

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE EDUCAÇÃO DO PLANALTO NORTE – CEPLAN
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MATÉRIA: PESQUISA OPERACIONAL

JEAN FELIPE Kruk DE OLIVEIRA
WELINGTON BRAYER COPATTI

TRABALHO I: TRABALHO DE EXTENSÃO

São Bento do Sul
2026

INTRODUÇÃO.....	3
MANUAL DE USO DA FERRAMENTA.....	4
1. APRESENTAÇÃO DA FERRAMENTA.....	4
2. INSTALAÇÃO E CONFIGURAÇÃO.....	5
2.1. REQUISITOS DO SISTEMA.....	5
2.2. DOWNLOAD DO PROGRAMA.....	5
2.3. INSTALAÇÃO DO PROGRAMA.....	6
3. INTERFACE DA FERRAMENTA E PRÉ-CONFIGURAÇÕES.....	7
4. DEFINIÇÃO DOS ELEMENTOS.....	10
4.1. VARIÁVEIS DE DECISÃO.....	10
4.2. FUNÇÃO OBJETIVO.....	11
4.3. RESTRIÇÕES.....	11
4.4. TIPOS DE VARIÁVEIS.....	12
4.4.1. VARIÁVEIS CONTÍNUAS.....	12
4.4.2. VARIÁVEIS INTEIRAS.....	13
4.4.3. VARIÁVEIS BINÁRIAS.....	13
5. EXEMPLO DIDÁTICO SIMPLES.....	14
6. RESOLUÇÃO NA FERRAMENTA.....	14
7. INTERPRETAÇÃO DE RESULTADOS.....	17
RESOLUÇÃO DE UM PROBLEMA COMPLEXO.....	18
1. DESCRIÇÃO DO PROBLEMA COMPLEXO.....	18
2. MODELAGEM MATEMÁTICA DO PROBLEMA.....	18
2.1. VARIÁVEIS DE DECISÃO.....	18
2.2. FUNÇÃO OBJETIVO.....	19
2.3. RESTRIÇÕES.....	19
2.4. TIPO DAS VARIÁVEIS.....	20
3. IMPLEMENTAÇÃO NA FERRAMENTA.....	20
4. RESULTADOS OBTIDOS.....	22
5. DISCUSSÃO E ANÁLISE.....	22
CONCLUSÃO.....	23
REFERÊNCIAS.....	23

INTRODUÇÃO

Em nossos últimos anos, nos mostraram um avanço tecnológico grande, e um aumento de volume de informações disponíveis ainda maior, que acabaram por transformar a maneira como empresas e organizações tomam suas decisões. Hoje, confiar apenas em experiência ou intuição já não basta para lidar com problemas mais complexos que envolvem custos, prazos, produção e distribuição de recursos. Nesse ponto de vista, a programação linear surge como uma ferramenta matemática que pode nos auxiliar na resolução desses tipos de problemas de forma mais eficiente e estratégica.

Podemos resumir a programação linear como uma técnica de otimização utilizada para encontrar sempre a melhor solução possível para determinados problemas, mas sempre respeitando as restrições impostas previamente. Por meio disso, podemos minimizar custos ou maximizar lucros. Reduzir esses desperdícios e melhorar o aproveitamento dos recursos que são disponibilizados numa empresa. Essa metodologia é amplamente utilizada em várias áreas como por exemplo: na logística, planejamento de produção, transporte, investimentos financeiros, controles de estoque, distribuição de mercadorias, etc.

Com evolução e mais dificuldades dentro das áreas organizacionais das empresas, acabou se tornando inviável realizar alguns cálculos manualmente, em especial quando há grande quantidade de variáveis e restrições incluídas. Sendo assim, o uso de ferramentas computacionais acabou por se tornar imprescindíveis para automatização de processos, aceleração de análises, tudo para garantir maior precisão nos resultados obtidos. Softwares especializados em programação linear nos permitem modelar problemas encontrados dentro das empresas a fim de encontrar soluções ótimas de maneira efetiva, contribuindo para as tomadas de decisão baseada nesses dados.

Além de facilitar nosso trabalho com cálculos matemáticos, essas ferramentas nos possibilitam ver sempre os melhores cenários nas análises, e permite identificar gargalos, limitações e possíveis oportunidades de melhorias dentro de um sistema. Dessa forma, as empresas sempre tomam as decisões mais seguras e alinhadas com os seus objetivos estratégicos individuais.

Diante desse tema importante que tratamos, o nosso trabalho tem como objetivo apresentar alguns dos fundamentos da programação Linear e demonstrar, na prática, a utilização de uma ferramenta computacional para modelagem e resolução de problemas de otimização. Além disso, vamos desenvolver um estudo aplicado envolvendo um problema complexo, buscando relacionar os conceitos teóricos com situações semelhantes que encontramos no mercado de trabalho atual.

MANUAL DE USO DA FERRAMENTA

1. APRESENTAÇÃO DA FERRAMENTA

A ferramenta que escolhemos para o desenvolvimento deste trabalho foi o Gusek. O Gusek é um ambiente utilizado para modelagem e resolução de problemas de programação linear, programação inteira, sendo bastante utilizado em contextos acadêmicos e profissionais devido a sua praticidade e eficiência.

O Gusek funciona integrado ao GLPK (GNU Linear Programming Kit), que permite que o usuário escreva modelos matemáticos de otimização de forma organizada, executando cálculos automaticamente. Essa ferramenta possui uma interface simples, facilitando a criação de variáveis, definições de restrições e construção da função objetivo.

A principal finalidade é resolver problemas que envolvem otimização de recursos, onde buscamos encontrar a melhor solução possível para determinadas situações. Assim sendo, o software pode ser utilizado em áreas como logística, transporte, planejamento de produção, controles de estoque, investimentos, distribuição de recursos, etc.

Entre algumas das principais funcionalidades do Gusek, destacam-se:

- Criação e edição de modelos matemáticos.
- Definição de variáveis de decisão e restrições.
- Resolução automática de problemas de maximização e minimização.
- Suporte para variáveis inteiras e binárias.
- Exibição dos resultados e da solução ótima encontrada.

- Integração com o solver GLPK para processamento dos cálculos.

Além dessas funções, o Gusek se destaca por ser uma ferramenta muito leve, gratuita e de fácil utilização, tornando-se uma boa opção para estudos e aplicações relacionadas à programação linear.

2. INSTALAÇÃO E CONFIGURAÇÃO

2.1. REQUISITOS DO SISTEMA

- Sistema operacional Windows.
- Espaço livre para instalação do programa(aproximadamente 5 a 10Mb).
- Permissão para executar aplicativos no computador.
- Por conta do software ser leve, não é necessário alto desempenho da máquina para funcionamento.

2.2. DOWNLOAD DO PROGRAMA

- Acesse uma página confiável para download ou o próprio site oficial do programa pelo link [AQUI](#).
- Procure pela opção “Download and Install”, ou vá diretamente nesse link [AQUI](#).
- Pressione o botão “Gusek Download” como indicado com a seta abaixo:

Download and Install

The GUSEK Project content is hosted at sourceforge.net/projects/gusek. The source can be reached by SVN: <https://gusek.svn.sourceforge.net/svnroot/gusek/trunk>.

To install GUSEK:

- Download the latest release from sourceforge: 
- Unzip it to any folder, at your option (GUSEK is a portable application)
- Open it running the `gusek.exe` file

Notes: GUSEK can be added to Windows Explorer contextual menu and associated to GMPL model extensions. See [Tools > Windows Integration](#) for options. Warnings: some Windows editions may require GUSEK running with Administrator rights to do the integration. If you move the GUSEK folder you need select this options again to register the new path.

