UTILIZAÇÃO DE SOFTWARES NA PESQUISA OPERACIONAL

USING SOFTWARE IN OPERATIONAL RESEARCH

Denise Helena Lombardo Ferreira, Orientadora

Carolina Baron, carolina.b2@puccamp.edu.br

Pontifícia Universidade Católica de Campinas - PUC-Campinas
Campinas, São Paulo

Submetido em 19/05/2016
Revisado em 20/05/2016
Aprovado em 31/10/2016

Resumo: A Pesquisa Operacional tem auxiliado no desenvolvimento de diversos modelos e algoritmos de otimização para a resolução de problemas nas mais variadas áreas. Tendo em vista os diversos *softwares* atualmente existentes para aplicação da Pesquisa Operacional, a presente pesquisa apresenta a manipulação de dois softwares de linguagem algébrica – AIMMS e GAMS, para um determinado problema, denominado de problema do Incinerador.

Palavras chave: Software. AIMMS. GAMS. Pesquisa Operacional.

Abstract: Operational Research has helped in the development of various models and optimization algorithms for solving problems in various areas. In view of the many currently existing software for the application of Operations Research, this research presents the manipulation of two algebraic language software - AIMMS and GAMS, to a certain problem, named The Incinerator Problem.
Keywords: Software. AIMMS. GAMS. Operational Research

Introdução

Nos últimos anos, a Pesquisa Operacional tem auxiliado no desenvolvimento de diversos modelos e algoritmos de otimização para a resolução de problemas nas mais variadas áreas: saúde, logística, telecomunicações, finanças, dentre outras. Para Horgren (1978) a Pesquisa Operacional trata da aplicação de métodos científicos para auxiliar na tomada de decisão. Ou seja, a Pesquisa Operacional representa o mundo real através de modelos matemáticos e, utilizando-se de métodos quantitativos, os resolve com o objetivo de otimizá-los. Goldbarg e Luna (2000) acrescentam que a Pesquisa Operacional pode ser utilizada para alocar eficientemente os recursos limitados e que podem ser disputados por atividades alternativas. Tendo em vista sua importância no auxílio da tomada de decisão, é comum as empresas fazerem uso dessa ciência, sobretudo nos dias de hoje em que os recursos são cada vez mais escassos, gerando grande competitividade econômica. Nessa linha, Murty (2003) assinala que modelos inteligentes são essenciais para se ter bons resultados.

Em 1939, o termo Pesquisa Operacional (ou "*Operations Research"*) foi empregado pela primeira vez, na tentativa de englobar diversas técnicas. Entretanto, somente a partir da Segunda Guerra Mundial é que a Pesquisa Operacional começou a ser aplicada, especialmente no tratamento de problemas militares (LACHTERMACHER, 2007).

Após a Segunda Guerra Mundial, a aplicação da Pesquisa Operacional foi iniciada nas indústrias para a programação da produção, controle de estoques, programação de vendas, problemas de transportes, manutenção e substituição de equipamentos, estudos de mercado, planejamento de atividades, investimentos, problemas de localização, dentre outros.

De uma maneira geral, a Pesquisa Operacional está apoiada em quatro ciências fundamentais: econômica, matemática, estatística e informática. As técnicas mais utilizadas são: Programação Linear, Programação Inteira, Programação Não Linear, Programação Dinâmica, Teoria dos Estoques, Teoria das Filas, Simulação, Teoria dos Jogos, Teoria dos Grafos, Planejamento com PERT/CPM.

Lachtermacher (2007) destaca que antes do aprimoramento da tecnologia muitos gerentes faziam uso apenas da intuição para tomar decisões, porém a quantidade de informações nos últimos anos cresceu exponencialmente, dificultando o tratamento dessas informações de forma intuitiva.

Tendo em vista os diversos *softwares* atualmente existentes para aplicação da Pesquisa Operacional, a presente pesquisa busca elucidar quatro deles, a saber, Microsoft Excel, AIMMS, GAMS, LINGO, com destaque para apenas dois que apresentam linguagem algébrica – AIMMS e GAMS. O problema em estudo neste artigo é denominado de problema de incinerador.

Softwares

Solver – Suplemento do Excel

O solver (suplemento do Excel) é uma ferramenta existente no Microsoft Office Excel que resolve problemas de Programação Linear, Não Linear e Inteira de pequeno porte. O solver utiliza o algoritmo Simplex para determinar a solução ótima de um modelo de Programação Linear. Para problemas de Programação Inteira, utiliza-se o algoritmo LP Simplex. Já para problemas não lineares, o solver utiliza o algoritmo GRG2 (*Generalize Reduced Gradient*).

