CONCEPÇÃO E TESTE DE SOFTWARE NO PROCESSO DE ENSINO-APRENDIZAGEM DA DISCIPLINA PESQUISA OPERACIONAL

Giancarlo de F. Aguiar – giancarl@up.com.br
Universidade Positivo - UP, Engenharia da Computação
Rua Pedro Viriato Parigot de Souza, 5300, Campo Comprido
81280-330 – Curitiba – Paraná
Alessandro Brawerman – brawerman@up.com.br
Bárbara de C. X. C. Aguiar – babi.eg@ufpr.br
Universidade Federal do Paraná - UFPR, Departamento de Expressão Gráfica
Centro Politécnico, Jardim das Américas
81280-330 – Curitiba – Paraná
Volmir E. Wilhelm – volmirw@gmail.com
Universidade Federal do Paraná, Departamento Matemática
Centro Politécnico, Jardim das Américas
81531-990 – Curitiba – Paraná

Resumo: *O desenvolvimento de softwares acadêmicos em cursos de graduação pode tornar o processo de ensino e aprendizagem mais atraente, sejam eles concebidos por professores ou por estudantes. Este texto ilustra o processo de criação de um software para o cálculo de problemas de programação linear, desenvolvido na disciplina Pesquisa Operacional por estudantes do curso de Engenharia da Computação da Universidade Positivo, em Curitiba no estado do Paraná. Criar um software requer dos estudantes o conhecimento técnico da linguagem de programação utilizada, e um conhecimento sólido, lógico e aprimorado do tema proposto. Este texto apresenta um software tutorial capaz de resolver problemas de programação linear com um número finito de variáveis e restrições. Este instrumento tem como objetivo auxiliar novos estudantes no entendimento da disciplina Pesquisa Operacional, bem como, servir de estrutura para novas implementações.*

Palavras-chave: *Software Tutorial, Programação Linear, Processo de Ensino e Aprendizagem*

1 INTRODUÇÃO

Atualmente, a utilização de recursos computacionais para o tratamento de dados, sejam eles qualitativos ou quantitativos tornou-se imprescindível. O mercado exige agilidade e precisão no estudo dos dados. Quando esta tarefa cabe a um ser humano ele tanto pode cometer erros de precisão, como entrar em fadiga, caso exista o excesso de trabalho. Pelo contrário, os computadores modernos possuem excelente precisão e são muito mais rápidos que os seres humanos (AGUIAR *et al*., 2006).

Dessa forma, foi inevitável o desenvolvimento acelerado de recursos computacionais

direcionados a resolução dos mais variados problemas, sejam eles de ordem econômica, administrativa, industriais e de engenharia. Tornar-se-á mais comum a cada dia, a construção de laboratórios virtuais acadêmicos em cursos de graduação, sejam eles desenvolvidos por professores ou por estudantes.

Este trabalho apresenta o processo de construção de um software desenvolvido na disciplina de Pesquisa Operacional com o apoio das lógicas de programação trabalhadas na disciplina de Fundamentos de Programação, realizado com estudantes do curso de Engenharia da Computação da Universidade Positivo, em Curitiba no estado do Paraná, no ano de 2011.

Segundo Drozdek (1998) e Tannenbaum (1995) as literaturas de Fundamentos de Programação muitas vezes apresentam certas estruturas de forma simples e sem exemplos de aplicação prática, o que pode desestimular os estudantes. Os laboratórios virtuais acadêmicos ou softwares podem contribuir na relação ensino-aprendizagem como fator motivacional, despertando o interesse nos estudantes tanto para a pesquisa como na aplicação dos conhecimentos adquiridos no decorrer do curso.

Desenvolver um software requer dos estudantes tanto o conhecimento técnico da linguagem de programação utilizada, como também, um conhecimento sólido, lógico e aprimorado do tema abordado. Segundo Lopes (2011) a evolução da computação contribui para a melhoria da capacidade mental. Os avanços reforçaram mudanças em todas as áreas de conhecimento moderno, uma vez que em todos os seguimentos verificamos alguma nova forma de tecnologia para executar as operações que antes eram executados pelos homens.

