COMPARAÇÃO DE DESEMPENHO E USABILIDADE ENTRE OS SOFTWARES COMERCIAIS DE OTIMIZAÇÃO E O MÉTODO DUAL PARA O PROBLEMA CLÁSSICO DO TRANSPORTE

CAROLINA MEDUNA BAZIEWICZ – UFPR – carol.meduna@gmail.com
ALEXANDRE DE JESUS FANTE – UFPR – fanteaex@gmail.com
CASSIUS TADEU SCARPIN – UFPR – cassiusts@gmail.com
ARINEI CARLOS LINDBECK DA SILVA – UFPR – arineicls@gmail.com
GUSTAVO VALENTIM LOCH – UFPR – gustavo.gvalentim@gmail.com

RESUMO:
Este trabalho tem o intuito de comparar desempenho, eficácia e usabilidade entre os *softwares* de otimização *IBM ILOG CPLEX 12.4*®, *LINGO* 12® e da programação do Método *Dual*, o qual é baseado no Método *Stepping-Stone*, em linguagem *Visual Basic.Net*, para a resolução do problema Clássico de Transporte. Foram geradas dezesseis instâncias aleatórias, variando quantidade de pontos de origens, pontos de destinos, demandas e ofertas, para que se pudesse realizar a análise comparativa proposta. A análise consiste em relação ao desempenho, avaliar o tempo computacional gasto em cada forma de solução, em relação à eficácia, obter ou não a solução ótima (caso não seja ótima, quanto é o desvio da função objetivo), e finalmente em relação à usabilidade, avaliar-se a interface com o usuário, a dificuldade de aprendizagem e a linguagem de programação necessária. Obteve-se como conclusão que para problemas de maiores dimensões, o *software CPLEX* apresentou os melhores índices de desempenho. A programação do Método *Dual* também obteve resultados válidos e bastante satisfatórios principalmente para problemas pequenos, mas também seu uso se justifica para problemas maiores. Assim, verificou-se que a escolha da melhor forma de resolução depende das necessidades do usuário, do conhecimento técnico de programação e das características do problema.

Palavras-chave: Problema Clássico de Transporte, *CPLEX 12*.4®, *LINGO12*®, Método *Stepping-Stone*

ABSTRACT:
This work aims to compare performance, effectiveness and usability among the optimization software *IBM ILOG CPLEX 12*.4®, *LINGO 12* ® and the programming of the Dual Method, which is based on the Stepping-Stone Method, written in Visual Basic.Net language, for solving the Classic Transportation Problem. Sixteen random instances were generated. It was varied an amount of origin points, destination points, demand and supply, so that subsequently the comparative analysis proposal could be performed. The analysis consists of in relating to performance, evaluate the computational time spent on each solution type; regarding efficacy, getting the optimal solution (if it is not possible, obtaining the deviation of the best objective function value); and finally in relation to usability, evaluate the user interface, the difficulty of apprenticeship and the programming language required. It was obtained as a conclusion that for larger problems, the *CPLEX* software showed the best performance indices. The programming of the Method Dual also achieved significant results, mainly for small problems, but its use is also justified for larger problems. Therefore, it was found that the choice of the best means of resolution depends on the user's requirements, of technical knowledge and programming characteristics of the problem.

Key-words: The Transportation Problem, *CPLEX 12.4*®, *LINGO12*®, Stepping-Stone Method

1. INTRODUÇÃO

Para que as empresas sejam competitivas no mercado, a busca pela redução de custos é contínua. Na maioria das empresas, os processos de distribuição física de produtos costumam absorver cerca de sessenta por cento do gasto logístico total (BALLOU, 2001). O objetivo de um modelo linear referente ao problema clássico do transporte é determinar a solução na qual os custos estejam minimizados e que as expectativas de entregas dos produtos acabados dos fornecedores para os consumidores sejam atendidas.

Há diferentes abordagens para resolver um modelo linear envolvendo transporte. Alguns *softwares* conhecidos hoje são o *CPLEX*® e *LINGO*®, sendo que ambos buscam a solução ótima do modelo, atribuindo os melhores valores para cada variável do problema. Outra forma de se resolver é através do Canto Noroeste, para obtenção da solução inicial, e posteriormente aplicar o Método *Dual*, baseado no método *Stepping-Stone* (PIZZOLATO, GANDOLPHO, 2009), visando chegar à solução ótima.

No geral, o tempo para resolução de problemas reais que possa ser formulado matematicamente como problema linear está se tornando cada dia menor, para problemas de qualquer tamanho, devido à utilização de *softwares* como *LINGO*, *CPLEX*, *Xpress* e outros (SILVA, 2012).

No que é referente aos *softwares*, o *CPLEX*® e o *LINGO*® são solucionadores de alto desempenho matemático para a programação linear, programação inteira mista, programação quadrática e problemas de programação quadrática com restrições. São executáveis que podem ler um arquivo de forma interativa ou arquivo com determinado padrão permitindo diferentes maneiras de entrada dos dados. A entrada de dados é através da modelagem linear do problema clássico de transportes, sendo que cada *software* possui uma linguagem específica para tal. Essa formatação será melhor apresentada no decorrer deste trabalho.