Quick Guide

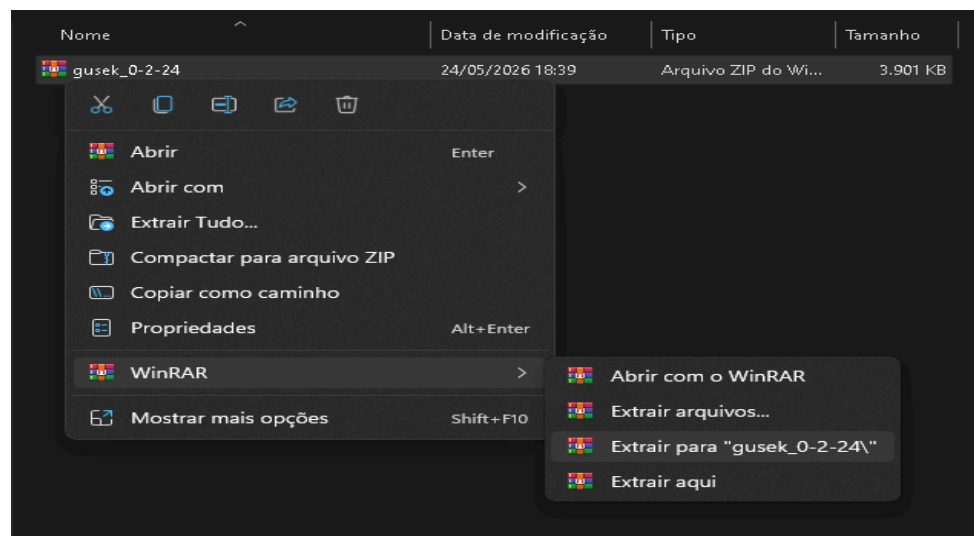
User Interface

GUSEK can open multiple files simultaneously, using file extensions to identify them:

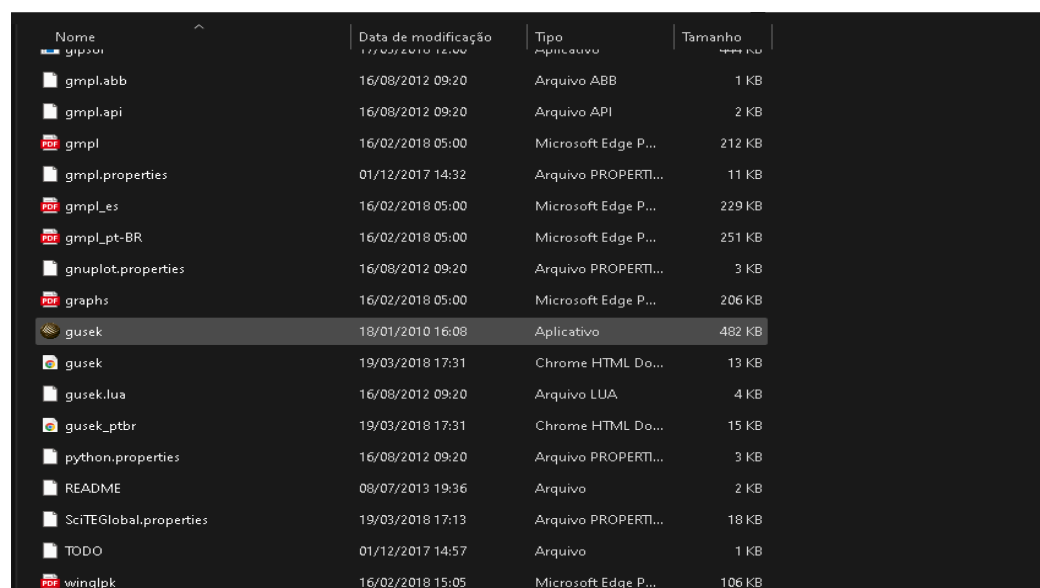
- `.mod`: GMPL Model file (can contain problem data too);
- `.dat`: GMPL Data file;
- `.out`: GLPK Output file;

2.3. INSTALAÇÃO DO PROGRAMA

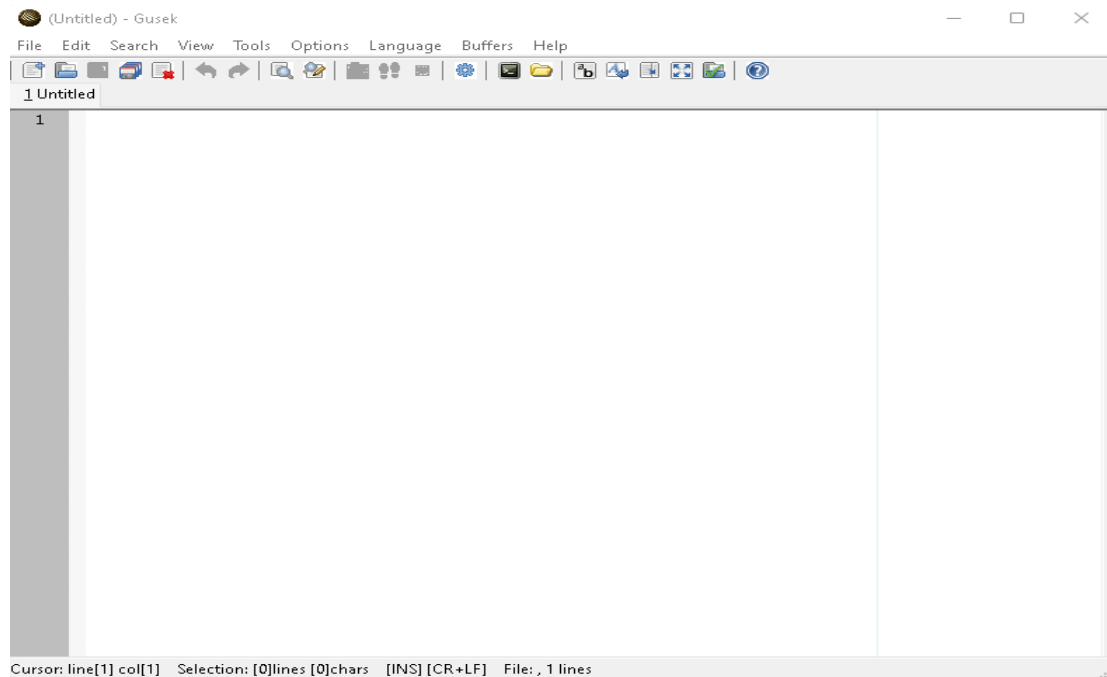
- Procure pelo arquivo .RAR que foi baixado para o seu computador, que provavelmente se encontra na pasta “Downloads”.
- Clique sobre o arquivo com o botão direito do mouse, e selecione a opção para descompactar o arquivo:



