FACULDADE DE TECNOLOGIA DE SÃO PAULO

MARÍLIA VALÉRIO ROCHA

Pesquisa Operacional com auxílio do IBM ILOG CPLEX

SÃO PAULO
2023

FACULDADE DE TECNOLOGIA DE SÃO PAULO

MARÍLIA VALÉRIO ROCHA

Pesquisa Operacional com auxílio do IBM ILOG CPLEX

Trabalho submetido como exigência parcial
para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas

Orientador: Professor Mestre Valter Yogui

SÃO PAULO

2023

FACULDADE DE TECNOLOGIA DE SÃO PAULO

MARILIA VALÉRIO ROCHA

Pesquisa Operacional com auxílio do IBM ILOG CPLEX

Trabalho submetido como exigência parcial para obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador

_____

_____

_____

Conceito/Nota Final: _____

Atesto o conteúdo contido na postagem do ambiente TEAMS pelo aluno e assinado por mim para avaliação do TCC.

Orientador: Professor Mestre Valter Yogui

SÃO PAULO, 12 de junho de 2023.

Assinatura do Orientador

Assinatura do aluno

EPÍGRAFE

A finalidade de toda informação é reduzir o grau de incerteza que a decisão implica (ANDRADE, 2009, p.8).

RESUMO

O objetivo deste estudo é apresentar *o software IBM CPLEX Optimizer* (CPLEX), como uma alternativa para aproximar o estudo da Pesquisa Operacional, em particular da programação linear, aos conhecimentos de programação, adquiridos no curso de Análise e Desenvolvimento de Sistemas. Foram apresentados sete problemas típicos e, implementados e discutidos, dez exemplos com o auxílio do *CPLEX*. Os exemplos foram também desenvolvidos com o auxílio do solver, presente no Excel. A análise das possibilidades deste software, feita a partir dos algoritmos desenvolvidos nos exemplos, permite afirmar que foram usados conceitos desenvolvidos nas disciplinas de programação, entre eles, a definição de variáveis, criação de funções, uso de laços e formatação de resultados. Esta análise permite considerar que essa abordagem possibilita momentos de discussão sobre a modelagem de diferentes problemas de programação linear.

Palavras-chave: Pesquisa Operacional, IBM CPLEX, Programação Linear.

ABSTRACT

The objective of this study is to present the IBM CPLEX Optimizer software (CPLEX) as an alternative to bring the study of Operations Research, in particular linear programming, closer to the programming knowledge acquired in the Systems Analysis and Development course. Seven typical problems were presented and, implemented and discussed, ten examples with the help of CPLEX. The examples were also developed with the help of the solver, present in Excel. The analysis of the possibilities of this software, made from the algorithms developed in the examples, allows us to state that concepts developed in the programming disciplines were used, among them, the definition of variables, creation of functions, use of loops and formatting of results. This analysis allows us to consider that this approach allows moments of discussion about the modeling of different linear programming problems.

Palavras-chave: Operations Research, IBM CPLEX, Linear Programming.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, o aumento da capacidade e a velocidade de processamento de informações pelos computadores, em conjunto com a disponibilização de softwares de resolução de problemas, permitiram o avanço da aplicação da Pesquisa Operacional (PO). A PO é um “método científico de tomada de decisões” (SILVA *et al.*, 1998, p.11). De acordo com Andrade (2009), um estudo de PO:

> [...] consiste, basicamente, na construção de um modelo para um sistema real que sirva como instrumento de análise e compreensão do comportamento desse sistema, com o objetivo de levar o sistema a apresentar o desempenho desejado (ANDRADE, 2009, p.9).

Na FATEC-SP, a ementa da disciplina Programação Linear e Aplicações, ministrada no curso de Análise e Desenvolvimento de Sistemas (ADS), apresenta tópicos da Pesquisa Operacional: Programação Linear - métodos gráfico e Simplex; e Aplicações - Método do Transporte (Anexo A).

Em situações reais, os modelos matemáticos de programação linear contam com um número grande de variáveis, trazendo complexidade em sua resolução manual, pois podem necessitar de um número considerável de iterações para se obter uma solução. Em geral, após a modelagem do problema, é apresentado o método Simplex para resolução e obtenção da solução ótima. Após alguns exemplos e exercícios, percebe-se que a resolução manual, ou mesmo com o auxílio de planilha eletrônica, é repetitiva, sobretudo, quando há várias variáveis de decisão ou são necessárias várias iterações para obter o resultado. Uma alternativa para resolução é o uso do Solver, disponível como suplemento do Excel, que apresenta a solução e também é usado na disciplina. Prado (2016) cita uma consolidação dos principais softwares disponíveis no mercado para resolução de problemas de programação linear (Tabela *1*).