AIMMS

O *software* AIMMS (*Advanced Integrated Multidimensional Modeling Software*) é uma linguagem algébrica que resolve problemas de Programação Quadrática[1], Linear, Não Linear e Inteira de alta complexidade. Utiliza pacotes de otimização como BARON, CPLEX[2], MINOS, XPRESS, entre outros. Possui interface com as linguagens de programação C++, Java, NET, Excel. O AIMMS se mostra favorável frente à outras opções (OPS Studio, Dot Net Solver Foundation), por conta de facilidades com integração, análises de sensibilidade, interface gráfica.

[1]Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382004000100011.

[2]A sigla CPLEX é a combinação da letra C, em referência à linguagem de programação C utilizada no desenvolvimento deste algoritmo, com a terminação PLEX, em referência ao algoritmo simplex de solução de problemas de PL. Este solver foi desenvolvido pela CPLEX Optimization Inc., empresa fundada em 1988 com a ideia de comercializar algoritmos de PL que pudessem ser utilizados para solucionar, de forma rápida, problemas grandes e difíceis de programação linear. Atualmente o CPLEX é um produto de propriedade da ILOG S.A.

O AIMMS foi introduzido em 1993, como uma ferramenta que auxilia a modelagem matemática. Essa ferramenta tem sido utilizada por empresas em diversas áreas: suprimentos, planejamento de produção, logística, gestão de risco, bem como em diversas universidades, tanto nas disciplinas como nas pesquisas. Dentre as empresas que utilizam o AIMMS, pode-se citar a Bayer, ExxonMobil, Heineken, Merck, Perdigão, Petrobras, Philips, Shell, Unilever.

O AIMMS possui uma versão gratuita para *download*, limitada em 200 identificadores. Para sua utilização há uma licença a qual está disponível no site: http://main.aimms.com/aimms/overview/.

GAMS

O *software* GAMS (*General Algebraic Modeling System*) é um sistema de modelagem algébrica utilizado para a solução de problemas complexos, envolvendo Programação Linear, Não Linear e Inteira. Essa ferramenta possui interface que interage com diversos pacotes de otimização, tais como: CPLEX, MINOS, XPRESS, LINDO, entre outros.

O GAMS foi originalmente desenvolvido por um grupo de economistas do *World Bank* com o objetivo de facilitar a resolução de problemas matemáticos complexos por meio de um computador. Para sua utilização há uma versão gratuita no site: http://www.feg.unesp.br/~fmarins/GAMS/apostilagams.pdf.

LINGO

O LINGO (*Language for Intective General Optimizer*) é um *software* de modelagem e resolução de problemas lineares e não-lineares de otimização da empresa LINDO Systems. Sua versão completa não apresenta limitações quanto ao número de restrições, variáveis reais e inteiras. Os solvers disponíveis são: Não Linear Geral, Global, Multistart, Barrier, Simplex e Inteira Mista. O LINGO possui uma versão gratuita disponível no site http://www.lindo.com/.

Utilizando os *softwares* AIMMS e GAMS

Nesta seção é apresentada a definição de um problema da área de Pesquisa Operacional, bem como a sua formulação matemática e por fim a sua

transferência para as linguagens algébricas AIMMS e GAMS com sua respectiva solução.

O problema em estudo, denominado problema do incinerador, consiste em minimizar o custo total com a coleta e destino do lixo de duas cidades (MENDES; FERREIRA, 2015), problema adaptado de Salles Neto (2015). A cidade 1 produz 500 toneladas de lixo por dia, a cidade 2 produz 400 toneladas e a cidade 3 produz 450 toneladas. O lixo deve ser incinerado em três incineradores, 1, 2 e 3, e cada incinerador pode processar até 500 toneladas de lixo por dia. O custo por incineração do lixo é de R$40/ton no incinerador 1, R$30/ton no 2 e R$35/ton no 3. A incineração reduz cada tonelada de lixo a 0,2 toneladas de resíduos que devem ser armazenadas em três aterros. Cada aterro pode receber até 200 toneladas de resíduos por dia. Para transportar uma tonelada de material, lixo ou resíduo, há um custo de R$3,00 por quilômetro. As distâncias entre as cidades e os incineradores e entre os incineradores e os aterros são mostradas na Tabela 1.