Na tentativa de garantir o ganho de eficiência na aprendizagem, os processos de ensino tem sofrido mudanças constantes, aperfeiçoando e se utilizando de novas tecnologias, numa relação pedagógica entre teoria e prática motivadora e de interação. O novo cenário educacional exige o caminhar conjunto entre os sistemas de ensino e as novas tecnologias (FONSECA, et al., 2009).

Aliando teoria e prática, os estudantes tiveram que desenvolver um software tutorial com o objetivo de auxiliar novos estudantes no entendimento da matéria Pesquisa Operacional, bem como, servir de estrutura para novas implementações em Fundamentos de Programação.

2 DESENVOLVIMENTO

A seguir são delineados quatro momentos base para o desenvolvimento do trabalho.

1- No início do ano letivo foram dedicadas 2 horas-aula de cada disciplina participante do projeto para a explanação do trabalho. Seguem algumas das orientações:
   a. Divisão de equipes de trabalho (duplas);
   b. Objetivos de cada equipe (desenvolver um software para o método simplex);
   c. Tarefas das equipes durante o ano letivo (montar um algoritmo para o método simplex e desenvolver o software);
   d. Apresentação dos trabalhos em seminários para a turma;

2- Em seguida, utilizando uma ferramenta de programação visual (C, C++ ou Visual Studio), os estudantes iniciaram o desenvolvimento dos softwares tutoriais, com o auxílio dos conhecimentos adquiridos na disciplina de Fundamentos de Programação.
3- Na disciplina de Pesquisa Operacional os estudantes tiveram aulas de programação linear e construíram uma base lógica do algoritmo para resolução de problemas de programação linear (método simplex). Onde a cada bimestre foi realizado um encontro com o professor orientador para a ponderação de nota para o trabalho desenvolvido até aquele presente momento.
4- O prazo de término do trabalho foi o 4° bimestre com a data de apresentação dos seminários já pré-definida. Após as explanações e defesas de trabalhos.

2.1 Softwares

Os softwares desenvolvidos são tutoriais que ilustram de forma prática como resolver problemas de programação linear e armazenar um conjunto de informações, através de exemplos didáticos bem conhecidos de problemas nessa área. Para a implementação dos softwares foram definidos alguns algoritmos padrão para o início do desenvolvimento. A Figura 1 a seguir ilustra um resumo do algoritmo simplex.

Passo 0 **Inicialização:** com solução básica factível x^0 , $t \leftarrow 0$;

Passo 1 **Direções simplex:** construir Δx associada com não básica x_j , calcular custo reduzido $\underline{c_j} = c\Delta x$;

Passo 2 **Otimalidade:** se nenhuma direção melhora ($\nexists\ \underline{c_j} > 0$ para max $\underline{c_j} < 0$ para min), então parar, x^t é ótima; senão escolher nova direção Δx que melhora valor função objetivo; seja x_p a variável que entra na base;

Passo 3 **Passo:** se todos componentes de Δx forem não negativas, então parar, o modelo é ilimitado; senão determinar δ e escolher variável que deixa a base, x_r : $\delta \leftarrow (\frac{x_r}{-\Delta x_r})$;

Passo 4 **Novo ponto e base:** determinar nova solução $x^{t+1} = x^t + \delta \Delta x^{t+1}$ e trocar x_r por x_p ; t = t + 1; ir para o Passo 1;

Figura 1 – Resumo do algoritmo simplex (ajustado)
Fonte: http://www.dca.fee.unicamp.br/~gomide/courses/EA044/transp/EA_044_AlgoritmoSimplexI.pdf

Cada equipe ficou responsável pela implementação e desenvolvimento de uma estrutura de armazenamento diferente. Sendo então definida uma competição entre eles para obtenção do menor tempo na obtenção dos resultados.

A partir deste ponto, cada equipe foi alterando o seu algoritmo em busca de um tempo inferior ao obtido pelo método apresentado na literatura. A Figura 2 a seguir ilustra um código realizado em C.

