao Paulo">,
        <"Campinas", "Rio de Janeiro">,
        <"Sao Paulo", "Campinas">,
        <"Sao Paulo", "Rio de Janeiro">,
        <"Rio de Janeiro", "Campinas">,
        <"Rio de Janeiro", "Sao Paulo">];
```

3.1.4 Variáveis e Expressões de Decisão

O modelo matemática é propriamente definido através das variáveis. Vejamos como são tratadas. OPL permite dois tipos relacionais diretamente à programação matemática.

- Variáveis de Decisão (*dvar*)
- Expressões de Decisão Vetores (*dexp*)

As variáveis são a resposta do modelo. É de crucial importância ao modelar matematicamente atribuir domínios para para definir bem o problem quanto para garantir uma boa convergência.

codes/variaveis.mod

```
/* Dominio de variavel de decisao */
dvar int+ x;       // inteiros nao-negativos
dvar float+ y;     // reais nao-negativos
dvar boolean z;    // booleano
dvar int i in I;   // ranges

/* Dominio de variavel de decisao com tipos definidos */
tuple Lane {
     string orig;
     string dest;
};

{Lane} Lanes = ...;

dvar int Transp[Lanes] in 0..42;
```

As expressões são um recurso do OPL para aumentar a legibilidade de código. Com elas, é possível invocar expressões que envolvam várias variáveis e/ou parâmetros como uma outra variável. Esse processo é tratado eficientemente pelo OPL.

codes/variaveis.mod

```
/* Expressao de variaveis ou parametros */
dexpt int TranspTotal = sum(l in Lanes) Transp[l];

dvar float+ x[I];
dvar float+ y[I];


dexpr float diff[i in I] = x[i] - y[i];
```

3.1.5 Restrições

As restrições linerares têm papel crucial para o modelo. A sintaxe em OPL é puramente simplificada. As *constraints* podem ou não serem nomeadas. Esse recurso permite maior controle ao usuário ao rodar a aplicação. É possível, por exemplo, inspecionar quais restrições foram violadas e mesmo optar pelo relaxamento.

codes/restricoes.mod

```
subject to {
  forall(r in Recursos)
    ctCapacidade:
      sum(p in Produtos)
        Consumo[p][r] <= Capacidade[r];

  forall(p in Products)
    ctDemanda:
      Producao[p] >= Demanda[p];
}
```

3.2 Modelos-exemplo

Considera-se primeiramente o problema de planejamento de produção. O modelo tem por objetivo minimizar o custo de produção de um conjunto de produtos, enquanto

satisfizer a demanda. Para esse exemplo, suponha que cada produto possa ser produzido internamente ou externamente e que o custo externo seja maior que o custo interno. A produção interna está limitada pelos recursos da companhia, enquanto a externa será considerada ilimitada.

codes/producao.mod

```
{string} Produtos = ...;
{string} Recursos = ...;

float Consumo[Produtos][Recursos] = ...;
float Capacidade[Recursos] = ...;
float Demanda[Produtos] = ...;
float CustoInterno[Produtos] = ...;
float CustoExterno[Produtos]  = ...;

dvar float+ ProdInterna[Produtos];
dvar float+ ProdExterna[Produtos];

minimize
  sum(p in Produtos)
    (CustoInterno[p] * Interno[p] + CustoExterno[p] * Externo[p]);

subject to {
  forall(r in Recursos)
    ctCapacidade:
      sum(p in Produtos)
        Consumo[p][r] * ProdInterna[p] <= Capacidade[r];

  forall(p in Products)
    ctDemanda:
      ProdInterna[p] + ProdExterna[p] >= Demanda[p];
}
```

Os dados consistem em conjuntos de produtos e recursos e parâmetros associados. Para esse caso, de produção de três linhas de alimentos, têm-se o consumo e capacidade de cada recurso e os custos interno e externo para cada produto. As variáves de decisão são as quantidades a produzir interna e externamente de cada produto.

codes/producao.dat

```
Produtos =  { "spaghetti", "capellini", "fettucine" };
Recursos = { "trigo", "ovos" };