A linguagem computacional utilizada para a geração das instâncias e programação da solução inicial pelo método do canto noroeste e sua otimização foi o *Visual Basic.NET*® (VB.Net). Além disso, na mesma linguagem, foram gerados arquivos com a formatação necessária para a utilização dos softwares citados. Verificou-se que o tempo computacional gasto para obtenção da solução varia de acordo com o tamanho do problema, a grandeza dos dados de demanda e oferta e, claro, o meio de solução executado. Esse tempo varia, também, conforme o número de pontos de oferta, pontos de demanda e o próprio balanceamento do transporte. Segundo Lopes, Gomes e Montané (2013) a escolha do problema influencia no número de variáveis, restrições e, assim, no tempo de execução e resolução do problema, que costuma ser dispendioso.

Neste trabalho a proposta é analisar e comparar os tempos gerados pelos *CPLEX*®, *LINGO*® e Método Canto Noroeste / *Dual-Stepping-Stone* referente ao tempo de manipulação de dados e otimização, assim como a usabilidade de cada opção de solução. Para tal, os mesmos modelos lineares serão resolvidos pelas três formas citadas, sendo que os modelos gerados estão separados em grupos de diferentes dimensões de demanda e oferta.

Para tanto o artigo está divido em 5 seções: a seção 2 descreve formalmente o problema clássico de transporte, enquanto a seção 3 apresenta as ferramentas computacionais e metodologias utilizadas, e a seção 4 apresenta os resultados testados sobre as instâncias aleatórias geradas e a análise comparativa. Finalmente na seção 5 apresenta-se as conclusões do artigo e sugestões para trabalhos futuros.

2. PROBLEMA CLÁSSICO DE TRANSPORTE

O Problema Clássico de Transportes, formulado por Hitchcock em 1941 (PIZZOLATO, GANDOLPHO, 2009), está dentre os modelos determinísticos da Pesquisa Operacional, e é resolvido através de algoritmos de Otimização Linear. De acordo com Murty (1983), um problema qualquer de programação linear pode ser modelado com a intenção de reproduzir o seu funcionamento ou determinar uma estrutura ideal. São três os elementos principais do modelo: as variáveis de decisão que são incógnitas a serem determinadas e os

parâmetros que são os valores fixos do problema; as restrições que o limitam; e a função objetivo, que é a função matemática que determina a qualidade da solução em função das variáveis de decisão.

O Problema de Transporte pode ser entendido como o transporte de determinadas mercadorias de pontos de *origens* (por exemplo, centro de distribuição) para *destinos* (por exemplo, mercados consumidores), no qual se deve obter a quantidade ótima a ser transportada de cada origem para cada destino. Seja *m* um conjunto de origens, *n* um conjunto de destinos e $C_{(mxn)}$ a matriz que associa os custos c_{ij} de transporte de se enviar produtos da origem *i* para o destino *j*. A cada ponto de origem está associada uma capacidade a_i e a cada ponto de destino está associada um demanda b_j. Define-se x_{ij} a variável de decisão a ser determinada, ou seja, qual a quantidade a ser transportada de *i* até *j*.

Para que a solução seja factível, devem-se satisfazer as seguintes restrições:

- A quantidade total de produtos enviada por uma origem não excede a sua capacidade;
- A quantidade total de produtos recebida por um destino deve satisfazer sua demanda;

Segundo Murty (1983), o problema de transportes pode ser modelado com um problema de programação linear (1) - (7), dados os parâmetros apresentados anteriormente:

$$\text{Minimizar } Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1)$$

$$\text{Sujeito à: } \sum_{j=1}^{n} x_{ij} \le a_i \qquad 1 \le i \le m, \qquad (2)$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j \qquad 1 \le j \le n, \qquad (3)$$

$$a_i > 0 \quad ,\forall i \qquad (4)$$

$$b_j > 0 \quad ,\forall j \qquad (5)$$

$$x_{ij} > 0 \quad \textit{para todo } i \textit{ e todo } j \qquad (6)$$

A Função Objetivo (1) estabelece a minimização do custo total do transporte. A restrição (2) refere-se à capacidade de fornecimento de uma origem *i*. A restrição (3) refere-se à demanda de cada destino j. As restrições (4), (5) e (6) especificam que as variáveis de decisão e os parâmetros devem ser não negativos. Para ser considerado equilibrado, o modelo atende à $\sum a_i = \sum b_j$, isto é, a capacidade de fornecimento de todas as origens é igual à soma absoluta de todas as demandas.

O modelo linear sugere que o problema a ser resolvido esteja balanceado. O balanceamento do modelo de transporte é evidenciado quando a soma de toda oferta é igual à soma de toda demanda. Em caso de desigualdade entre esses valores, um ponto de oferta ou um ponto de demanda é criado. Quando a soma de ofertas é superior à soma de demandas, por exemplo, um "*consumidor fictício j*" é criado, isto é, adiciona-se um ponto de demanda ao modelo. Os custos de transporte c_{ij} relacionados ao envio de mercadorias para esse destino são considerados zero, não exercendo qualquer influência sobre a função objetivo, e o valor absoluto da demanda para o mesmo é a diferença, em módulo, existente entre a soma de ofertas e demandas. Quando ocorre o inverso, a soma das demandas é superior à soma das ofertas, cria-se um *"fornecedor fictício i"*, também com custos associados valendo zero e com a importância da diferença, em módulo, entre a soma de ofertas e demandas.

Ao se gerar instâncias aleatórias os modelos podem não ser necessariamente balanceados. Faz-se então, antes da resolução, o balanceamento dos problemas gerados. O

equilíbrio foi programado em linguagem VB.Net e serviu para deixar o modelo no formato correto para a execução dos softwares *CPLEX*® e *LINGO*® e para o método canto noroeste, o qual, trata-se de uma solução inicial gerada para o problema de transporte. A solução inicial obtida através desse método geralmente está distante do ótimo, porém é possível aplicar o método *Dual / Stepping-Stone,* visando atingir a solução ótima.