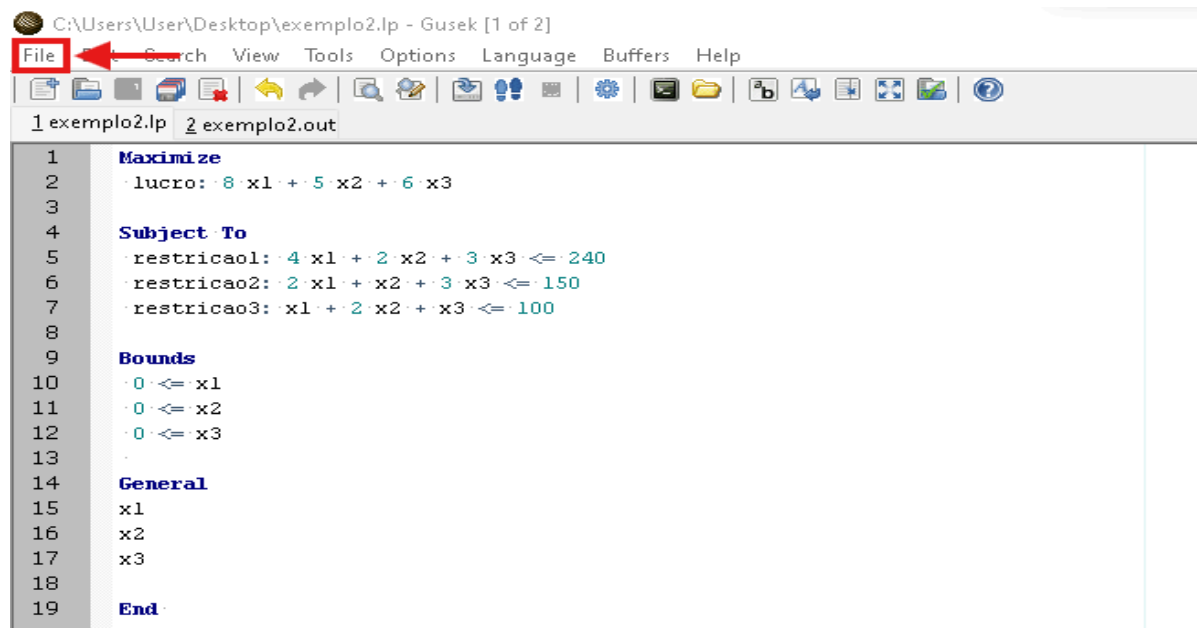
- Após descompactado, abra a pasta com os arquivos chamada “gusek”, e procure pelo arquivo chamado “gusek.exe”, e execute o programa:



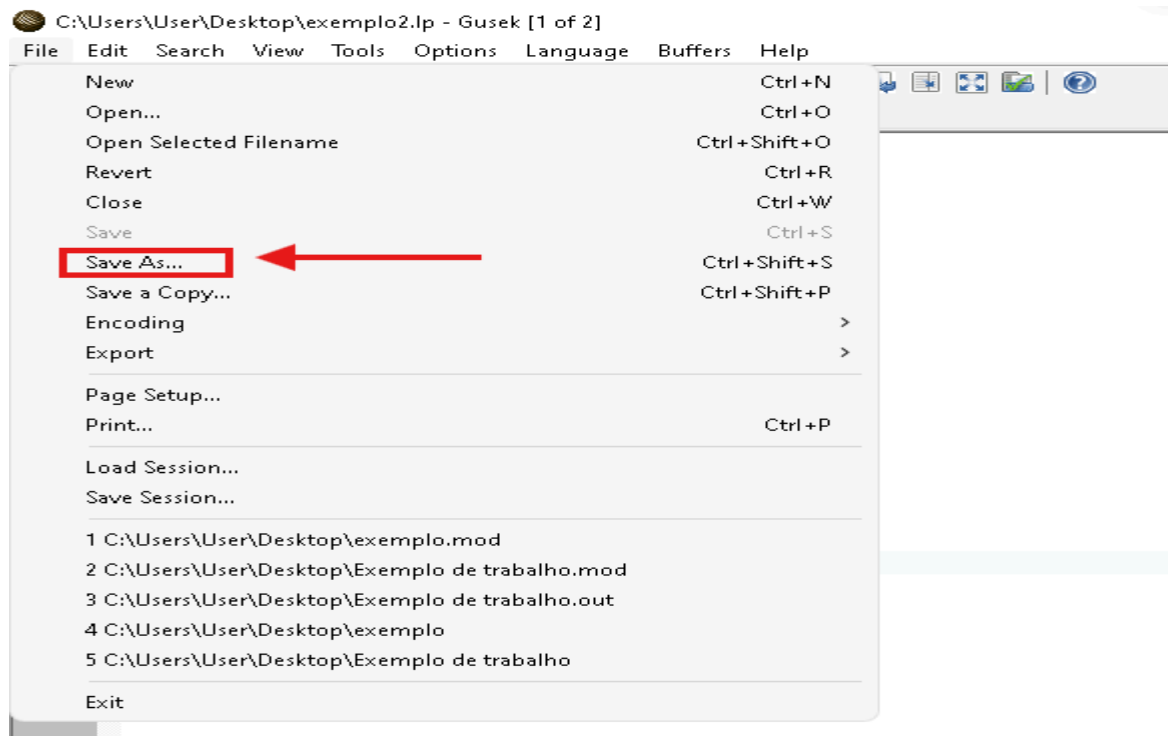
3. INTERFACE DA FERRAMENTA E PRÉ-CONFIGURAÇÕES



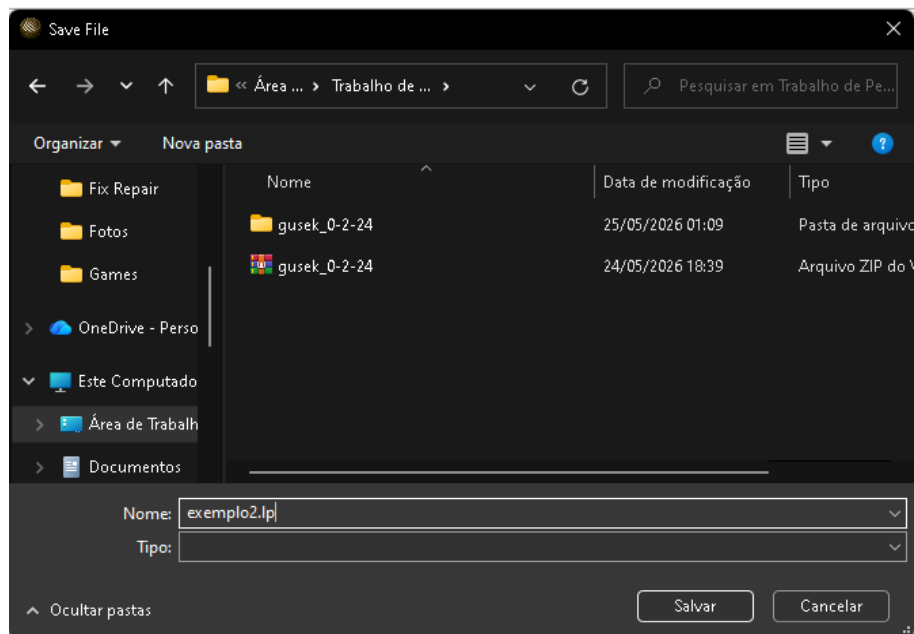
Aqui temos a interface do programa demonstrada. Após aberto o programa, podemos digitar nosso código contendo nosso problema a ser resolvido, e após completo, pressionamos Ctrl + S, ou vamos na função de salvar o programa atual para definir o formato em que ele será executado.



Clicamos primeiro em “File” como mostrado acima.