Consumo = [ [0.5, 0.2], [0.4, 0.4], [0.3, 0.6] ];
Capacidade = [ 20, 40 ];
Demanda = [ 100, 200, 300 ];
CustoInterno = [ 0.6, 0.8, 0.3 ];
CustoExterno  = [ 0.8, 0.9, 0.4 ];
```

codes/producao.mod

```
Solucao Final com objetivo de 372.0000:
  ProdInterna = [40.0000 0.0000 0.0000];
  ProdExterna = [60.0000 200.0000 300.0000];
```

Em outro caso, teremos o clássico exemplo de um problema de *Blending* modelo como um PPLFP. Trata-se da mistura de ingredientes para obter um produto com certas características e propriedades.

codes/blending.mod

```
{string} Gasolinas = ...;
{string} Oleos = ...;

tuple GasolinaTipo {
  float demanda;
  float custo;
  float octanagem;
  float chumbo;
}

tuple OleTipo {
  float capacidade;
  float custo;
  float octanagem;
  float chumbo;
}

TipoGasolina Gasolinaolina[Gasolinas] = ...;
TipoOle Oleo[Oleos] = ...;
float MaxProducao = ...;
float CustoPRoducao = ...;

/* Variaveis */
dvar float+ A[Gasolinas];
dvar float+ Blend[Oleos][Gasolinas];

/* Modelo */
maximize
  sum( g in Gasollinas , o in Oleos )
    (Gasolina[g].custo - Oil[o].custo - CustoProducao) * Blend[o][g]
      - sum(g in Gasolinas) A[g];

subject to {

  forall( g in Gasolinas )
    ctDemanda:
      sum( o in Oleos )
        Blend[o][g] == Gasolina[g].demanda + 10*A[g];

  forall( o in Oleos )
    ctCapacidade:
      sum( g in Gasolinas )
        Blend[o][g] <= Oleo[o].capacidade;

  ctMaxProducao:
    sum( o in Oleos , g in Gasolinas )
      Blend[o][g] <= MaxProducao;

  forall( g in Gasolinas )
    ctOctanagem:
      sum( o in Oleos )
        (Oil[o].octanagem - Gasolina[g].octanagem) * Blend[o][g] >= 0;

  forall( g in Gasolinas )
    ctChumbo:
      sum( o in Oleos )
        (Oil[o].chumbo - Gasolina[g].chumbo) * Blend[o][g] <= 0;
```

```
58 }
```

codes/blending.dat

```
Gasolinas = { "Super", "Normal", "Diesel" };
2 Oleos = { "Tipo1", "Tipo2", "Tipo3" };

4 Gasolina = [ <3000, 70, 10, 1>,
          <2000, 60, 8, 2>,
6         <1000, 50, 6, 1> ];

8 Oleo = [ <5000, 45, 12, 0.5>,
          <5000, 35, 6,  2>,
10        <5000, 25, 8,  3> ];


12

MaxProducao = 14000;
14 CustoProducao = 4;
```

codes/blending.mod

```
Solucao final com objetivo de 287750.0000:
2   Blend = [[2088.8889 2111.1111 800.0000]
             [777.7778 4222.2222 0.0000]
4            [133.3333 3166.6667 200.0000]];
  A = [0.0000 750.0000 0.0000];
```

Comentários Finais

Apenas durante esses últimos meses de UniSoma que pude consolidar e compreender os conceitos da Matemática Aplicada que, dentro de contexto, se tornaram essencialmente simples. A experiência de um ambiente de pesquisa fora da universidade de muito contribuiu para reformular minhas convicções. Fazer parte de uma dinâmica frenética de desenvolvimento e ter tido a oportunidade de trabalhar com tecnologias e plataformas das mais recentes tem sido um grandioso privilégio.

Esse manual preliminar de OPL cumpre a função antes de ser uma referência própria. Futuramente, é possível que se abra para contribuições e melhorias para colaboradores da UniSoma e de interessados.

Referências

[1] Paragon Decision Technology. Advanced Interactive Multidimensional Modeling System. http://www.aimms.com, 1993–2014.

[2] IBM. Decision Optimization Center. http://www-03.ibm.com/software/products/pt/decision-optimization-center, 1988–2014.