Na educação, em particular nos cursos de ADS em que a disciplina é ministrada no 4º semestre, os alunos possuem conhecimentos de programação adquiridos em disciplinas ministradas nos semestres anteriores (Algoritmos e Lógica de Programação, Linguagem de Programação e Estruturas de Dados).

Buscando alternativas para empregar os conceitos adquiridos nas disciplinas citadas na resolução de problemas de PO, observou-se uma alternativa disponível no mercado: o software *IBM CPLEX Optimizer* (CPLEX). A partir do tutorial de Ferreira

(2017) identificou-se que essa solução atendia ao objetivo de aproximar os conhecimentos de programação aos tópicos da PO.

Tabela 1 - Softwares

Software	Fornecedor	Capacidade		Programação Inteira		Formato para entrada de dados	
		Restrições (linhas)	Variáveis (colunas)	binária	geral	Planilha	linguagem de programação
LINDO/LINGO	Lindo	ilimitado	ilimitado	sim	sim	sim	sim
CPLEX	IBM	50.000	100.000	sim	sim	sim	sim
Solver-Excel	Microsoft	100	200	sim	sim	sim	
Solver-Premium	Frontline	ilimitado	1000	sim	sim	sim	

Fonte: adaptado de Prado (2016, p.57)

De acordo com a desenvolvedora, o CPLEX é “um resolvedor de programação matemática de alto desempenho para programação linear, programação inteira mista e programação quadrática” (IBM). Há uma versão acadêmica gratuita para universidades cadastradas. É possível também obter uma licença para teste, após o cadastro no site https://www.ibm.com/br-pt/analytics/cplex-optimizer.

O diferencial do CPLEX é permitir abstrair o modelo para diferentes notações (expandida, matricial e com uso de somatório) e possibilitar a construção de um algoritmo para a resolução. Permite, também, a formatação do resultado com auxílio da linguagem *JavaScript*. A apresentação da solução obtida, usando *JavaScript*, possibilita integrar, também, o conteúdo ministrado em uma disciplina eletiva (Linguagem de Programação IV). Observou-se que o CPLEX favorece o uso de diferentes saberes na construção de uma solução para um problema de programação linear.

A proposta deste estudo de conclusão de curso é apresentar uma alternativa para aproximar o estudo da PO aos conhecimentos de programação, com auxílio do CPLEX. Reforça-se que o uso de um software não elimina a necessidade de se conhecer a resolução manual dos problemas, com poucas variáveis de decisão, permitindo a compreensão da dinâmica de resolução. Quando o software é inserido na proposta didática, pode contribuir para ampliar a discussão da modelagem de diferentes problemas.

Após a definição do software para resolução de problemas de PO, foi pesquisado o histórico do desenvolvimento da PO, apresentado no capítulo 2, no item

2.1 e os passos para se realizar uma PO, explanado no item 2.2. Buscou-se os problemas típicos apresentados nos livros didáticos e foram apresentados sete destes problemas no item 2.3. Os recursos do CPLEX foram explorados por meio da resolução de dez exemplos desses problemas típicos, com variações, apresentados no capítulo 3. Os exemplos foram resolvidos também por meio do solver, ferramenta disponível no Excel (Microsoft), com o intuito de validar os resultados obtidos com o uso do CPLEX (Apêndice B). Estas planilhas podem ser usadas para auxiliar na compreensão das notações utilizadas nos algoritmos. O resumo da instalação e uma apresentação geral do software constam no Apêndice A.

2 FUNDAMENTAÇÃO TEÓRICA

A PO, em particular, as técnicas de programação linear, a partir da segunda metade do séc. XX, vêm contribuindo para redução de custos nas empresas. Belfiore e Fávaro (2013, p.28) citam uma pesquisa, de 2004, das 500 maiores empresas americanas listadas pela revista Fortune. Nesta pesquisa, 85% dos entrevistados afirmaram que usavam ou já haviam usado a técnica de programação linear.