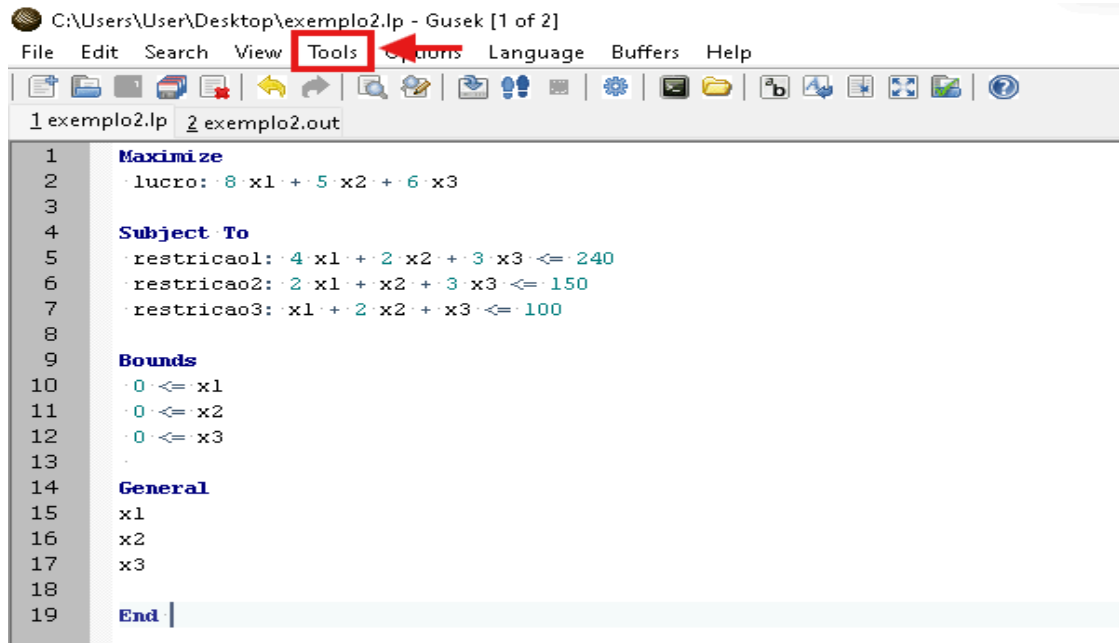


Agora clicamos em “Save As...”



Podemos nomear o arquivo com qualquer nome de sua preferência, e no final colocamos “.lp”, e aí sim salvamos o arquivo para depois executa-lo.

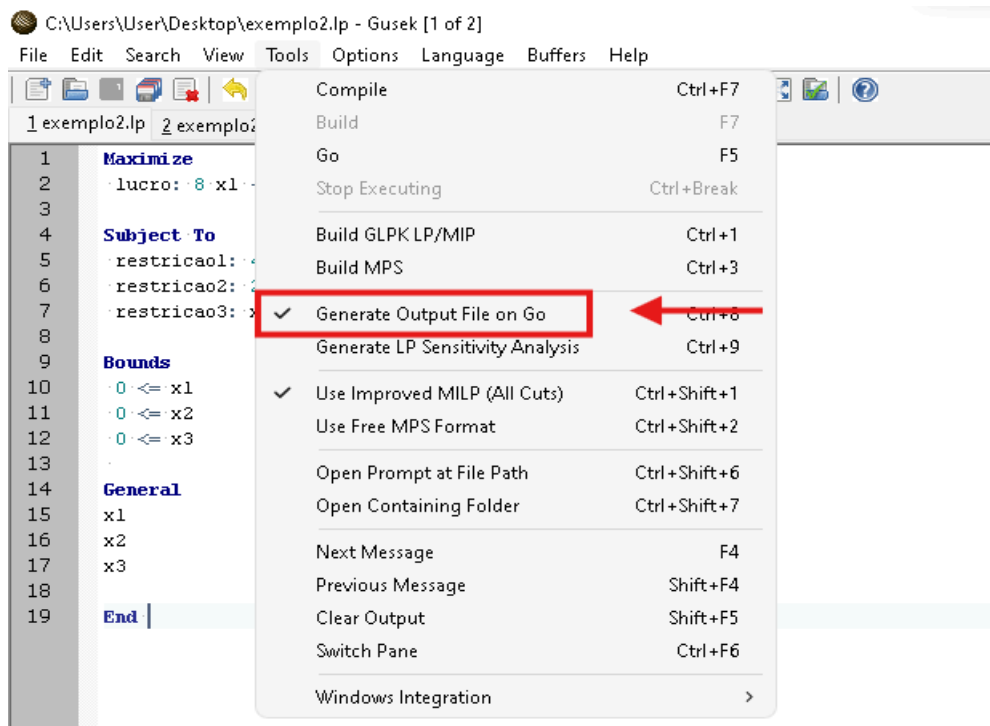
Por fim, vamos ativar uma última função que serve para mostrar os resultados em forma de tabela, gerando um novo arquivo, facilitando nossa interpretação dos resultados.



The screenshot shows the GUSEK software interface. The title bar reads "C:\Users\User\Desktop\exemplo2.lp - Gusek [1 of 2]". The menu bar includes File, Edit, Search, View, Tools, Options, Language, Buffers, and Help. The "Tools" menu is highlighted with a red box, and a red arrow points to it from the right. Below the menu bar is a toolbar with various icons. The main window displays a linear programming problem in a text editor. The problem is as follows:

```
1 Maximize
2   lucro: 8 x1 + 5 x2 + 6 x3
3
4 Subject To
5   restricao1: 4 x1 + 2 x2 + 3 x3 <= 240
6   restricao2: 2 x1 + x2 + 3 x3 <= 150
7   restricao3: x1 + 2 x2 + x3 <= 100
8
9 Bounds
10  0 <= x1
11  0 <= x2
12  0 <= x3
13
14 General
15  x1
16  x2
17  x3
18
19 End
```

Para isso, precisamos ir na opção “Tools” localizada na parte superior da interface do programa, como indicado pela seta vermelha acima.



E mantemos ativa essa opção mostrada acima, “Generate Output File on Go”. Dessa forma, ao executar o programa, ele criará na mesma pasta em que foi salva, outro arquivo com o resultado dos cálculos.

Abaixo no próximo tópico mostraremos a forma de utilização das demais funções do sistema junto com a montagem de um problema/exemplo de programação linear para solução.

4. DEFINIÇÃO DOS ELEMENTOS

4.1. VARIÁVEIS DE DECISÃO

As variáveis de decisão de um problema, apresentam os valores que o modelo precisa para calcular e encontrar as melhores soluções possíveis. Em problemas de programação linear, essas variáveis normalmente representam quantidades de produtos, recursos, transportes, investimentos ou qualquer outro elemento que possa ser ajustado durante a resolução do problema.

No formato (.lp) utilizado pelo Gusek e pelo solver GLPK, as variáveis não precisam ser declaradas previamente. Elas são identificadas automaticamente pelo sistema ao serem utilizadas na função objetivo ou nas restrições do modelo.

4.2. FUNÇÃO OBJETIVO.

A função objetivo nada mais é do que aquilo que desejamos otimizar dentro do problema. Numa programação linear, normalmente buscamos:

- Maximizar lucros;
- Diminuir custos ;
- Reduzir tempo;
- Aumentar produção;
- Em geral, otimizar recursos.