Entrada de Dados:

```
printf ("Num. var.: "); scanf ("%u",&n); printf ("Num. inec.: "); scanf ("%u",&e); c = calloc ( n+1, sizeof( float)); a = calloc ( e+1, sizeof( float*)); x = calloc ( e+1, sizeof( unsigned)); h = calloc ( n+1, sizeof( float)); v = calloc ( e+1, sizeof( float)); for ( i=0;i<n;i++) { printf ("Coef. x(%u) en la ec. del Maximo:",i+1); scanf ("%f",&c[i]); } for ( j=0;j<e;j++) { a[j] = calloc ( n+1, sizeof(float)); for ( i=0;i<n;i++) { printf ("Coef. x(%u) en la %u ec.:",i+1,j+1); scanf ("%f",&a[j][i]); } printf ("Term.Indep. de la %u ec.:",j+1); scanf ("%f",&a[j][n]); printf ("Subindice de variable aux. de la inecuacion h(i)"); scanf ("%u",&x[j]); x[j]--; }
```

Organizando a Matriz de Dados:

```
for ( i=0; i<=n;i++ ){ a[e][i]=-c[i]; for ( j=0;j<e;j++){ a[e][i]+=a[j][i]*c[x[j]]; }

} for (i=0; i< n; i++ ){ printf ("c(%u)= %7.2f\n", i+1,c[i]); }
```

Obtendo a Coluna do Elemento Pivoteador:

```
bi=1; for ( i=0;i<n;i++){ if ( a[e][i]< a[e][bi] ) bi=i; } if ( a[e][bi]>=0 ) break;
```

Pbtendo a Linha do Elemento Pivoteador:

```
bj=1; for ( j=0;j<e;j++){ if ( a[j][n]*a[bj][bi] < a[bj][n]*a[j][bi] ) bj=j; } printf ("+"); for ( i=0;i<n+2;i++) printf ("--------"); printf ("+\n"); for ( j=0; j< e; j++){ printf ("| x%u | %7.2f | ",x[j]+1,a[j][n] ); for ( i=0; i<n; i++ ){ if ((i==bi)&&(j==bj)) { printf ("%7.2f*",a[j][i] ); } else { printf ("%7.2f ",a[j][i] ); } } printf("|\n"); } printf ("+"); for ( i=0;i<n+2;i++) printf ("--------"); printf ("+\n"); printf (" | %7.2f | ",a[e][n] ); for ( i=0; i<n; i++ ){ printf ("%7.2f ",a[e][i] ); } printf("|\n\n");
```

Guardando os Dados da Linha e Coluna:

```
for (i=0;i<=n;i++ ){ h[i]=a[bj][i]; } for ( j=0;j<=e;j++ ){ v[j]=a[j][bi]; } for (i=0;i<=n;i++ ){ for ( j=0;j<=e;j++ ){ a[j][i] -= h[i]*v[j]/h[bi]; } } for (i=0;i<=n;i++ ){ a[bj][i]=h[i]/v[bj]; } x[bj]= bi; }
```

Saída de Dados com a Solução:

```
printf ("+"); for ( i=0;i<n+2;i++) printf ("--------"); printf ("+\n"); for ( j=0; j< e; j++){ printf ("| x%u | %7.2f | ",x[j]+1,a[j][n] ); for ( i=0; i<n; i++ ){ printf ("%7.2f ",a[j][i] ); } printf("|\n"); }printf ("+"); for ( i=0;i<n+2;i++) printf ("--------"); printf ("+\n"); printf (" | %7.2f | ",a[e][n] ); for ( i=0; i<n; i++ ){ printf ("%7.2f ",a[e][i] ); } printf("|\n\n"); printf ("\nSolucion:\n"); for ( j=0; j<e; j++){ printf ("x%u=%7.2f\n",x[j]+1,a[j][n]); }
```

Espaço de Variáveis:

```
for ( j=0; j<=e; j++) free(a[j]); free(c); free(a); free(x); free(h); free(v); return 0;}
```

Figura 2 – Algoritmo simplex escrito em linguagem C (ajustado)
Fonte: http://html.rincondelvago.com/algoritmo-en-c-del-metodo-simplex.html

3 RESULTADOS

A seguir está ilustrado um conjunto de figuras com algumas das interfaces de um dos trabalhos selecionados. A figura 3 a seguir ilustra a tela inicial do Software.