Tabela 01: Distâncias (km) entre cidade e incinerador e entre incinerador e aterro.

	Incinerador 1	Incinerador 2	Incinerador 3
Cidade 1	30	5	22
Cidade 2	36	42	34
Cidade 3	28	18	30
Aterro 1	5	9	4
Aterro 2	8	6	11
Aterro 3	9	10	7

A formulação matemática para esse problema é apresentada abaixo.

Sejam:

x_{ijk} = quantidade (toneladas) de lixo transportada da cidade i para o incinerador j e para o aterro k;

c_j = custo do incinerador j;

d_{ijk} = distância em km da cidade i para o incinerador j e para o aterro k (Tabela 1);

$prod_i$ = quantidade do lixo produzida pela cidade i;

cin_j = capacidade de recepção do lixo pelo incinerador j;

cat_k = capacidade de recepção do lixo pelo aterro k.

A função objetivo min {custo de incineração + custo de transporte} é definida pela equação (1):

$$\min \{\sum_{j=1}^{3} c_j (\sum_{i=1}^{3}\sum_{k=1}^{3} x_{ijk}) + 3\sum_{k=1}^{3}\sum_{j=1}^{3}\sum_{i=1}^{3} d_{ijk} x_{ijk}\} \qquad (1)$$

As restrições a serem atendidas são referentes a quantidade de produção de lixo das cidades: equação (2), capacidade dos incineradores: equação (3) e a capacidade dos aterros sanitários: equação (3).

Cidades:

$$\sum_{j=1}^{3}\sum_{k=1}^{3} x_{ijk} = prod_i \quad sendo\ i = 1,2,3;\ prod_i = (500,400,450) \qquad (2)$$

Incineradores:

$$\sum_{i=1}^{3}\sum_{k=1}^{3} x_{ijk} \leq cin_j \quad j = 1,2,3;\ \ cin_j = (500,500,500) \qquad (3)$$

Aterros:

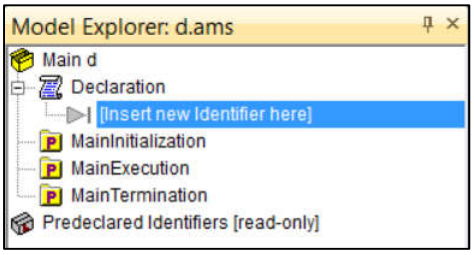
$$0,2\sum_{i=1}^{3}\sum_{j=1}^{3} x_{ijk} \leq cat_k \quad k = 1,2,3;\ \ cat_k = (200,200,200) \qquad (4)$$

Resolução usando o *software* AIMMS

A construção do modelo é feita ao abrir o programa AIMMS, selecionar a opção "*Create a New Project*" e inserir um nome para que o modelo seja salvo e em seguida, escrever o modelo.

Na janela principal do programa, encontra-se a estrutura inicial do modelo como mostrado na Figura 1.

Figura 01: Estrutura inicial do modelo.



A estrutura apresentada na Figura 1 contém uma seção de declarações (*declaration*), onde devem ser inseridos os elementos básicos do modelo matemático em estudo e três procedimentos pré-definidos: *MainInitialization, MainExecution, MainTermination*. No problema do incinerador, somente é utilizada a seção *MainExecution* e *declaration*.

O primeiro passo para iniciar a construção do modelo é definir os conjuntos (*set*). Para isso, é preciso selecionar o ícone ⊞, ao lado de *declaration*. Em seguida, deve-se adicionar *sets, parameters, variables,* e *constraints.* Ambos os ícones ficam na barra de ferramentas (Figura 2) na parte superior da tela, identificados por S, P, V e C, respectivamente.

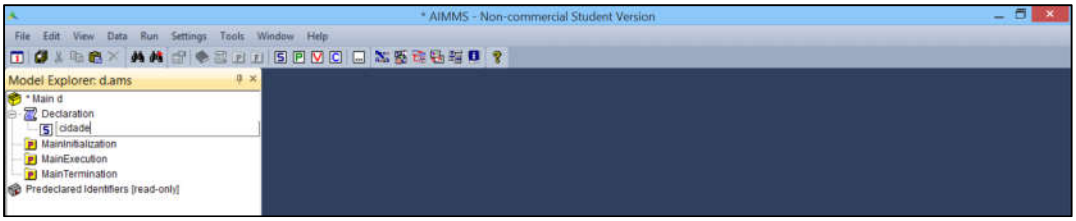
Figura 02: Barra de ferramentas do AIMMS.