No formato .lp a função objetivo pode ser definida utilizando as palavras-chave “Minimize” ou “Maximize”, assim como no exemplo abaixo:

```
1 Maximize
2 lucro: 8 x1 + 5 x2 + 6 x3
3
```

Nesse caso, o modelo busca encontrar os melhores valores para as variáveis, de modo que o lucro total seja maximizado. O mesmo será feito utilizando “Minimize”, porém ao contrário, ele buscará os melhores valores para as variáveis de forma que o “Lucro” seja minimizado. Nesse contexto, poderíamos substituir o Lucro por Custo, aí sim o Custo seria reduzido:

```
1 Minimize
2 custo: 8 x1 + 5 x2 + 6 x3
3
```

O espaço que vemos entre os valores e as variáveis simboliza a multiplicação, não necessitando do uso de “*” para interpretação no cálculo.

4.3. RESTRIÇÕES

As restrições representam os limites e condições que devem ser respeitados durante a resolução do problema. Elas impedem que o modelo apresente soluções inviáveis dentro do contexto as quais inserimos o problema.

As restrições podem representar coisas como:

- Limite de matéria-prima;
- Capacidade de produção;
- Orçamento disponível;
- Demanda mínima;
- Limite de transporte;
- Horas de trabalho.

No formato .lp, as restrições são inseridas após o comando “Subject To”, como mostramos no exemplo abaixo:

```

3
4 Subject To
5 ·r1: 4·x1 + 2·x2 + 3·x3 ≤ 240
6 ·r2: 2·x1 + x2 + 3·x3 ≤ 150
7 ·r3: x1 + 2·x2 + x3 ≤ 100
8

```

Aqui, após o Subject To, usamos r1, r2 e r3 apenas para nomear cada restrição, já que precisam de identificação. O espaço entre os valores e as variáveis simboliza a multiplicação, assim como foi explicado na definição da função objetivo. Sinais de igualdade também são utilizados para definir as restrições, como menor que (<), maior que (>) e o igual (=). Também os utilizamos como no exemplo, menor ou igual à (<=).

4.4. TIPOS DE VARIÁVEIS

No Gusek, temos basicamente três tipos de variáveis que podemos utilizar, elas são as contínuas, as inteiras e as binárias. Explicamos cada uma delas abaixo:

4.4.1. VARIÁVEIS CONTÍNUAS

As variáveis contínuas são as que podem assumir valores decimais e fracionários, e podem ser definidas no programa como mostramos no exemplo abaixo:

```

8
9 Bounds
10 ·0 ≤ x1
11 ·0 ≤ x2
12 ·0 ≤ x3
13

```

Nesse caso, as variáveis citadas podem assumir qualquer valor real positivo. Muito utilizado em situações que as variáveis tratam peso, distância, volume, tempo, etc...

4.4.2. VARIÁVEIS INTEIRAS

As variáveis inteiras, como diz o próprio nome, assumem apenas números inteiros. No formato .lp, elas são definidas na seção General, como mostramos abaixo:

13	.
14	General
15	x1
16	x2
17	

Nesse exemplo, as variáveis x1 e x2 vão ter somente valores inteiros nos resultados obtidos. Utilizamos muito para tratar quantidade de objetos, como carro por exemplo, máquinas, produtos unitários, etc...

4.4.3. VARIÁVEIS BINÁRIAS

As variáveis binárias assumem apenas os valores 0 e 1, e no formato .lp elas são definidas na seção Binary, como no exemplo abaixo:

17	
18	Binary
19	x3
20	

As variáveis binárias são utilizadas em decisões como sim ou não, verdadeiro e falso, ativo ou inativo, etc...

5. EXEMPLO DIDÁTICO SIMPLES

No exemplo que daremos, vamos utilizar um problema bem simples de uma empresa. Essa empresa produz três produtos com madeira, mesas, cadeiras e estantes. O lucro delas por peça, respectivamente é R\$8,00, R\$5,00 e R\$6,00. O objetivo da empresa é aumentar a lucratividade na venda desses produtos, seguindo as restrições que definem as horas semanais de trabalho, e os materiais

usados para montagem de cada produto. Abaixo deixamos a função objetivo, e a tabela de relação dos produtos, materiais, horas de trabalho e demais informações:

$$\text{Função: } Z \text{ (maximizar)} = 8x_1 + 5x_2 + 6x_3$$

Produto	Lucro	Madeira	Pintura	Montagem
Mesa	R\$8,00	4	2	1
Cadeira	R\$5,00	2	1	1
Estante	R\$6,00	3	3	1
Horas Semanais	-	240h	150h	100h

Seguindo nossa tabela, temos as seguintes restrições:

$$4x_1 + 2x_2 + 3x_3 \leq 240 \text{ (Horas de madeira)}$$

$$2x_1 + 1x_2 + 3x_3 \leq 150 \text{ (Horas de pintura)}$$

$$1x_1 + 2x_2 + 1x_3 \leq 100 \text{ (Horas de montagem)}$$

$$x_1, x_2, x_3 \geq 0 \text{ (Variáveis não podem ser negativas)}$$

6. RESOLUÇÃO NA FERRAMENTA

Utilizando a ferramenta para resolver o problema proposto no tópico anterior, podemos inserir as informações no Gusek da seguinte forma mostrada a seguir:

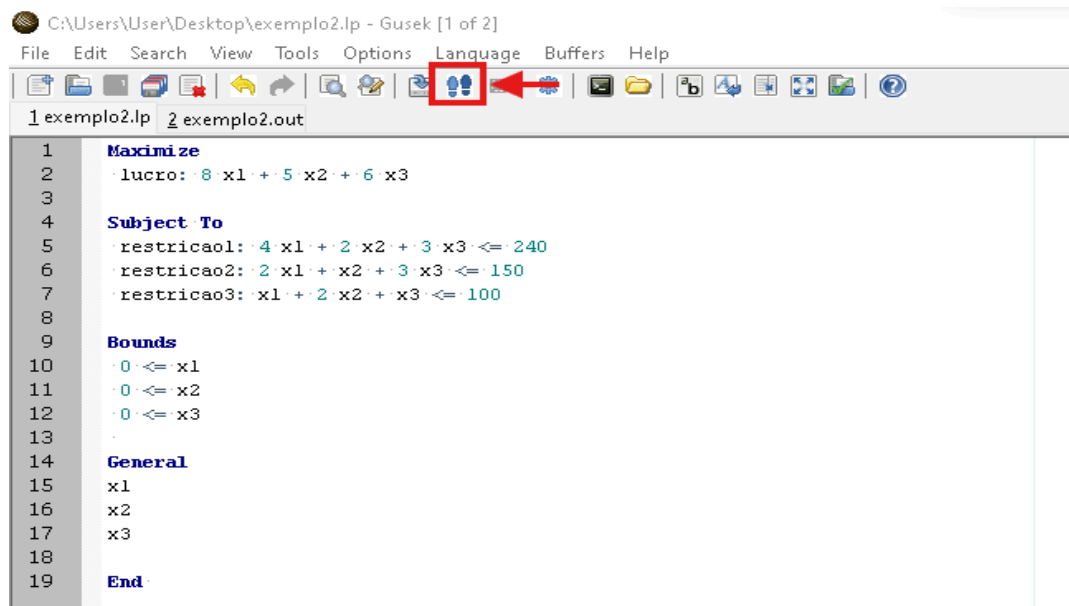
```

1  Maximize
2  ·lucro: ·8·x1·+·5·x2·+·6·x3
3
4  Subject To
5  ·restricao1: ·4·x1·+·2·x2·+·3·x3·<=·240
6  ·restricao2: ·2·x1·+·x2·+·3·x3·<=·150
7  ·restricao3: ·x1·+·2·x2·+·x3·<=·100
8
9  Bounds
10 ·0·<=·x1
11 ·0·<=·x2
12 ·0·<=·x3
13
14 General
15 x1
16 x2
17 x3
18
19 End |