O problema clássico de transportes é um problema que já é de antemão bastante estudado e desenvolvido. Destaca-se, então, que objetivo desse trabalho é evidenciar algumas das diferentes formas de uso dos softwares escolhidos, para o problema em questão. Em seguida, as instâncias aleatórias serão correlacionadas à utilização de cada forma de obtenção da solução. A próxima seção discutirá as três ferramentas utilizadas.

3. FERRAMENTAS COMPUTACIONAIS UTILIZADAS

3.1. CPLEX *OPTIMIZER*®

Segundo a *IBM*®, o *IBM ILOG CPLEX Optimizer* é considerado um *solver* de programações matemáticas de alto desempenho, podendo resolver problemas reais com milhões de restrições e variáveis. Ele abrange as áreas de programação linear, bem como programação com variáveis inteiras, programações quadráticas e problemas de programações quadráticas ilimitadas.

Segundo Moré e Wright (1994), o *CPLEX Optimizer* é um componente do *IBM ILOG CPLEX Optimization Studio*. O *Optimizer* tem um ambiente de modelagem chamado *Concert*, que possui interface para programação nas linguagens C++, C# e Java. Também possui uma interface de linguagem *Python* baseada em interface em C, e conectores para o *Microsoft Excel e MATLAB*. O CPLEX *Optmizer* ainda pode ser acessado através de sistemas de modelagem independentes, como AIMMS, AMPL, GAMS, MPL, OpenOpt, OptimJ e Tomlab, ou ser utilizado através de sua própria linguagem, o OPL (*Optimization Programming Language*).

3.1.1. Metodologia Utilizada

Para utilização desse software na resolução do problema de transportes, utilizou-se o *Prompt* de Comando do *Windows*, através da inicialização do programa e subsequentes execuções de comandos que o *CPLEX* possui.

Primeiramente, é necessário ter um arquivo de texto que o *CPLEX* seja capaz de reconhecer e ler, contendo todas as informações necessárias do problema: a função objetivo, os valores dos custos de transporte, as demandas e as ofertas. O modelo de texto em extensão *LP* utilizado para escrever o problema está relacionado na Figura 1, como segue:

Minimize (8)

$c_{11}\ x_{11} + \ c_{12}\ x_{12} + \ \ldots \ + c_{ij}\ x_{ij}$ (9)

Subject to (10)

$x_{i1} + \ x_{i2} + \ \ldots \ + x_{in} \leq \ a_i \quad 1 < i < m$ (11)

$x_{1j} + \ x_{2j} + \ \ldots \ + x_{mj} \geq \ b_j \quad 1 < j < n$ (12)

Generals (13)

$x_{11} \ \ x_{12} \ \ \ldots \ \ x_{ij} \quad \forall i, j$ (14)

End (15)

FIGURA 1 - MODELO DE ARQUIVO DE EXPORTAÇÃO CPLEX
Fonte: Os Autores (2013)

Nesse modelo, (8) - "*Minimize*" define a orientação do problema, que no caso do problema de transporte significa minimização. Ao invés de "*Minimize*", o usuário poderia utilizar "*Maximize*", "*Minimum*" ou "*Maximum*", caso necessário. Em (9), descreve-se a função objetivo do problema. O coeficiente c_{ij} deve estar separado por um "Espaço" do valor

x_{ij}, e um sinal (+ ou -) deve dividir dois termos $c_{ij}\ x_{ij}$. Todos os termos podem ser escritos em uma só linha, sem sinais de pontuação. Em (10), "*Subject to*" inicia a leitura das restrições do problema. Em (11), as restrições referentes à oferta de cada origem devem ser escritas com um "Espaço" entre um sinal de pontuação e um valor x_{ij}, seguindo-se de sinal de menor ou igual (<=) como descrito acima, seguido ainda do valor da oferta da respectiva origem. A restrição de cada origem deve ser escrita em uma linha (basta separar duas restrições através de um *Enter*).

Em (12), o mesmo procedimento deve ser adotado para as restrições de oferta, dessa vez com o sinal de igual (=), cada restrição em uma linha. Não é necessário incluir nenhum sinal de pontuação ao final de cada linha. Em (13), "*Generals*" significa a definição de uma variável de decisão inteira positiva do problema. Outras opções possíveis são "*Binary*", para definir variáveis binárias e "*Semi-Continous*" para variáveis canalizadas. Em (14), descrevemos cada uma das variáveis relacionadas separando-as por em "Espaço". Para indicar que o problema foi completamente escrito, escrevemos em (15) "*End*".

A maneira como utilizamos o *CPLEX* consiste basicamente em executar o *Prompt* de Comando através da classe *Process* do VB.Net, uma classe que permite disparar , controlar e monitorar aplicações internas e externas. Para tanto, cria-se um arquivo na extensão *.BAT*, que é uma linha de comando que vai executar um comando no *MS-DOS*. Quando acionamos o *MS-DOS* via *Process*, ele executa automaticamente o que está descrito no arquivo *.BAT*. Utilizamos o arquivo.*BAT* para executar o comando "*cplex < Script.txt*". Esse comando, por sua vez, associará os comandos descritos no arquivo *Script.txt* ao *CPLEX*. O modelo do arquivo "*Script.txt*" segue na Figura 2:

read "Arquivo.lp"	*(16)*
optimize	*(17)*
write "Arquivo.sol"	*(18)*
quit	*(19)*

FIGURA 2 - MODELO DE ARQUIVO SCRIPT
Fonte: Os Autores (2013)

Em execução, esse *Script* faz com que o arquivo salvo em extensão *LP* seja lido em (16); que a solução ótima seja encontrada em (17); que um arquivo contendo a solução completa do programa seja exportado para leitura em (18), e que o programa seja encerrado em (19), retornando à interface do programa desenvolvido.