A PO "é aplicada a problemas que compreendem a condução e a coordenação de operações [atividades] em uma organização" (HILLIER; LIBERMANN, 2013, p. 13). Andrade (2009) afirma que a PO é "uma metodologia administrativa cujo arcabouço teórico agrega quatro ciências fundamentais para o processo de preparação, análise e tomada de decisão: a economia, a matemática, a estatística e a informática" (ANDRADE, 2009, p. IX), o que evidencia sua característica interdisciplinar.

Lachtermacher (2009) cita as seguintes áreas de aplicação de programação linear: administração da produção, análise de investimentos, alocação de recursos limitados, planejamento regional, logística, custo de transporte, localização da rede de distribuição e alocação de recursos de publicidade entre diversos meios de comunicação. Estas aplicações demonstram que a natureza da organização é uma questão secundária para problemas de PO.

A PO preocupa-se com a gestão da organização, procurando contribuir para tomada de decisão e tem apresentado melhoria na eficiência das organizações. A solução de um problema de PO procura indicar uma melhor solução à organização, ou solução ótima, ao modelo definido. "Um estudo de PO busca soluções que são ótimas para a organização como um todo em vez de soluções subotimizadas, que são boas apenas para um integrante" (HILLIER; LIBERMANN, 2013, p. 8), ou um departamento, em particular.

Andrade (2009) ressalta que a PO, por permitir o uso de modelos, facilita o processo de análise de decisão.

> Essa abordagem permite a 'experimentação', ou seja, a possibilidade de uma tomada de decisão ser mais bem avaliada e testada, antes de ser efetivamente implementada. Por si sós, a economia de recursos e a experiência adquirida com a experimentação justificam o conhecimento e a utilização da PO, como instrumento de gerência (ANDRADE, 2009, p.1).

Após uma breve explanação sobre a origem da pesquisa operacional, foram listadas as fases que caracterizam a PO, reforçando a natureza secundária da

organização, pois, uma vez entendido o problema e definida a modelagem cria-se um modelo de resolução, conhecido como modelo típico, que será abordado na sequência.

2.1 Origem da pesquisa operacional

A origem da pesquisa operacional (PO) remonta ao séc. XVI, mas na segunda guerra mundial sua importância foi ressaltada. Nesse período, o estudo da alocação dos recursos disponíveis e de problemas bélicos foram dados aos cientistas, pelos ingleses e norte-americanos e desempenhados com sucesso. Pizzolato e Gandolpho (2009) afirmam que

> a lista de projetos desenvolvidos na época é vasta, mas, examinando-a, é possível afirmar que a conjunção de dois fatores, identificação da essência do problema e busca de dados para compreender causas e efeitos, foi determinante na geração de soluções inteligentes e não antevistas pelos militares diretamente envolvidos nas atividades bélicas (PIZZOLATO E GANDOLPHO, 2009, p.2).

Em razão do sucesso, estabeleceu-se uma área de estudo denominada pesquisa operacional aplicada na área militar, expandindo-se à indústria, comércio e governo, no início dos anos 1950. Hillier e Lieberman (2013) atribuem esse sucesso ao progresso das técnicas da PO, entre elas, o método Simplex, programação linear, teoria das filas e teoria do inventário formalizadas nesse período. Acrescentam que a revolução computacional foi outro fator de impulsionamento da PO, que permitiu a rapidez na execução dos cálculos necessários e, mais recentemente, a partir da década de 1980, a disponibilização de pacotes de software para PO.

De acordo com Prado (2016, p.26), em 1936, Wassily Leontieff criou um modelo com equações lineares, que é considerado o início para o estabelecimento de técnicas de programação linear. Em 1939, L.V. Kantorovick publicou um trabalho sobre planejamento da produção, usando equações lineares. Em 1940, Frank L. Hitchcock apresentou um estudo do problema de transportes.