```

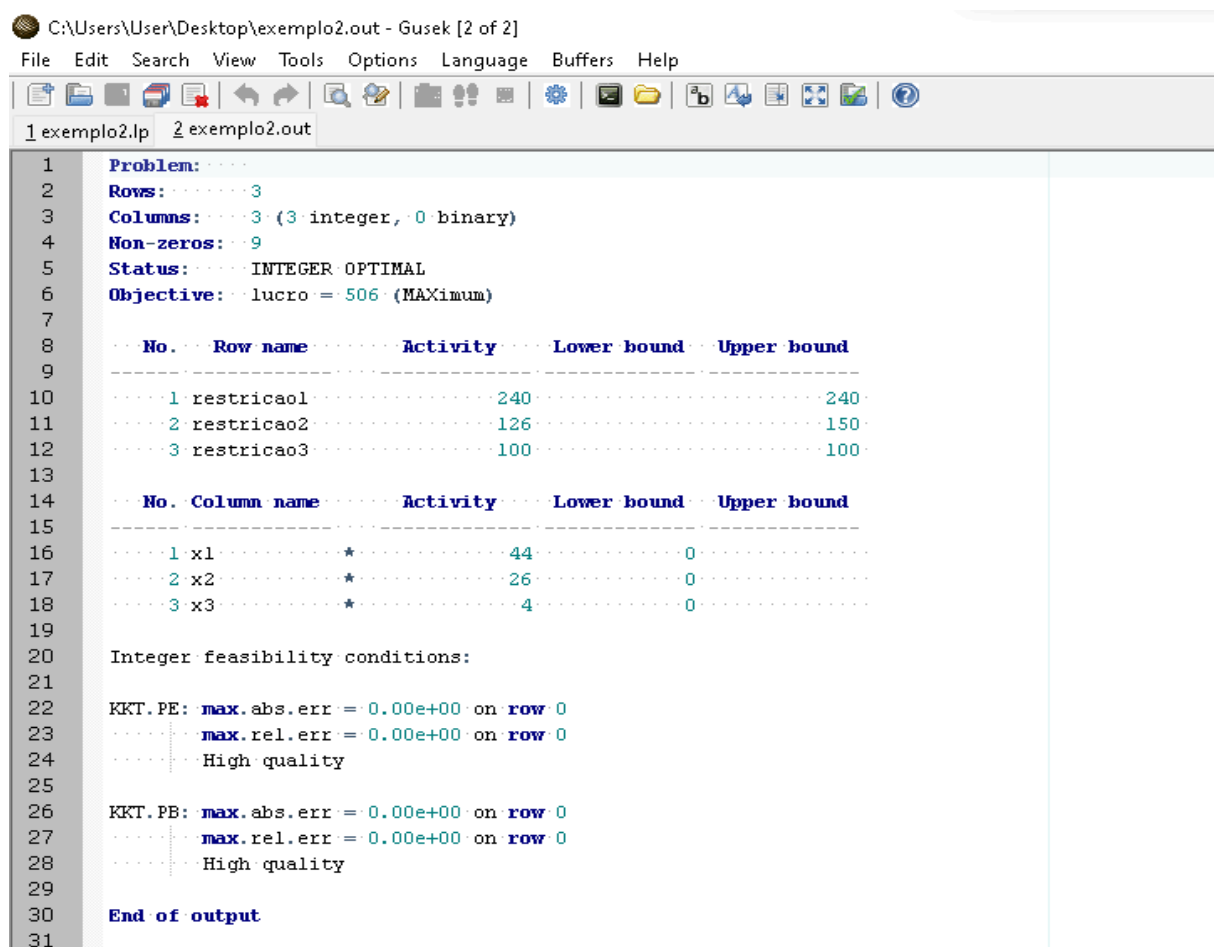
Aqui temos a maximização com a seção “Maximize” seguido da função objetivo. Na seção “Subject To” temos nossas três restrições que tratam a quantidade de material utilizado para produção de cada item, e o custo de horas de trabalho. Em “Bounds” garantimos que os valores finais das variáveis sejam maiores ou igual a zero. E por fim, a seção “General” nos permite pedir para que o programa entregue o valor de cada item a ser produzido como valor inteiro, já que se trata de objetos unitários.

Para que possamos executar esse código pronto agora, basta pressionarmos a tecla F5, ou clicar sobre o ícone de duas pegadas de sapato.



```
1 Maximize
2   lucro: 8 x1 + 5 x2 + 6 x3
3
4 Subject To
5   restricao1: 4 x1 + 2 x2 + 3 x3 <= 240
6   restricao2: 2 x1 + x2 + 3 x3 <= 150
7   restricao3: x1 + 2 x2 + x3 <= 100
8
9 Bounds
10  0 <= x1
11  0 <= x2
12  0 <= x3
13
14 General
15  x1
16  x2
17  x3
18
19 End
```

O resultado que temos rodando o programa é o seguinte:



```
1 Problem: .....
2 Rows: ..... 3
3 Columns: ..... 3 (3 integer, 0 binary)
4 Non-zeros: ..... 9
5 Status: ..... INTEGER OPTIMAL
6 Objective: ..... lucro = 506 (MAXimum)
7
8 .. No. .. Row name ..... Activity ..... Lower bound ..... Upper bound
9 .....
10 ..... 1 restricao1 ..... 240 ..... 240 .....
11 ..... 2 restricao2 ..... 126 ..... 150 .....
12 ..... 3 restricao3 ..... 100 ..... 100 .....
13
14 .. No. Column name ..... Activity ..... Lower bound ..... Upper bound
15 .....
16 ..... 1 x1 ..... * ..... 44 ..... 0 .....
17 ..... 2 x2 ..... * ..... 26 ..... 0 .....
18 ..... 3 x3 ..... * ..... 4 ..... 0 .....
19
20 Integer feasibility conditions:
21
22 KKT.PE: max.abs.err = 0.00e+00 on row 0
23 ..... max.rel.err = 0.00e+00 on row 0
24 ..... High quality
25
26 KKT.PB: max.abs.err = 0.00e+00 on row 0
27 ..... max.rel.err = 0.00e+00 on row 0
28 ..... High quality
29
30 End of output
31
```

7. INTERPRETAÇÃO DE RESULTADOS

Na tela anterior podemos visualizar o programa entregando a quantidades de restrições encontradas no código, a quantidade colunas, e também nos informa que foi encontrado uma solução ótima com o resultado de “lucro = 506”, tudo na tela abaixo:

```
1 Problem: .....
2 Rows: ..... 3
3 Columns: ..... 3 (3 integer, 0 binary)
4 Non-zeros: .. 9
5 Status: ..... INTEGER OPTIMAL
6 Objective: .. lucro = 506 (MAXimum)
7
```

A seguir, mostra as horas máximas que poderiam ser utilizadas, que foram definidas por cada uma das três restrições, e nos indica quantas horas de cada uma será utilizada para essa solução ótima, onde na Restrição 1 foram utilizadas 240 horas das 240 que haviam disponíveis. Na Restrição 2, foram utilizadas 126 horas, das 150 que haviam disponíveis. E na Restrição 3 também foram utilizadas 100 horas das 100 que haviam disponíveis, como mostrado abaixo:

```
7
8 No. Row name Activity Lower bound Upper bound
9
10 1 restricao1 240 240
11 2 restricao2 126 150
12 3 restricao3 100 100
13
```