O código apresentado na Figura 3 representa o botão do programa em VB.Net que cria todos os arquivos necessários e resolve os modelos.

```
ChDir("c:\vbtestes")
Dim Nome As String
Dim posicao1 As Integer
 posicao1 = InStr(Arquivo, ".")
Nome = Mid(Arquivo, 1, posicao1 - 1
FileClose(1)
FileOpen(1,"c:\vbtestes\Script.txt", OpenMode.Output)
 PrintLine(1, "read " & Arquivo)
PrintLine(1, "optimize")
PrintLine(1, "write " & Nome & ".sol")
Print(1, "quit")
 FileClose(1)
FileOpen(1, "c:\vbtestes\executa.bat", OpenMode.Output)
PrintLine(1, "del " & Nome & ".sol")
Print(1, "cplex < Script.txt")
FileClose(1)
```

```
Dim p As New Process()
Dim l As New Stopwatch

p.StartInfo.FileName = "executa.bat"
p.StartInfo.Arguments = ""
 l.Start()
p.Start()
 do while not p.HasExited
Loop
 l.Stop()
MsgBox("resolvido")
Form2.TextBox2.Text =
l.ElapsedMilliseconds.ToString
 p.Dispose()
```

FIGURA 3 - CÓDIGO PROGRAMADO PARA GERAÇÃO DOS ARQUIVOS
Fonte: Os Autores (2013)

Ao término desse processo, obteve-se e exportou-se a solução ótima em um arquivo de texto (.SOL), que posteriormente poderá ser lida pelo programa desenvolvido ou diretamente do diretório salvo, através do *Bloco de Notas*. Pelo código, é possível perceber que a ferramenta *Stopwatch* tem a finalidade de mensurar o tempo total do processamento da solução, que totaliza o tempo de abertura do *MS-DOS*, *CPLEX*, leitura do problema, resolução e salvamento.

3.2. *LINGO*®

Segundo Junior e Souza (2004), o *LINGO* é um *software* de modelagem e resolução de problemas lineares e não lineares, que pode ter a leitura dos dados feita a partir do próprio *LINGO*, de um arquivo de *Bloco de Notas* ou de uma Planilha do *Excel*.

O *LINGO* é uma ferramenta da *LINDO SYSTEMS* projetada para tornar a construção e a resolução de problemas lineares, não lineares, quadráticos, quadráticos restritos, estocásticos e inteiros mais rápida e intuitiva para o usuário. Com o *LINGO*, o tempo de desenvolvimento do problema tende a ser inferior quando comparado a outros softwares como o *CPLEX*, pois a sua interface é de fácil utilização e não necessita um elevado nível de conhecimento de programação. É principalmente por essas características que o *LINGO* pode ser escolhido dentre alguns *softwares*, ainda que ele possa ser superado em termos de desempenho.

3.2.1. Metodologia Utilizada

De maneira similar à metodologia empregada para o *CPLEX Optimizer*, para utilização do *LINGO*, utilizou-se um processo que executa comandos automaticamente, fazendo com que o problema seja lido pelo *software* de resolução por meio de um arquivo de texto, resolva-o, e imprima a solução em outro arquivo em formato de texto.

O *LINGO* não oferece a mesma possibilidade que o *CPLEX*® possui de utilizar o *MS-DOS* para executar os comandos e economizar tempo de processamento referente à inicialização do programa. É necessário executar o processo para abrir o *LINGO12*® e resolver de dentro do próprio software. Primeiramente, de maneira análoga, é necessário ter um arquivo de texto que o *LINGO*® seja capaz de reconhecer e ler. O modelo de texto em extensão *LNG* utilizado para escrever o problema está relacionado abaixo, como segue na Figura 4.

Model:	*(20)*
$MIN = c_{11} * x_{11} + c_{12} * x_{12} + ... + c_{ij} * x_{ij};$	*(21)*
$x_{i1} + x_{i2} + ... + x_{in} \leq a_i; \quad 1 < i < m$	*(22)*
$x_{1j} + x_{2j} + ... + x_{mj} \geq b_j; \quad 1 < j < n$	*(23)*
$@Gin(x_{ij}); \quad \forall i, j$	*(24)*
End	*(25)*
set terseo 0	*(26)*
go	*(27)*
divert "Arquivo.sol"	*(28)*
solu	*(29)*
rvrt	*(30)*
quit	*(31)*

FIGURA 4 - MODELO DE ARQUIVO DE EXPORTAÇÃO LINGO
Fonte: Os Autores (2013)

O modelo deverá ser inicializado por "*Model*:", em (20); Em (21), define que a função objetivo deve ser iniciada com a palavra "*MIN* =" ou "*MAX* =". A função objetivo deve ser escrita tal como no modelo, com os espaços e sinais de mais intercalando os termos $c_{ij} * x_{ij}$, que possuem um sinal de multiplicação (*). É possível que para problemas muito grandes, a interface do *LINGO* não aceite a Função Objetivo (FO) inteira por falta de espaço. Nesse

caso, basta continuar escrevendo a FO na linha abaixo, e somente ao término de toda a função objetivo coloca-se no final da linha um ponto-e-vírgula (;). Sem a necessidade de informar ao programa que as restrições serão inseridas, basta escrever uma restrição por linha, todas finalizadas por um ponto-e-vírgula(;). Esse procedimento deve ser adotado para todas as restrições existentes no problema.