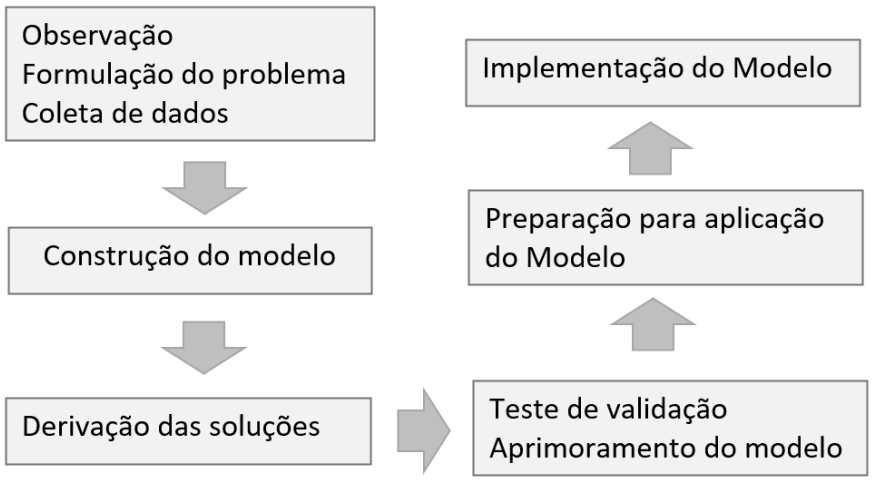
O método Simplex foi desenvolvido por George Dantizig, em 1947, quando trabalhou na Força Aérea Americana, no projeto SCOOP (*Scientific Computation of Optimum Programs*). Este método permite resolver qualquer problema de programação linear. No início, os cálculos eram manuais, mas nas décadas seguintes

os computadores passaram a contribuir com muitos cálculos necessários para se chegar a uma solução ótima.

2.2 Abordagem da modelagem dos problemas

De acordo com Hillier e Lieberman (2013), um estudo de PO apresenta as seguintes fases (Figura 1):

Figura 1 – Fases do estudo de PO



Fonte: a autora.

1. Observação, formulação do problema e coleta de dados:

Na análise do problema, busca-se um enunciado bem definido, que abranja os objetivos, restrições, relação entre áreas da organização envolvidas no problema, caminhos alternativos e limites de tempo. Ressalta-se a importância de listar objetivos que, quando idealmente formulados, são de toda a organização. “Os objetivos no estudo devem ser os mais específicos e, ao mesmo tempo, englobar os principais objetivos do tomador de decisões e manter um grau de consistência razoável com os mais altos objetivos” (HILLIER; LIBERMANN, 2013, p. 8).

Os pesquisadores realizam uma análise técnica do problema, propõem alternativas que contribuam na tomada de decisão. Estas alternativas são analisadas, posteriormente, pela gerência que considera aspectos tangíveis e intangíveis.

Nesta fase, levantam-se os dados que a empresa possui e os que seriam necessários para posterior implantação do modelo a ser definido na próxima fase. A análise dos dados disponíveis evidenciará os investimentos na melhoria destas

informações, como elaboração ou tratamento do banco de dados, integração das informações entre áreas e identificação e tratamento dos dados relevantes. Atualmente, nessa etapa, recorre-se às técnicas de *data mining*, para tratar os dados e buscar padrões de interesse.

Andrade (2009) apresenta a fase chamada de definição do problema e reforça a necessidade de descrição exata dos objetivos do estudo; identificação das alternativas de decisão existentes e reconhecimento das limitações, restrições e exigências do sistema.

2. Construção de um modelo matemático:

Após aprovação da gerência, parte-se para a construção de um modelo, que permita uma abstração da essência do problema real.

> Parte-se da hipótese de que esse modelo é uma representação suficientemente precisa das características essenciais da situação e de que as conclusões (soluções) obtidas do modelo também são válidas para o problema real (HILLIER; LIBERMANN, 2013, p. 2).

De modo geral, um modelo usado em PO é formulado, usando:

- Variáveis de decisão (x_1, x_2,..., x_n): são as n decisões quantificáveis relacionadas, cujos valores serão determinados.
- Função objetivo: função que expressa uma medida de desempenho e é expressa por meio das variáveis de decisão. É um objetivo a ser otimizado. O problema pode ter mais de uma função objetivo, uma para cada objetivo definido pela gerência e, nesse caso, cria-se uma medida global de desempenho, o que acrescenta complexidade ao processo. Neste texto, foram tratados problemas típicos com uma função objetivo.
- Restrições: expressas por desigualdades ou equações, indicam restrições a serem atendidas pelas variáveis de decisão.
- Parâmetros: são constantes que podem compor a função objetivo e as restrições. Na prática, esses parâmetros restritivos a serem utilizados são obtidos por meio de análise de dados relevantes, conhecida como análise de sensibilidade e podem ser aprimorados nos testes do modelo.