Em nossa última análise, vemos quanto de cada variável foi utilizada para a solução, ou ainda, quanto de cada produto será produzido para que a empresa obtenha o melhor lucro possível. Nesse caso, para o melhor lucro com as horas e materiais disponíveis, deverão ser produzidas 44 cadeiras, 26 mesas e 4 estantes para a melhor lucratividade da empresa. Mostramos o resultado a seguir na última tabela:

13					
14		No. Column name	Activity	Lower bound	Upper bound
15					
16		1 x1	*	44	0
17		2 x2	*	26	0
18		3 x3	*	4	0
19					

RESOLUÇÃO DE UM PROBLEMA COMPLEXO

1. DESCRIÇÃO DO PROBLEMA COMPLEXO

Um dos problemas mais clássicos dentro da pesquisa operacional, é o problema de transporte com programação linear, muito utilizado como exemplos e demonstrações dos métodos utilizados na programação linear. A seguir será apresentado esse problema, mas com as alterações necessárias para que ele se torne um problema mais complexo e atenda os requisitos solicitados.

Uma empresa do ramo de logística precisa distribuir mercadorias de três centros de produção para quatro centros de distribuição espalhados geograficamente. Enquanto os centros de produção apresentam capacidades máximas de envio por semana, os centros de distribuição possuem demandas mínimas que devem ser atendidas.

O objetivo da empresa é minimizar o custo total semanal de transporte, mas seguindo as seguintes diretrizes: a empresa possui uma limitação de transporte, quantidades mínimas de produtos que devem ser entregues, há rotas que possuem carga máxima que pode ser levada, os carregamentos devem acontecer em quantidades inteiras de lote e algumas rotas são obrigatórias por questões logísticas.

2. MODELAGEM MATEMÁTICA DO PROBLEMA

2.1. VARIÁVEIS DE DECISÃO

- x_1 = quantidade transportada da Fábrica 1 para o Centro A;
- x_2 = quantidade transportada da Fábrica 1 para o Centro B;

- x_3 = quantidade transportada da Fábrica 2 para o Centro C;
- x_4 = quantidade transportada da Fábrica 2 para o Centro D;
- x_5 = quantidade transportada da Fábrica 3 para o Centro A;
- x_6 = quantidade transportada da Fábrica 3 para o Centro C.

2.2. FUNÇÃO OBJETIVO

Minimizar $Z = 12x_1 + 15x_2 + 10x_3 + 18x_4 + 14x_5 + 11x_6$

Onde:

12 = custo da rota $F1 \rightarrow A$

15 = custo da rota $F1 \rightarrow B$

10 = custo da rota $F2 \rightarrow C$

18 = custo da rota $F2 \rightarrow D$

14 = custo da rota $F3 \rightarrow A$

11 = custo da rota $F3 \rightarrow C$

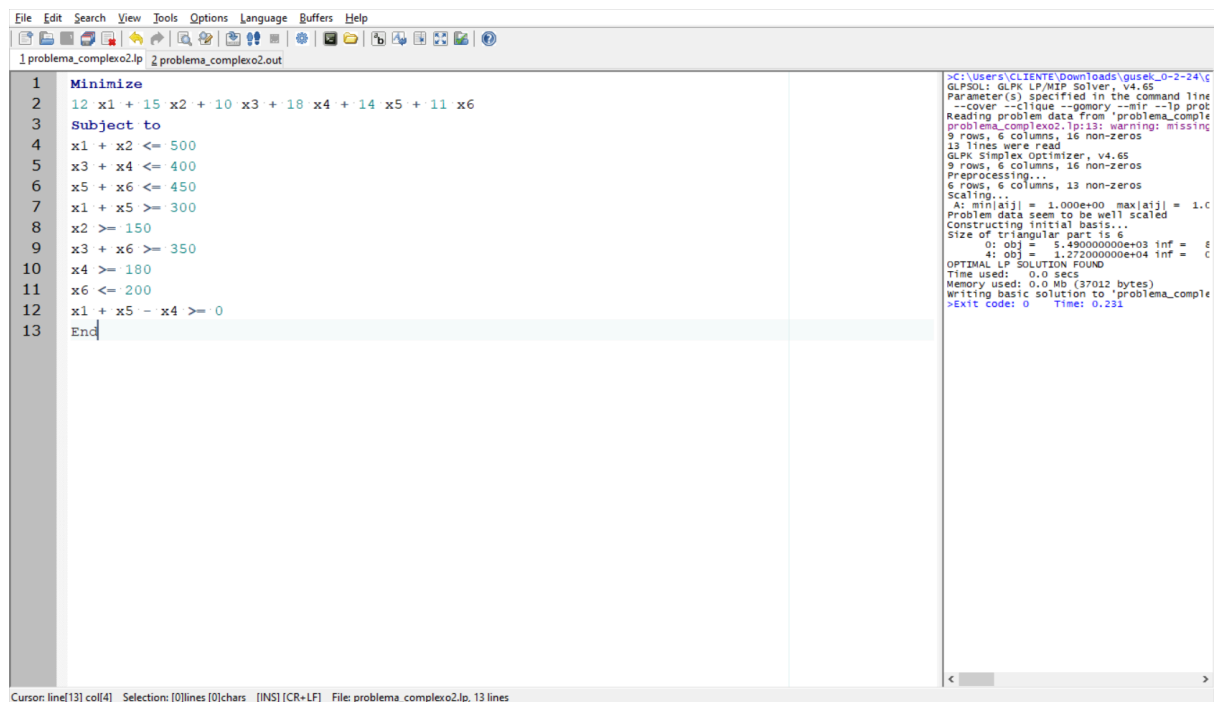
2.3. RESTRIÇÕES

- $x_1 + x_2 \leq 500$ - Restrição de capacidade da Fábrica 1
- $x_3 + x_4 \leq 400$ - Restrição de capacidade da Fábrica 2
- $x_5 + x_6 \leq 450$ - Restrição de capacidade da Fábrica 3
- $x_1 + x_5 \geq 300$ - Restrição de demanda do Centro A
- $x_2 \geq 150$ - Restrição de demanda do Centro B
- $x_3 + x_6 \geq 350$ - Restrição de demanda do Centro C
- $x_4 \geq 180$ - Restrição de demanda do Centro D
- $x_6 \leq 200$ - Restrição logística especial
- $x_1 + x_5 - x_4 \geq 0$ - Restrição estratégica
- $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$ - Restrição de não negatividade

2.4. TIPO DAS VARIÁVEIS

Todas as variáveis são inteiras e não negativas, isso porque, só podem assumir valores inteiros e não podem assumir valores negativos, já que elas representam quantidades físicas de produtos transportados.