Em (24), faz-se a definição de variáveis do tipo inteiras: para cada variável, escreve-se "*@gin(x_{ij});*" Existem as opções "*@bin(x_{ij})*" para variáveis binárias, "*@free(x_{ij})*" para variáveis que possam assumir qualquer valor real, entre outros. É importante salientar que cada variável deve ser escrita em uma linha e que todas devem conter o ponto-e-vírgula. Para indicar que o problema foi completamente escrito, escrevemos na sequência "*End*", como em (25). De (26) à (31), estão dispostos alguns comandos internos do *LINGO*. Em síntese, *"set terseo 0*" significa desabilitar a exibição das soluções pelo programa do *LINGO* que não a criação do arquivo *.SOL*; "*go*" significa solicitar a otimização do problema; "*divert*" significa criar um novo arquivo no qual será escrito a solução; "*solu*" significa escrever a solução no arquivo gerado; "*rvrt*" significa fechar este arquivo de solução gerado; e por fim "*quit*" que significa fechar o *LINGO*.

Para utilização do *LINGO*, o procedimento adotado foi, de maneira análoga ao do *CPLEX*, executar um processo (através da classe *Process*()). Aqui, o processo consiste basicamente em inicializar o *LINGO*, fazendo com que ele leia o arquivo no formato *LNG* fornecido. Os comandos existentes no final do arquivo se encarregam de resolver o problema e escrever a solução em arquivo na extensão *SOL*. O código apresentado na Figura 5 representa o botão do programa em VB.Net que executa o processo e resolve os modelos:

```
ChDir("c:\vbtestes")
Arquivo = Replace(Arquivo.ToString, ".sol", ".lng")
Dim p As New Process()
Dim l As New Stopwatch

p.StartInfo.FileName = "c:\lingo12\lingo12.exe"
p.StartInfo.Arguments = " -t" & Arquivo
l.Start()
p.Start()
Do While Not p.HasExited
Loop
l.Stop()
MsgBox("resolvido")
Form2.TextBox2.Text = l.ElapsedMilliseconds.ToString
p.Dispose()
```

FIGURA 5 - CÓDIGO PROGRAMADO PARA A RESOLUÇÃO
Fonte: Os Autores (2013)

Da mesma maneira, ao término desse processo, obteve-se e exportou-se a solução ótima em um arquivo de texto (.SOL), que posteriormente poderá ser lida pelo programa desenvolvido ou diretamente do diretório salvo, através do *Bloco de Notas*. Pelo código, é possível perceber que a ferramenta *Stopwatch* tem a finalidade de mensurar o tempo total do processamento da solução, que totaliza o tempo de abertura do *LINGO12*®, leitura do problema, resolução e salvamento de arquivo.

3.3. PROGRAMAÇÃO DO MÉTODO DUAL – *STEPPING-STONE*

3.3.1 Canto Noroeste

Uma solução básica viável (SBV) deve conter um total de $n + m - 1$ variáveis básicas (Murty, 1983). A alocação dessas variáveis para uma solução inicial deve ser feita de acordo com algum critério. (HILLIER, LIEBERMAN, 2012). O método começa considerando a

célula que está ao canto noroeste do quadro, $Célula_{ij}$ sendo que os valores atribuídos para *i* e *j* serão iguais a um. A variável corresponde a x_{ij} e será igual ao mínimo entre Capacidade F_i e Demanda D_j [$X_{ij} = min(F_i;D_j)$] O método, baseado em Coutinho (2011), é descrito a seguir:

$D_j > F_i$, significa que a capacidade esta esgotada, porém as necessidades do cliente ainda não foram totalmente atendidas. Dessa forma, $D_j = D_j - F_i$; $F_i = 0$, isto é, a oferta ainda disponível é toda designada para o destino, diminuindo a demanda ainda não atendida e zerando a quantidade a ser ofertada. Atribui-se a variável *i* mais um (*i*+1), deslocando-se para a próxima linha e reduzindo o quadro com a exclusão da capacidade F_i desde que $i < m$.

$D_j < F_i$, significa que as necessidades do cliente podem ser atendidas, porém haverá uma ociosidade referente à capacidade. Dessa forma, $D_j = 0$; $F_i = F_i - D_j$, isto é, a oferta ainda disponível não será toda designada para o destino, diminuindo a quantidade ofertada e zerando a quantidade da demanda ainda não atendida. Atribui-se a variável *j* mais um (*j*+1), deslocando-se para a próxima coluna e reduzindo o quadro com a exclusão da demanda D_j desde que $j < n$.

$D_j = F_i$, significa que as necessidades do cliente podem ser atendidas e não haverá ociosidade quanto à capacidade. Dessa forma, $D_j = F_i = 0$, isto é, a oferta ainda disponível é toda designada para o destino, zerando as quantidades ainda disponíveis de oferta e demanda. Atribui-se as variáveis *i* e *j* mais um [(*i*+1); (*j*+1)], deslocando-se para a próxima linha e coluna, reduzindo o quadro com a exclusão da demanda D_j e capacidade F_i desde que $i < m$ e $j < n$. Para que o número de variáveis básicas seja atendido é necessário escolher uma célula à direita ou abaixo, $Célula_{(ij+1)}$ *ou* $Célula_{(i+1;j)}$, para assumir o valor zero, isto é, criar uma variável básica degenerada. Na última designação o valor da oferta que está sobrando e demanda que ainda não foi atendida deve ser iguais e esta quantidade é o valor da última variável básica definida pelo método.