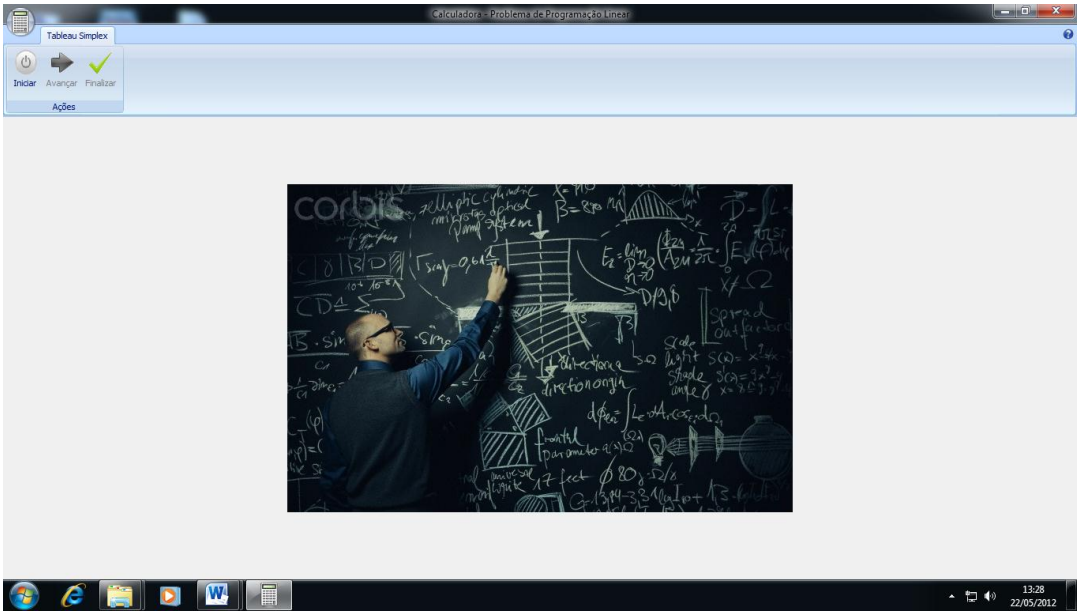

Figura 3 – Interface inicial do software

Ao clicar na ferramenta iniciar o programa irá abrir uma nova janela para inserir os dados do problema de programação linear como mostra a Figura 4 abaixo.