- Sets

No problema em estudo, os "*sets*" (conjuntos) são: 'cidade', 'incinerador' e aterro'.

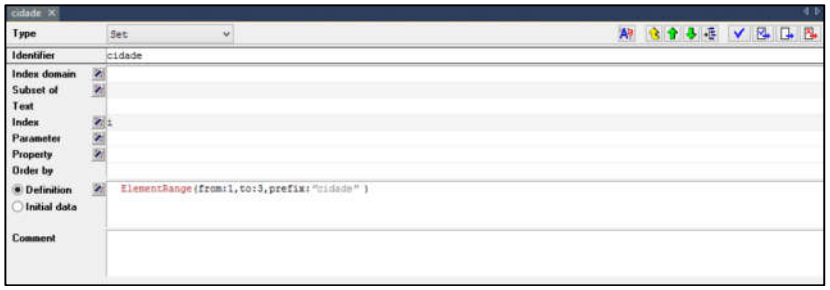
Para adicioná-los é preciso selecionar o ícone [S] (S) e determinar o nome de identificação do conjunto. A Figura 3 mostra o conjunto 'cidade'.

Figura 03: Inserção do conjunto 'cidade'.



Dois cliques devem ser dados no *set* ‘cidades’ e uma nova tela é aberta (Figura 4).

Figura 04: Conjunto ‘cidade’.



Nesta tela é preciso identificar o índice na caixa que está escrito index e identificar a quantidade de cidades participantes através da função *ElementRange*. Para finalizar este conjunto deve-se selecionar a opção *Check, Commit and Close* ( ).

O mesmo procedimento deve ser repetido para os outros dois conjuntos ‘incinerador’ e ‘aterro’.

- Parameters

A adição dos parâmetros necessários ao modelo é similar à inserção dos conjuntos.

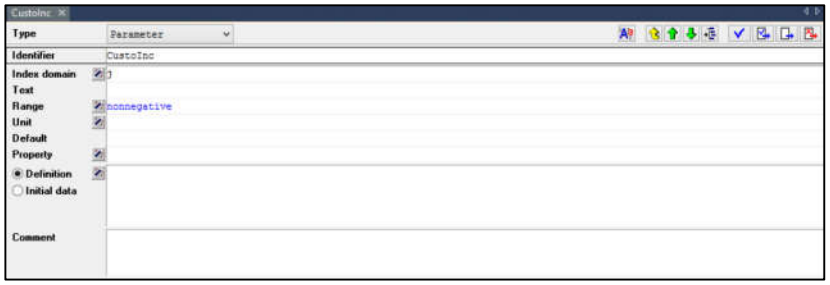
Os parâmetros para o problema em estudo são: ‘CustoInc’, ‘QlixoCid’, ‘CapRecepInc’, ‘CapRecepAter’, ‘Dist_1’, ‘Dist_2’ e ‘Dist_Total’.

Para a adição desses parâmetros basta selecionar a opção (P), definir o nome de identificação e clicar duas vezes para abrir uma nova janela. A seguir é mostrado o exemplo para a adição do parâmetro ‘CustoInc’ (Figura 5).

Nesta janela deve-se identificar o índice referente ao parâmetro (*index domain*) e definir se esse parâmetro é não-negativo, não-positivo, livre, binário ou inteiro (*range*). O parâmetro pode ter uma dimensão ou mais.

Para fechar a janela, deve-se selecionar a opção *Check, Commit and Close* ( ) e repetir estes passos para os demais parâmetros.

Figura 05 – Parâmetro ‘CustoInc’.



- Variable

Para declarar uma variável é preciso selecionar a opção (V), definir o nome e clicar duas vezes para adicionar suas características. No problema em estudo, as variáveis são: ‘X’ e ‘Fobj’.

A variável X indica a quantidade de lixo que é produzido pela cidade *i*, incinerado pelo incinerador *j* e destinado ao aterro *k* a fim de minimizar o custo total (Figura 6).

Figura 06: Variável ‘X’.

Para a caracterização das variáveis, basta determinar *index domain* e *range*, conforme já mencionado anteriormente para os parâmetros.