3.3.2. Método *Dual - Stepping-Stone*

Uma vez encontrada a solução inicial, a próxima fase do algoritmo objetiva a melhoria da solução, sendo ela progressiva e interrompida quando a solução ótima é encontrada. O método é estabelecido sobre o equilíbrio das equações enquanto se faz a avaliação da contribuição das variáveis não-básicas. Algumas variáveis devem decrescer uma unidade e outras aumentar uma unidade enquanto o efeito conjunto dessas variações é mantido (PIZZOLATO, GANDOLPHO, 2009).

Quando a variável candidata traz algum benefício à função objetiva, a mesma toma o maior valor possível $\theta \geq 0$. Tanto para medir o efeito de uma variável não básica quanto para encontrar o maior valor para θ, é necessário definir um circuito fechado de variáveis básicas.

3.3.3. Metodologia Utilizada

Para definir qual será a variável não básica para entrar na base do momento, caso a solução atual não esteja no ótimo, faz-se o seguinte cálculo: $c_{ij} - u_i - v_j = 0$ para as variáveis básicas, escolhendo arbitrariamente um valor para algum u_i ou v_j. Como os valores c_{ij} ´s são conhecidos, é possível obter todos os outros valores de u_i e v_j.

Após isso, para cada variável não básica calcula-se $\overline{c_{ij}} = c_{ij} - u_i - v_j$. Caso $\overline{c_{ij}}$ seja um valor positivo para todo *i* e todo *j* então a solução é ótima. Caso exista um ou mais valores zeros e os outros positivos, então as variáveis x_{ij} tais que $\overline{c_{ij}} = 0$, poderão entrar na base sem aumentar o valor da função objetivo. Porém, também não diminuirão o valor da função objetivo, caracterizando múltiplas soluções. Quando existir algum $\overline{c_{ij}} < 0$, então a variável associada ao "mais" negativo entrará na base e uma variável básica deixará de ser básica. O circuito fechado comentado é a transferência de uma quantidade transportada para a nova variável básica. No programa desenvolvido utilizou-se uma técnica que consiste em:

(1) Fixar a variável não básica como candidata a entrar na base;
(2) Analisar todas as linhas e colunas a fim de encontrar o circuito fechado, o ciclo. O artifício utilizado consiste em um processo de eliminação que percorre todas as linhas e colunas. Quando a linha ou coluna apresenta apenas uma variável básica em sua extensão, a mesma será excluída e uma nova linha ou coluna será analisada. Esse procedimento persiste até o momento que restem somente as linhas e colunas com duas variáveis básicas;
(3) Com o ciclo determinado, atribui-se um determinado valor para a variável candidata a entrar na base. A transferência de quantidades transportadas para as variáveis básicas do circuito se alternam em valores positivos e negativos, ou seja, algumas variáveis receberam quantidades enquanto outras recompensaram visando o equilibro das equações. O valor da nova variável básica será igual a menor quantidade presente dentre as variáveis que recompensam as restrições;
(4) Com o valor da quantidade determinada, as variáveis do circuito são atualizadas, a variável candidata torna-se básica e a variável escolhida que teve suas quantidades zeradas deixa de ser básica;
(5) Calcula-se novamente o $c_{ij} - u_i - v_j = 0$ para as variáveis básicas e posteriormente para as não básicas a fim de verificar se a solução já se encontra no ótimo, quando não, todo o programa desenvolvido é novamente executado buscando a solução ótima;

4. INSTÂNCIAS GERADAS

Como foi visto na seção anterior, para fazer-se uso dos softwares indicados, faz parte da solução gerar arquivos de texto nos formato *LP* ou *LNG*. Pensando-se na inconveniência de digitar manualmente os arquivos de texto para problemas demasiadamente grandes, o programa feito em VB.Net gera esses arquivos automaticamente, uma vez que as instâncias (dados) estejam inseridas na tela.

A geração de instâncias consiste basicamente em gerar números aleatórios para compor todos os dados que o usuário teria que informar manualmente para o programa: os custos de transporte C_{ij}, os valores das demandas e das ofertas. Para essas instâncias foram estabelecidas intervalos de valores de números inteiros entre 0 e 100 para os valores dos custos e entre 50 e 300 para os valores de demanda e oferta. Os dados coletados após a resolução de cada modelo gerado foram dispostos na Tabela 1:

GRUPO I - DIMENSÃO ENTRE 5 e 15							
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a solução ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
5 X 10	CPLEX	346	SIM	7 X 12	CPLEX	380	SIM
	LINGO	3828	SIM		LINGO	1570	SIM
	CN / SS	4	SIM		CN / SS	5	SIM
	SI/ SO	0 / 4	-		SI/ SO	0	-
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
12 X 12	CPLEX	948	SIM	10 x 15	CPLEX	571	SIM
	LINGO	4281	SIM		LINGO	4102	SIM
	CN / SS	5	SIM		CN / SS	5	SIM
	SI/ SO	0 / 4	-		SI/ SO	0	-
GRUPO II - DIMENSÃO ENTRE 30 e 60							
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
49 X 32	CPLEX	1150	SIM	50 X 50	CPLEX	307	SIM