3. IMPLEMENTAÇÃO NA FERRAMENTA



```
File Edit Search View Tools Options Language Buffers Help
1 problema_complexo2.lp 2 problema_complexo2.out

1 Minimize
2 12 x1 + 15 x2 + 10 x3 + 18 x4 + 14 x5 + 11 x6
3 Subject to
4 x1 + x2 <= 500
5 x3 + x4 <= 400
6 x5 + x6 <= 450
7 x1 + x5 >= 300
8 x2 >= 150
9 x3 + x6 >= 350
10 x4 >= 180
11 x6 <= 200
12 x1 + x5 - x4 >= 0
13 End
```

```
>C:\Users\CLIENTE\Downloads\gusek_0-2-24\g
GLPSOL: GLPK LP/MIP Solver, v4.65
Parameter(s) specified in the command line
--cover --clique --memory --nir --lp prot
Reading problem data from 'problema_comple
problema_complexo2.lp:13: warning: missing
9 rows, 6 columns, 16 non-zeros
13 lines were read
GLPK Simplex Optimizer, v4.65
9 rows, 6 columns, 16 non-zeros
Preprocessing...
6 rows, 6 columns, 13 non-zeros
Scaling...:
A: min|a[ij]| = 1.000e+00 max|a[ij]| = 1.0
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 6
0: obj = 5.490000000e+03 inf = 8
4: obj = 1.272000000e+04 inf = 0
OPTIMAL LP SOLUTION FOUND
Time used: 0.0 secs
Memory used: 0.0 Mb (37012 bytes)
Writing basic solution to 'problema_comple
>Exit code: 0 Time: 0.231
```

O primeiro passo para a implementação na ferramenta é colocar qual o tipo da função a ser resolvida, no caso do problema apresentado o objetivo da função é minimizar, então basta apenas digitar “Minimize”. Se a função fosse para maximizar, bastaria apenas digitar “Maximize”. Logo após colocar o objetivo da função, é necessário digitar a função-objetivo, porém os valores das variáveis, a nomenclatura das variáveis e as operações devem estar espaçadas, como é mostrado na imagem acima. Após isso, é necessário digitar “Subject to” para inserir as restrições do problema, e então é só digitar as restrições normalmente, mas seguindo a mesma diretriz dos espaçamentos. Ao final, depois que a função e as restrições estiverem completas na ferramenta, é só digitar a palavra “End” que o software entende que a função acabou ali.

```

File Edit Search View Tools Options Language Buffers Help
1 problema_complexo2.lp 2 problema_complexo2.out

1 Problem:
2 Rows: 9
3 Columns: 6
4 Non-zeros: 16
5 Status: OPTIMAL
6 Objective: obj = 12720 (MINimum)
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

```

No.	Row name	St	Activity	Lower bound	Upper bound	Marginal
1	r.4	B	450		500	
2	r.5	NU	400		400	-1
3	r.6	B	130		450	
4	r.7	NL	300	300		12
5	r.8	NL	150	150		15
6	r.9	NL	350	350		11
7	r.10	NL	180	180		19
8	r.11	B	130		200	
9	r.12	B	120	0		

No.	Column name	St	Activity	Lower bound	Upper bound	Marginal
1	x1	B	300	0		
2	x2	B	150	0		
3	x3	B	220	0		
4	x4	B	180	0		
5	x5	NL	0	0		2
6	x6	B	130	0		

```

Karush-Kuhn-Tucker optimality conditions:

```

```

>C:\Users\CLIENTE\Downloads\gusek_0-2-24\G
GLPSOL: GLPK LP/MIP Solver, v4.65
Parameter(s) specified in the command line
--cover --clique --gomory --mtr --lp prot
Reading problem data from 'problema_comple
problema_complexo2.lp:13: warning: missing
9 rows, 6 columns, 16 non-zeros
13 lines were read
GLPK Simplex Optimizer, v4.65
9 rows, 6 columns, 16 non-zeros
Preprocessing...
6 rows, 6 columns, 13 non-zeros
Scaling...
A: min|a[ij]| = 1.000e+00 max|a[ij]| = 1.0
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 6
0: obj = 5.480000000e+03 inf = 0
4: obj = 1.272000000e+04 inf = 0
OPTIMAL LP SOLUTION FOUND
Time used: 0.0 secs
Memory used: 0.0 Mb (37012 bytes)
Writing basic solution to 'problema_comple
>Exit code: 0 Time: 0.231

```

Cursor: line[1] col[1] Selection: [0]lines [0]chars [INS] [CR+LF] File: problema_complexo2.out, 48 lines

O arquivo em que o problema/função foi descrito deve ser salvo com a extensão “.lp” ao final, e após rodar esse “código” em “Go”, será criado, e executado logo em seguida, um novo arquivo com formato “out” que será a saída do problema, ou seja, a sua resolução. Nesse arquivo é possível visualizar as respostas ao problema descrito. Logo nas primeiras linhas, é possível visualizar em “Status” que está como “OPTIMAL”, que significa que a solução ótima foi encontrada, e logo abaixo do status está o resultado obtido da função já minimizada, que seria $Z_{min} = 12720$.

Abaixo dessas primeiras informações, o Gusek apresenta duas tabelas. Na segunda tabela são mostrados os resultados das variáveis de decisão. Então, a solução ótima para esse problema é:

- $x_1 = 300$
- $x_2 = 150$
- $x_3 = 220$
- $x_4 = 180$
- $x_5 = 0$
- $x_6 = 130$

4. RESULTADOS OBTIDOS

Substituindo os valores das variáveis de decisão na função-objetivo e realizando o cálculo, é possível ver que o resultado que será obtido é o mesmo apresentado no software:

$$Z_{\min} = 12(300) + 15(150) + 10(220) + 18(180) + 14(0) + 11(130)$$

$$Z_{\min} = 3600 + 2250 + 2200 + 3240 + 0 + 1430$$

$$Z_{\min} = 12720$$

Com isso, pode-se entender que a melhor solução para o problema da empresa é:

Fábrica 1 → Centro A $x_1 \rightarrow 300$

Fábrica 1 → Centro B $x_2 \rightarrow 150$

Fábrica 2 → Centro C $x_3 \rightarrow 220$

Fábrica 2 → Centro D $x_4 \rightarrow 180$

Fábrica 3 → Centro A $x_5 \rightarrow 0$

Fábrica 3 → Centro C $x_6 \rightarrow 130$

Essas são as quantidades ótimas que cada fábrica pode transportar para cada centro, com o custo mínimo total de R\$12.720 semanais.

5. DISCUSSÃO E ANÁLISE

O software GUSEK mostrou ser uma ferramenta bem intuitiva, fácil de aprender/usar e também muito importante e eficaz para a resolução de problemas relacionados à programação linear e otimização matemática. Ele se mostra muito útil principalmente para a resolução de problemas mais complexos e em contextos reais, que possuam muitas variáveis de decisão e valores muito altos.

CONCLUSÃO

A utilização do software GUSEK permitiu modelar e resolver um problema complexo de programação linear, sobre o transporte e distribuição de produtos. Através dele foi possível representar corretamente a função-objetivo, as variáveis de decisão e as restrições necessárias, possibilitando analisar a minimização do problema apresentado. O problema apresentou características bem próximas de um contexto real, levando em consideração uma empresa que deseja transportar seus produtos até um outro centro de distribuição, mas busca minimizar os custos totais dos transportes semanais, buscando sempre a melhor rota para cada ponto.

Com isso, o GUSEK demonstrou ser uma ferramenta muito útil e eficiente na aplicação real de soluções matemáticas e também na aplicação dos conceitos de pesquisa operacional, tornando-a uma ótima ferramenta didática.

REFERÊNCIAS

GUSEK. Disponível em: <https://gusek.sourceforge.net/gusek_ptbr.html>. Acesso em: 26 maio. 2026.

ALEXANDRE CHECOLI. **Treinamento GUSEK**. Disponível em: <<https://www.youtube.com/watch?v=DwK9ctCn1e0>>. Acesso em: 26 maio. 2026.