Para fechar a janela, basta clicar em *Check, Commit and Close* () e repetir estes passos para a variável ‘Fobj’, a qual contém a função objetivo descrita como equação (Figura 7).

Figura 07: Variável ‘Fobj’.

- Constraint

Para declarar uma restrição é preciso selecionar a opção (C), definir o nome e clicar duas vezes para adicionar suas características.

No problema em estudo, as restrições são: ‘Qlixo’, ‘QInc’ e ‘QAter’.

A Figura 8 destaca o exemplo da restrição ‘Qlixo’.

Figura 08: Restrição ‘Qlixo’.

Nesta tela é preciso identificar o índice na caixa que está escrito *index domain* e definir a equação dessa restrição no campo *definition*.

Para finalizar e concluir essa janela, basta selecionar a opção *Check, Commit and Close* ().

O mesmo deve ser feito para as demais restrições.

- Mathematical Program

No programa matemático é definida a função objetivo do problema.

Para declarar um programa matemático, é preciso pressionar , localizado na barra de ferramentas, escolher o item *Mathematical Program*, , e selecionar a opção ‘ok’. Em seguida, deve-se especificar o nome do programa matemático e preencher suas características (Figura 9).

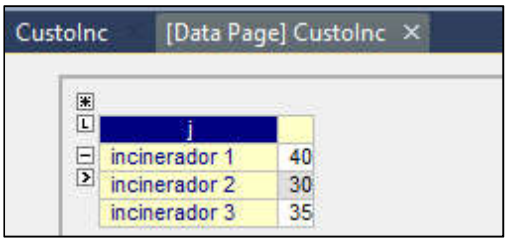
Figura 09: Programa Matemático ‘Decisao’.

Para a declaração do programa matemático ‘Decisao’, precisa-se determinar o objetivo, que nesse caso é Fobj (é a variável que traz o resultado final da função objetivo), a direção da função (se é para maximizar ou minimizar a função), as variáveis e as restrições que participam da decisão, e o tipo do modelo (linear, não-linear, inteiro, misto etc.).

Após a construção do modelo, deve-se inserir os dados do problema. Estes são inseridos na *data page* dos parâmetros.

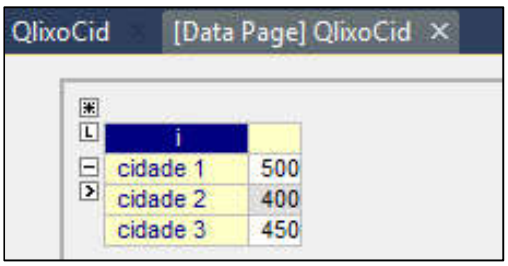
Para abrir a janela de inserção de dados, deve-se clicar com o botão direito em cima do parâmetro que deseja e selecionar *data page*. Uma tela é aberta, conforme o exemplo do parâmetro ‘CustoInc’ (Figura 10).

Figura 10: Data page ‘CustoInc’.



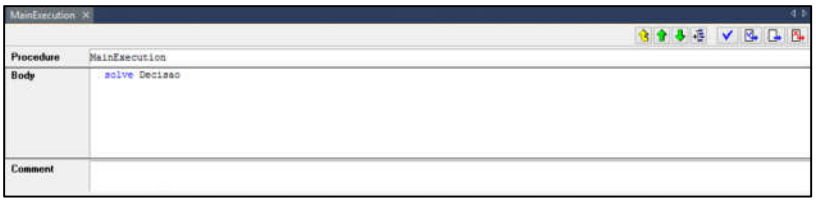
Nesta janela, deve-se colocar os valores fornecidos no enunciado do problema, referentes aos custos de cada incinerador (Figura 11). O mesmo deve ser feito para os demais parâmetros.

Figura 11: Data page ‘QlixoCid’.



Após a inserção dos dados, é preciso executar o programa. Para isso, utiliza-se o campo *MainExecution,* um procedimento previamente definido. Para utilizá-lo, basta completar o formulário. Para isso, deve-se clicar duas vezes em *MainExecution*. Uma nova janela é aberta (Figura 12). Nesta janela é preciso escrever em *body* o que se deseja otimizar no problema.

Figura 12: Características do *MainExecution*.