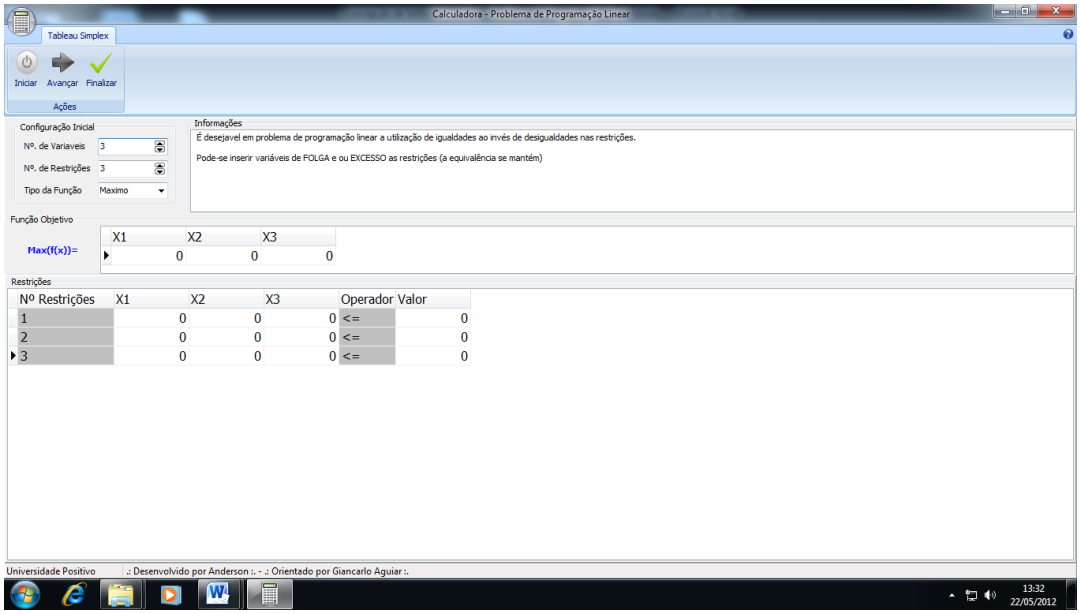

Figura 4 – Interface para inserir dados de Problemas de Programação Linear

O estudante pode escolher o número de variáveis e restrições do problema. Para o teste de software, iremos escolher um problema com duas variáveis e quatro restrições, como o modelo do problema linear 1 a seguir.

$$
\begin{aligned}
& Max\, Z = 600.x_1 + 800.x_2 \\
& s.a.: \quad x_1 + x_2 \leq 100 \\
& \qquad 3.x_1 + 2.x_2 \leq 240 \\
& \qquad x_1 \leq 60 \\
& \qquad x_2 \leq 80 \\
& x_1, x_2 \geq 0
\end{aligned}
\tag{1}
$$

Inserindo os coeficientes das variáveis no programa obtemos a interface (à esquerda) e clicando em avançar obtemos a interface (à direita), ilustradas na Figura 5.

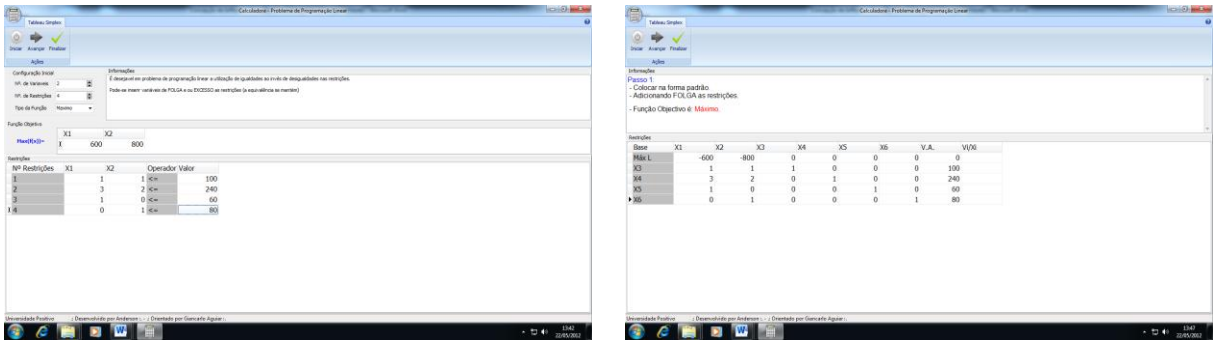

Figura 5 – Interface de dados para o problema linear 1 anterior

O problema foi colocado em sua forma padrão pelo software que adicionou as variáveis de folga as restrições. Clicando em avançar vamos obtendo novas interfaces do programa que vai rodando o algoritmo passo a passo como ilustra a figura 6 a seguir.