	LINGO	4945	SIM		LINGO	5410	SIM
	CN / SS	27	SIM		CN / SS	55	SIM
	SI/ SO	0 / 26	-		SI/ SO	0 / 49	-
CONTINUAÇÃO GRUPO II - DIMENSÃO ENTRE 30 e 60							
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
57 X 33	CPLEX	352	SIM	60 X 30	CPLEX	428	SIM
	LINGO	4319	SIM		LINGO	4068	SIM
	CN / SS	29	SIM		CN / SS	29	SIM
	SI/ SO	0 / 28	-		SI/ SO	0 / 28	-
GRUPO III - DIMENSÃO ENTRE 90 e 180							
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
90 X 95	CPLEX	1165	SIM	120 X 120	CPLEX	802	SIM
	LINGO	17923	SIM		LINGO	24681	SIM
	CN / SS	349	SIM		CN / SS	743	SIM
	SI/ SO	1 / 346	-		SI/ SO	2 / 739	-
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
176 X 1 3 5	CPLEX	1731	SIM	160 X 148	CPLEX	971	SIM
	LINGO	4350	SIM		LINGO	39699	SIM
	CN / SS	1411	SIM		CN / SS	1657	SIM
	SI/ SO	3 / 1407	-		SI/ SO	3 / 1652	-
GRUPO IV – DIMENSÃO ENTRE 250 e 800							
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
250 X 2 5 0	CPLEX	1664	SIM	422 X 395	CPLEX	6551	SIM
	LINGO	104683	SIM		LINGO	312263	SIM
	CN / SS	9045	SIM		CN / SS	39087	SIM
	SI/ SO	08 / 9035	-		SI/ SO	22 / 39062	-
Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?	Dimensão	Formas de resolução	Tempo (ms.)	Atingiu a Solução Ótima?
650 X 6 5 0	CPLEX	10233	SIM	789 x 654	CPLEX	15032	SIM
	LINGO	885895	SIM		LINGO	873226	SIM
	CN / SS	182045	SIM		CN / SS	222709	SIM
	SI/ SO	55 / 181984	-		SI/ SO	64 / 222638	-

TABELA 1 – RESULTADOS DAS INSTÂNCIAS GERADAS

Os modelos foram separados em quatro grupos, de acordo com a dimensão do problema, como demonstra a tabela acima. Os problemas foram resolvidos pelas três formas de resolução, *CPLEX*, *LINGO* e Método Dual-*Steeping-Stone* (*CN / SS* – Canto Noroeste / *Stepping-Stones*). A opção *SI / SO* que aparece na quarta linha como forma de resolução, na verdade, faz parte da resolução do *CN* / SS. Essa opção tem o intuito apenas de subdividir o tempo computacional total (*CN*/*SS*) entre o tempo gasto na resolução da solução inicial (SI), seguido do tempo gasto para posterior otimização do problema. A unidade de tempo utilizada foi o milissegundo (*ms*). Verificou-se que todas as simulações realizadas atingiram a solução ótima global.

4.1. ANÁLISE COMPARATIVA

A coleta de dados realizada na seção anterior traz como propósito uma avaliação da utilização de cada forma de resolução de maneira integral, incluindo como parâmetros de análise o desempenho, a eficácia, a dificuldade de aprendizagem, a linguagem de programação necessária, e as peculiaridades de cada *software*. A intenção maior é avaliar esse conjunto de fatores simultaneamente, de maneira a não somente analisar o tempo de processamento da solução em si, mas toda a iteração que é percebida pelo usuário desde a escrita dos modelos até a obtenção dos arquivos de solução final.

Primeiramente, em relação ao desempenho, verificou-se que para problemas de dimensões até as do grupo II, o método Dual-*Stepping-Stone* programado possui o maior desempenho. A partir de dimensões maiores, em torno de 150 até o máximo testado, o software *CPLEX Optimizer* se sobressai e passa a ser o primeiro nesse quesito. O software *LINGO12*® possui o pior desempenho em relação ao demais segundo essa análise, devido

principalmente às peculiaridades do próprio *software*. O *LINGO* é uma interface de programação, diferentemente do *CPLEX Optimizer*, que é fundamentalmente um solver.

Gomes, Lopes e Montané (2013) afirmam que o *LINGO* e o *CPLEX* retornam valores iguais para as variáveis na maioria dos problemas pelos autores resolvidos, mas os recursos utilizados pelo *CPLEX* para a apresentação de resultados são notoriamente melhores. Esses valores registrados para o *LINGO* são altos em virtude de o *LINGO* não permitir uma acessibilidade rápida através do *MS-DOS*, e no momento em que retorna a solução, carregar em seu próprio programa um conjunto enorme de gráficos e outros retornos visuais de solução. Contabilizou-se o tempo absoluto do processo de chamada e inicialização do *software*, bem como o retorno da solução para o usuário. Cabe destacar que, caso houvesse a possibilidade de usar o *LINGO* sem os retornos visuais, o desempenho do *software* poderia ser melhor.

No que concerne à eficácia, verificou-se que todos possuem a mesma, uma vez que todos os modelos resultaram na solução ótima. No que é referente à linguagem de programação necessária, pode-se considerar que para utilizar os softwares *LINGO* e o *CPLEX* através da interface criada, a dificuldade é a mesma. É necessário que o usuário desenvolva certa afinidade com o programa e tenha noções básicas de programação, para ser capaz de desenvolver o código que cria os arquivos de textos automaticamente, que leia o arquivo solução e mostre os resultados, e que faça o processo de resolução.