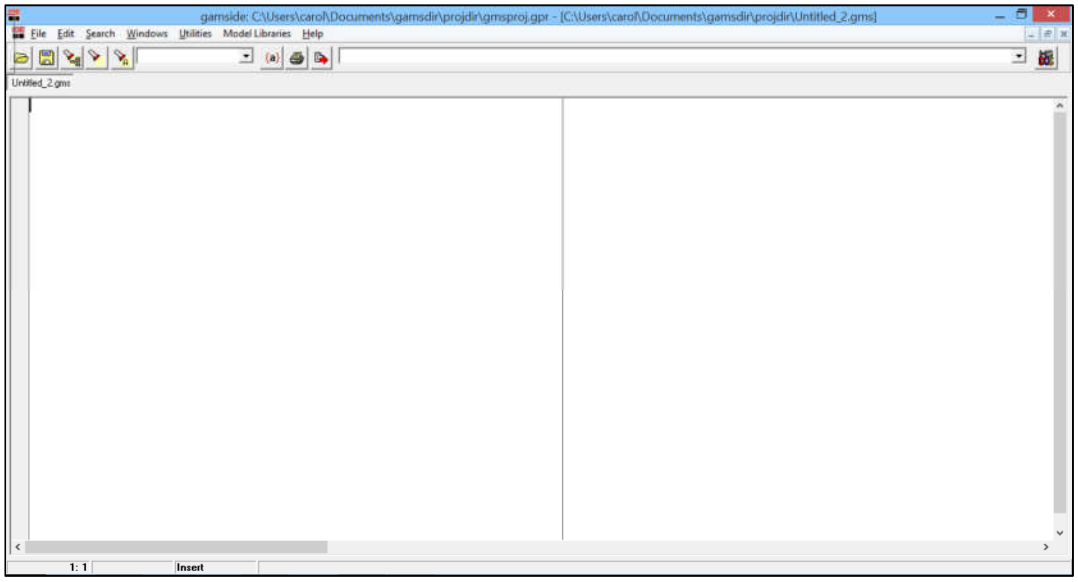
Em seguida é preciso executar este procedimento. Para isso, deve-se clicar com o botão direito em *MainExecution* e selecionar *Run Procedure*.

Os resultados obtidos estão localizados no *data page* das variáveis ‘X’ e ‘Fobj’.

Resolução usando o *software* GAMS

A construção do modelo é feita ao abrir o programa GAMS, selecionar a opção “*File*” e em seguida a opção *“New*”. Uma janela em branco é aberta (Figura 13). É nesta única janela que o problema será modelado.

Figura 13: Janela principal do GAMS.



- Sets

Para iniciar a modelagem, é necessário definir o *set* (conjunto). Este bloco consiste em definir uma série de conjuntos que correspondem aos índices na representação algébrica do modelo (Figura 14).

Figura 14 – Sets.

```
Sets
    i cidades / 1, 2, 3 /
    j incinedaror / 1, 2, 3 /
    k aterro / 1, 2, 3 / ;
```

No exemplo em estudo há três índices: *i, j, k*, que representam cidades, incineradores e aterros, respectivamente. Em verde encontram-se os membros de cada conjunto. Neste caso, são utilizados como membros os números 1, 2, e 3, entretanto tais identificações poderiam ser feitas por caracteres.

- Parameters/Tables

O próximo passo é declarar os dados do problema em *parameters*. Esse bloco consiste em guardar os dados e associá-los aos conjuntos (Figura 15).

Para o problema em estudo são definidos os seguintes parâmetros: custo do incinerador *j*, quantidade de lixo produzido da cidade *i*, capacidade de

recepção do lixo pelo incinerador *j*, capacidade de recepção de lixo pelo aterro *k* e distância da cidade-incinerador-aterro.

Figura 15: Parameters.

```
Parameters
   c(j) custo do incinerador j
   / 1 40
     2 30
     3 35 /

   pro(i) quantidade de lixo produzido da cidade i
   / 1 500
     2 400
     3 450 /

   cin(j) capacidade de recepção do lixo pelo incinerador j
   / 1 500
     2 500
     3 500 /

   cat(k) capacidade de recepção de lixo pelo aterro k
   / 1 200
     2 200
     3 200 /
```

O parâmetro distância da cidade-incinerador-aterro não contém dados diretos. Portanto, sua composição é feita através de uma equação que utiliza-se de outros dados (Figura 16). Esse parâmetro utiliza outro bloco de dados chamado *table* para construir sua equação. *Table* é um bloco o qual trabalha com dados de duas ou mais dimensões.