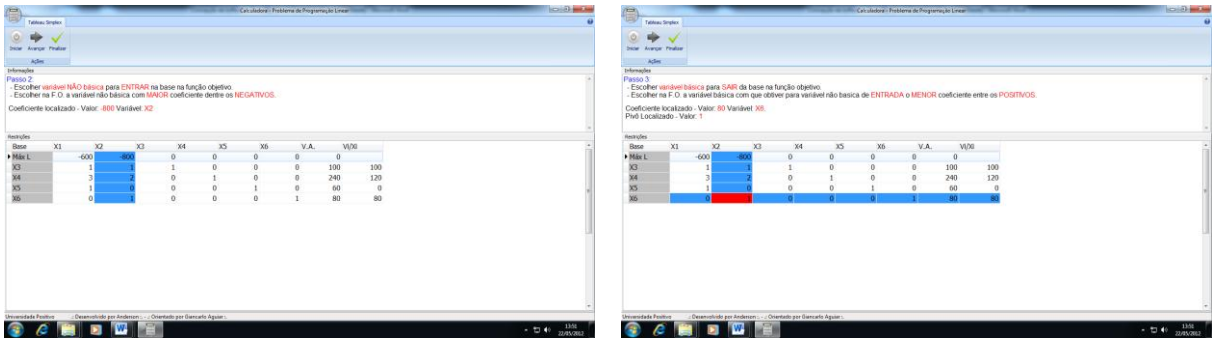

Figura 6 – Interface do programa rodando o algoritmo passo a passo

A figura 6 à esquerda ilustra que o programa escolheu a variável x_2 para entrar na base, enquanto que na figura da direita, o programa escolheu a variável x_6 para sair da base. O

elemento pivô está sombreado em vermelho. Clicando novamente em avançar obtemos a figura 7 a seguir.

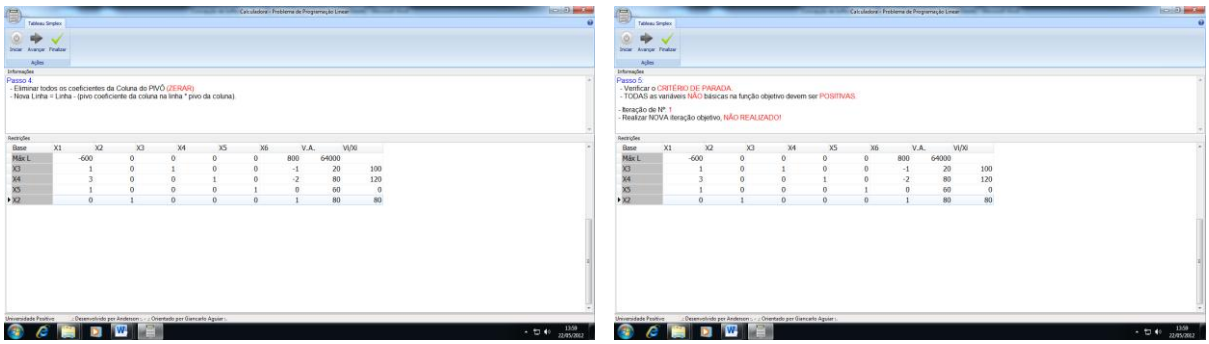

Figura 7 – Passos do algoritmo simplex realizados pelo software

A figura 7 à esquerda ilustra que o programa realizou o pivoteamento, zerando os coeficientes na coluna do elemento pivô, com exceção dele próprio. À direita investiga o critério de parada para o algorítimo (coeficientes das variáveis na função objetivo devem ser positivos). Como o coeficiente da variável x_1 ainda é negativo, então o programa irá fazer uma nova iteração do algoritmo. As Figuras 8 e 9 a seguir ilustram cada passo novamente.