É o método *Dual-Stepping-Stone*, porém, que necessita o maior nível de linguagem e conhecimentos específicos de pesquisa operacional e de programação. Para programar o algoritmo em questão, é necessário ter um domínio maior do conhecimento lógico concernente, o que implica um tempo relativamente maior de desenvolvimento da ferramenta. Uma vez que a ferramenta esteja programada, qualquer uma das três formas de resolução podem ser escolhidas, sem haver maior ou menor dificuldade de aprendizagem no uso entre elas.

Para finalizar, cabe fazer um comentário sobre a origem de cada software. O uso dos softwares de programação matemática implica em algumas inconveniências. Para Alencar, Rocha e Nogueira (2012), essa utilização está associada a um elevado custo para o licenciamento de cópias para Instituições de Ensino Brasileiras. Além disso, há certa dificuldade de conhecer o funcionamento interno, por serem eles fornecidos como caixas pretas e originalmente destinados ao uso profissional, e não acadêmico. Ainda, podem possuir usabilidade e flexibilidade inadequadas às necessidades dos usuários que não estão tão e somente preocupados com a obtenção do resultado ao "menor tempo de resolução possível".

5. CONCLUSÕES E TRABALHOS FUTUROS

Foram apresentadas neste trabalho duas alternativas de *softwares*, para a resolução do problema clássico do transporte, bastante difundidos na resolução de problemas diversos na área de Pesquisa Operacional, o *LINGO12*®, e o *CPLEX Optimizer*®. Outro método de resolução, o método *Dual-Stepping-Stones*, foi implementado e surgiu como uma terceira alternativa para a resolução do problema.

É bastante interessante avaliar a criação dessa terceira forma de resolução, uma ferramenta que foi em sua totalidade desenvolvida sem a utilização de *software* específico de otimização e que possui a mesma eficácia que os outros dois *softwares* de uso comercial.

Verificou-se que o uso dessa terceira forma de resolução é bastante eficiente e que atinge os mesmo resultados que os *softwares* citados em todo o desenvolver desse trabalho, e com um desempenho melhor para problemas considerados pequenos. Para problemas maiores, verificou-se que o *software CPLEX*® possui melhor desempenho quando comparado aos outros, dado o tempo de processamento considerado.

Entretanto, percebe-se que no mundo corporativo há dificuldade de investimento e de utilização de *softwares* comerciais para a resolução de problemas, inclusive o problema clássico do transporte. Existe então uma possibilidade de difundir esta prática ou cultura que

consiste em promover o desenvolvimento de soluções programadas por profissionais que detém o conhecimento dos métodos da Pesquisa Operacional. Assim as empresas poderiam chegar a resultados melhores, mais confiáveis e com reais condições de otimização de recursos e aumento de lucro, simplesmente por obter a solução otimizada.

Fica evidenciado por este estudo, por fim, que cada forma de resolução possui seus atributos e suas dificuldades. A escolha da melhor forma de resolução depende das necessidades do usuário, do conhecimento técnico de programação e das características do problema.

6. REFERÊNCIAS

[1] ALENCAR, W.S., ROCHA, B.M., NOGUEIRA, E.A. *Rhodes 2.0: Software Educacional para o Ensino de Programação Linear.* 2012, p 4.

[2] ARENALES, M. N. *Pesquisa Operacional*. Rio de Janeiro: Elsevier, 2007, p 524.

[3] BALLOU, R. H. *Gerenciamento da Cadeia de Suprimentos: planejamento, organização e logística empresarial*. Tradução de Elias Pereira. 4. ed. Porto Alegre: Bookman, 2001.

[4] COLIN, E. C. *Pesquisa Operacional: 170 aplicações em estratégia, finanças, logística, produção, marketing e vendas*. Rio de Janeiro, 2007.

[5] COUTINHO, R.C. *Algoritmos distribuídos para o problema de atribuição de clientes a servidores em redes de distribuição de conteúdos.* Dissertação de Mestrado em Computação da UFF. 2011

[6] GOMES, M. E. B. L., LOPES, O. P., MONTANÉ, F. A. T. *Estudo e Modelagem de Problemas de Fluxo em Redes utilizando softwares de programação matemática.* Confict. 2013.

[7] HILLIER, F. S., LIEBERMAN, G.J. *Introdução à Pesquisa Operacional*. Porto Alegre. 9ª Edição. 2012.

[8] JUNIOR, A. de C. G; SOUZA, M.J.F; *Lingo - Parte 1: Manual de Referência. Departamento de Computação*. Universidade Federal de Ouro Preto. Rio de Janeiro, 2004.

[9] LOPES, O. P., GOMES, M. E. B. L., & MONTANÉ, F. A. T. *Estudo e Implementação de modelos de localização utilizando os softwares Lingo E Cplex.* Confict. 2013.

[10] MORÉ, J.J., WRIGHT, S.J. *Optimization Software Guide*. ISPN. Philadelphia, Pennnsylvania. Second Printing. 1994.

[11] MURTY K. G. *Linear Programming*. Wiley, New York: John Wiley e Sons, 1983.

[12] PIZZOLATO, N.D., GANDOLPHO A.A. *Técnicas de Otimização*. Rio de Janeiro. 2009

[13] SILVA, T.L.C. *Nova Metodologia para resolução de problemas de transporte em casos esparsos.* Tese de Doutorado em Métodos Numéricos em Engenharia, UFPR, 2012.