Figura 16: Parâmetro distância cidade-incinerador-aterro.

```
Parameter
dist(i,j,k) distancia da cidade-incinerador-aterro ;
          dist(i,j,k) = dis1(i,j) + dis2(j,k);
```

Como o parâmetro dist(i,j,k) faz uso duas *tables*, é preciso declará-las anteriormente a esse parâmetro (Figura 17)

Figura 17: Tables.

```
Table dis1(i,j) distancia da cidade até incinerador (km)
      1    2    3
 1   30    5   22
 2   36   42   34
 3   28   18   30 ;

Table dis2(j,k) distancia do incinerador até o aterro (km)
      1    2    3
 1    5    8    9
 2    9    6   10
 3    4   11    7 ;
```

Em seguida, é preciso declarar *scalar* que é um bloco que trabalha com constantes, ou seja, que não estão associados à nenhum conjunto. O exemplo abaixo destaca o valor em reais para transportar uma tonelada por km rodado (Figura 18).

Figura 18: Scalar.

```
Scalar v valor em reais para transportar uma tonelada por km rodado /3/ ;
```

- Variables

Esse bloco é obrigatório o qual traz os resultados do problema. Cada variável tem um nome e um domínio apropriado (Figura 19).

Figura 19: Variables.

```
Variables
    z função objetivo
    x(i,j,k) quantidade (toneladas) de lixo transportada da cidade i para o incinerador j e para o aterro k;
```

A variável ‘z’ referente a função objetivo não possui domínio, pois é escalar. Todo problema de otimização precisa de uma variável para representar a quantidade a ser maximizada ou minimizada. Além disso, deve-se especificar o tipo: livre, positiva, negativa, binária e inteira (Figura 20).

Figura 20: Positive variable.

```
Positive Variable x ;
```

- Equations

As equações devem ser declaradas e definidas dentro do bloco *equations*. Primeiramente é preciso declarar a equação, com seu respectivo nome, índice e função (Figura 21).

Figura 21: Declarando as equações.

```
Equations
    fobj define a função objetivo
    qntlixo(i) quantidade limite de lixo produzido na cidade i
    qntinc(j) quantidade de lixo que passa pelo incinerador j
    qntater (k) quantidade de lixo que o aterro k recebe ;
```

Em seguida, deve-se definir as equações (restrições). Há dois tipos de equações, a função objetivo (não possui domínio) e as restrições (possuem domínio) (Figura 22).

Figura 22: Definindo as equações.

```
fobj .. z =e= sum((i,j,k), c(j)*x(i,j,k)) + 3 * sum((i,j,k), dist(i,j,k) * x(i,j,k));

qntlixo(i) .. sum(j, sum(k, x(i,j,k))) =e= pro(i);

qntinc (j) .. sum(i, sum(k, x(i,j,k))) =l= cin(j);

qntater (k) .. 0.2 * sum(i, sum(j, x(i,j,k))) =l= cat(k);
```

Os operadores relacionais utilizados nas restrições possuem os seguintes significados:

- =l= menor ou igual a
- =g= maior ou igual a
- =e= igual a

- Model and solve statements

A palavra *model* possui um significado preciso e é simplesmente uma coleção de equações. A estrutura do modelo é seguida pelo nome do modelo e pela lista de equações entre barras. Se todas as equações previamente definidas forem utilizadas para a resolução do problema, basta escrever o nome do modelo e /all/, na frente (Figura 23).

Figura 23: Model.

```
Model transport /all/ ;
```

O *software* possibilita selecionar as equações a serem utilizadas no modelo, como por exemplo: "*Model transport* /qntlixo, qntinc/".

Uma vez que o modelo foi declarado e suas equações atribuídas, é preciso declarar o solver (Figura 24).

Figura 24: Solver.

```
Solve transport using lp minimizing z ;
```

O solver é descrito pela palavra solve, o nome do modelo a ser resolvido, procedimento para a solução (lp, nlp, qcp, ...), palavra-chave *maximizing* ou *minimizing* e o nome da variável a ser otimizada.

- Display statements

Esta função pode gerar várias especificações do resultado, assim que o programa for executado. Para se obter os valores ótimos da variável dual e primal, pode-se chamar pelo *display* (Figura 25).