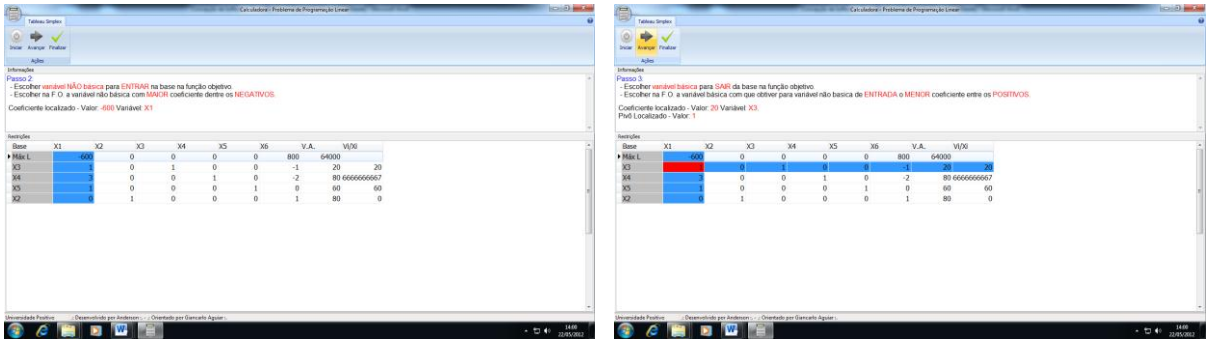

Figura 8 – Passos do algoritmo simplex realizados pelo software

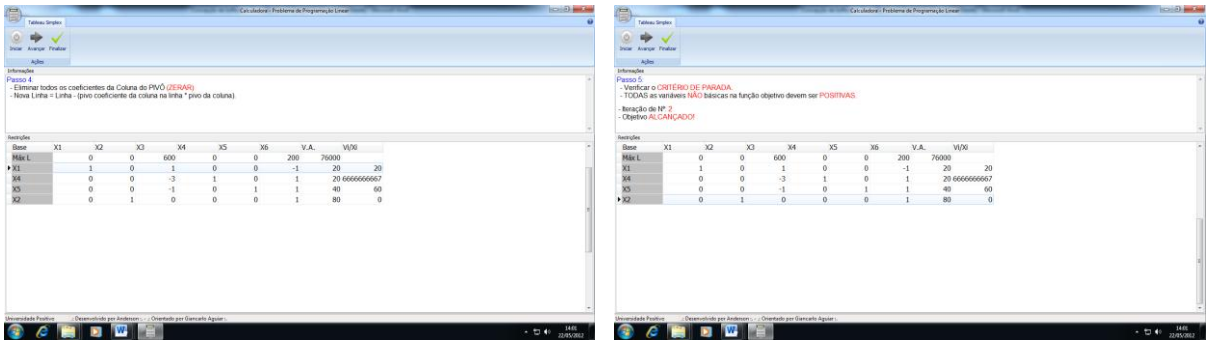

Figura 9 – Passos do algoritmo simplex realizados pelo software

Após a segunda iteração, o programa chega à solução do problema proposto como ilustra a Figura 10 abaixo.

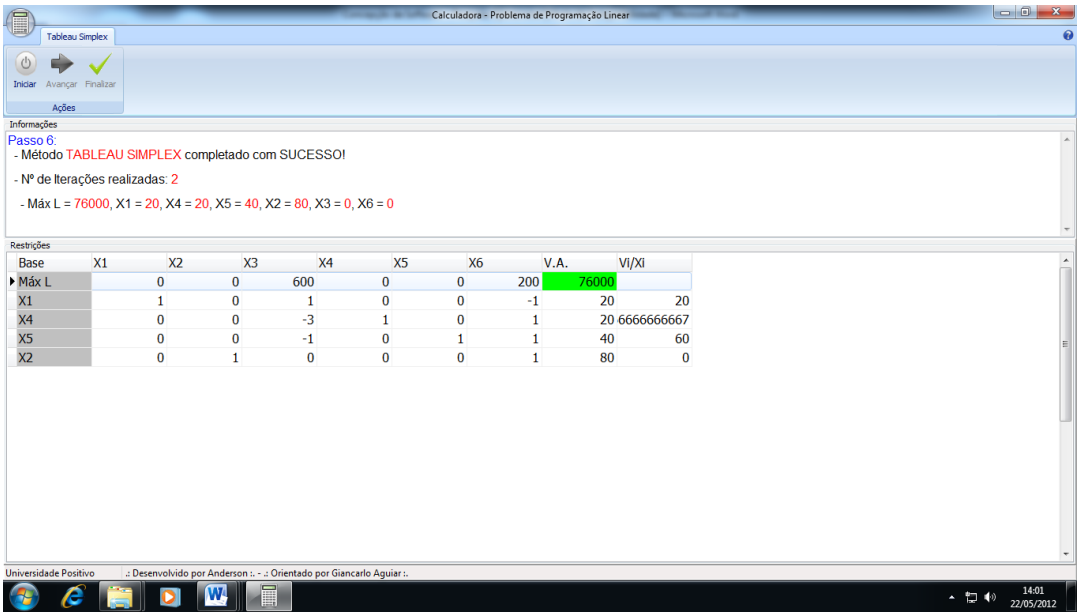

Figura 10 – Resultado final gerado pelo programa

O software ilustra a resposta final para o problema que é o máximo valor para a função objetivo de 76.000, e para as variáveis (x_1 = 20, x_2 = 80, x_3 = 0, x_4 = 20, x_5 = 40 e x_6 = 0).

4 CONSIDERAÇÕES FINAIS

O trabalho pode contribuir significativamente para o embasamento teórico (pesquisa dos tópicos selecionados pelo orientador, que neste caso foi o estudo do método simplex) quando realizado em conjunto com o desenvolvimento físico através da aplicação tecnológica (desenvolvimento do software).

Pode-se notar o engajamento dos estudantes quanto à forma de modelar a estrutura de dados para melhor adaptá-la ao problema proposto.

Durante o ano letivo, muitos estudantes se mostravam mais motivados no ensino e aprendizagem de novos conteúdos.

O desenvolvimento culminou em uma grande rede de grupos de estudo para a sua incubação, o que aproximou ainda mais os estudantes entre si.

Dadas às duas condições anteriores (motivação e interação), foi construído um processo ensino-aprendizagem que pode gerar resultados mais significativos aos alunos.

5 REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, G. F.; AGUIAR, B. C. X. C.; WILHELM, V. E. Obtenção de Indices de Eficiência para a Metodologia *Data Envelopment Analysis* Utilizando a Planilha Eletrônica Microsoft Excel. Revista da Vinci, Curitiba, v.3, n.1, p. 157-169, 2006.

DEL VAGO, R. **Ingeniería em sistemas computacionales**. Universidad Tecnológica de México. Disponível em: <http://html.rincondelvago.com/algoritmo-en-c-del-metodo-simplex.html>. Acesso em 14/04/2012.

DROZDEK, A. Estrutura de Dados e Algoritmos em C++. São Paulo: Thomson Learning Ltda, 1998.

FONSECA, L. M. M.; et al. Inovação Tecnológica no Ensino da Semiotécnica e Semiologia em Enfermagem Neonatal: do Desenvolvimento à Utilização de um Software Educacional. Revista Texto & Contexto, Florianópolis, v.18, n.3, p. 542-548, 2009.

GOMIDE, F. **Planejamento e análise de sistemas de produção**. FEEC–Unicamp. Disponívelem:<http://www.dca.fee.unicamp.br/~gomide/courses/EA044/transp/EA_044_AlgoritmoSimplexI.pdf>. Acesso em: 08/05/2012.

LOPES, A. C. C.; et al. Construção de Avaliação de Software Educacional sober Cateterismo Urinário de Demora. Revista da Escola de Enfermagem da USP, São Paulo, v.45, n.1, p. 215-222, 2011.

TANNENBAUM, A. M. Estruturas de Dados usando C. São Paulo: Makron Books, 1995.

DEVELOPMENT AND TESTING OF SOFTWARE IN LEARNING-TEACHING PROCESS OF OPERATIONAL RESEARCH DISCIPLINE

Abstract: *The Software development in undergraduate courses students can make the process of teaching and learning more motivating, whether developed by teachers or students. This text illustrates the process creating a software to calculate linear programming problems, developed in the discipline Operational Research by students Computer Engineering at Positivo University, in Curitiba state of Parana. The Developing a software requires of students both the technical knowledge of programming language, but also a solid knowledge, logical and improved of proposed material. This text presents a tutorial software appropriate to solve linear programming problems with a finite number of variables and constraints. This instrument has objective to assist new students in the understanding Operational Research course, as well as serve as framework for new implementations.*

Key-words: *Tutorial Software, Linear Programming, Learning-Teaching Process*