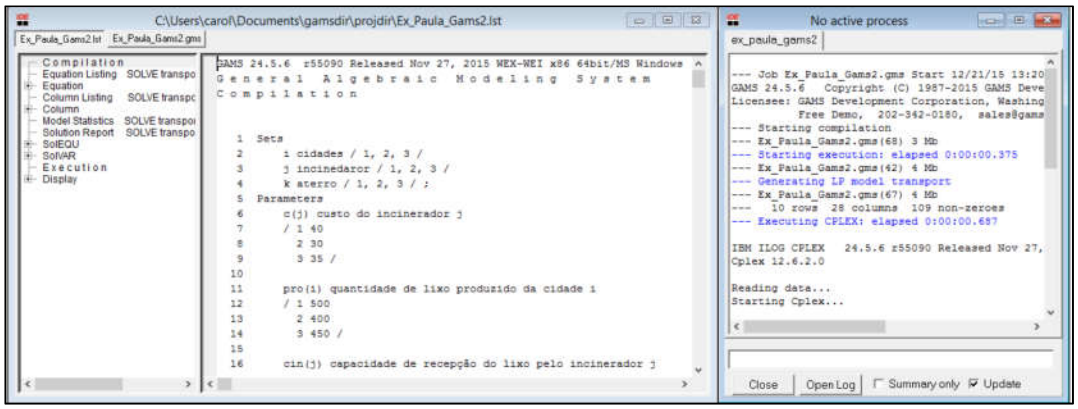
Figura 25: *Display.*

```
Display x.l, x.m ;
```

Para executar o programa, basta apertar a tecla F9, ou selecionar o ícone  , na barra de ferramentas.

Os resultados são mostrados em uma nova janela (Figura 26).

Figura 26: Resultado do problema do incinerador pelo GAMS.



A solução ótima encontrada na aplicação pelos *softwares* AIMMS e GAMS obteve um custo total de R$153.450,00 com 350 toneladas de lixo produzidas pela cidade 3 que devem passar pelo incinerador 1 e chegar ao aterro 1; 500 toneladas de lixo produzidas pela cidade 1 que devem passar pelo incinerador 2 e chegar ao aterro 2; 400 toneladas de lixo produzidas pela cidade 2 que devem passar pelo incinerador 3 e chegar ao aterro 1 e 100 toneladas de lixo produzidas pela cidade 3 que devem passar pelo incinerador 3 e chegar ao aterro 1.

Considerações finais

O objetivo principal deste trabalho foi estudar e analisar diferentes *softwares* que resolvem problemas de Pesquisa Operacional. Inicialmente, fez-se uma revisão bibliográfica sobre estes *softwares* para identificar sua metodologia. Devido a falta de conhecimento a respeito de alguns detalhes essenciais para evitar erros de compilação, algumas dificuldades surgiram na construção dos problemas.

A presente pesquisa contribuiu não apenas para o conhecimento dos *softwares*, mas também no entendimento de suas aplicações e limitações.

Além do problema do Incinerador apresentado neste artigo, outros problemas foram estudados e estruturados através dos *softwares* descritos com a finalidade de aprimorar o entendimento da área em questão.

Entende-se que a contribuição desse estudo ocorre ao disponibilizar às demais pessoas, uma base para o entendimento da Pesquisa Operacional bem como o conhecimento de alguns de seus *softwares* disponíveis no mercado.

Referências

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e programação linear:** modelos e algoritmos. Rio de Janeiro: Campus, 2000.

HORGREN, C. T. **Contabilidade de Custos: Um Enfoque Administrativo**. São Paulo: Atlas, 1978.

LACHTERMACHER, G. **Pesquisa Operacional na tomada de decisões**. Rio de Janeiro: Campus, 2007.

MENDES, P. V.; FERREIRA, D. H. L. Uma aplicação da Programação Linear. **Anais do 15º Congresso Nacional de Iniciação Científica CONIC-SEMESP**, 201z5. Disponível em: <http://conic-semesp.org.br/anais/files/2015/trabalho-1000019444.pdf>. Acesso em: 31 mar 2016.

MURTY, K. G. **Optimization Models for Decision Making**, 2003. Disponível em: <http://ioe.engin.umich.edu/people/fac/books/murty/opti_model/junior-0.pdf**>.** Acesso em: 03 fev. 2015.

SALLES NETO, L. L. **Tópicos de Pesquisa Operacional para o Ensino Médio.** Disponível em: <http://www.mat.ufg.br/bienal/2006/mini/leduino.pdf>. Acesso em: 31 jul. 2015.