Usando a notação citada em precedência, nota-se que o problema do modelo matemático de PO é encontrar os valores das variáveis de decisão de modo a obter o melhor resultado da função objetivo sujeita às restrições específicas.

O modelo que descreve o problema é construído e, possivelmente, será aprimorado ao longo dos testes. Ressalta-se que o modelo torna compreensível a estrutura geral do problema, revela relacionamentos de causa-efeito e permite o uso de técnicas matemáticas para identificar as soluções. Em geral, inicia-se com um modelo em versão simples e, progressivamente, considerando-se mais variáveis e restrições, busca-se aproximá-lo da realidade.

Uma vez resolvido o primeiro modelo, busca-se a aderência dos resultados ao fenômeno real, ou seja, identifica-se a validade desse modelo. Para tanto, constrói-se testes e ajustes contínuos no modelo, buscando uma estabilidade. Este processo é feito na fase de construção do modelo, que é estabilizado à medida que não se agrega maior complexidade a ele. “É necessário haver uma correlação entre a previsão realizada pelo modelo e o que realmente acontece no mundo real. [...] O equilíbrio básico a ser sempre considerado é entre a *precisão* e a *tratabilidade* do modelo” (HILLIER; LIBERMANN, 2013, p. 10).

Andrade (2009) acrescenta que a fase de construção do modelo exige mais criatividade do analista.

3. Derivação de soluções com base no modelo com uso de recurso computacional:

Com um modelo estável, buscam-se alternativas computacionais para obter as soluções, pois a identificação de uma solução requer muitos cálculos que, se feitos manualmente, inviabilizariam sua implementação. Nesse ponto, buscam-se pacotes disponíveis no mercado para tratar problemas de PO.

Dentre as soluções possíveis do modelo, em geral, busca-se uma solução ótima, que poderá indicar o caminho a ser traçado na situação real. Uma tendência na área de PO é, considerando uma abordagem pragmática, observar não só uma solução ótima, mas aquela satisfatoriamente boa, criando objetivos com níveis mínimos de desempenho satisfatórios nas áreas da organização.

Nesta fase, após identificar uma solução ótima, passa-se para a análise de pós-otimalidade (ou análise e se), em que o modelo é estressado, ou seja, consideram-se

diferentes cenários futuros e avalia-se o comportamento da solução ótima. Em geral, estes cenários são levantados pelos administradores, que conhecem o negócio. Nesta fase, identificam-se os parâmetros sensíveis, aqueles “cujos valores não podem ser modificados sem se alterar a solução ótima” (HILLIER; LIBERMANN, 2013, p. 13). Os valores iniciais atribuídos aos parâmetros são estimados e passam por um refinamento nessa fase.

Em particular, alguns problemas de PO usam modelos de programação linear, que serão detalhados na sequência.

4. Teste de validação e aprimoramento do modelo

De acordo com Andrade,

> um modelo é válido, se, a despeito de sua inexatidão em representar o sistema, ele for capaz de fornecer uma previsão aceitável do comportamento do sistema e uma resposta que possa contribuir para a qualidade da decisão a ser tomada (ANDRADE, 2009, p.10).

Mesmo após os refinamentos anteriores, provavelmente, não estarão bem estimados alguns parâmetros e inter-relacionamentos relevantes. A elaboração de experimentos adequados para testar o modelo permite realizar estas correções.

5. Preparação para aplicação contínua do modelo na organização:

Consiste na instalação do sistema na organização, com documentação adequada. Considera a instalação do software usado para obter a solução do modelo, do banco de dados (ou de sua adequação) e da solução proposta (sistema de apoio à decisão), que gerará relatórios para ajuda à tomada de decisão.

6. Implementação do modelo:

Esta fase envolve treinamento e coleta dos resultados. Observa-se a necessidade de acompanhamento dos resultados obtidos e eventuais revisões.

Nas fases apresentadas, observa-se o constante refinamento do modelo, obtido pela observação de sua adequação ao problema real. Esse processo evidencia a dificuldade de se estabelecer as variáveis, a função objetivo e a atribuição dos valores aos parâmetros.

Quanto aos resultados, uma solução é qualquer especificação de valores para as variáveis de decisão, independente de se tratar de uma escolha desejável ou permissível. Todos os resultados que atendem a todas as restrições são chamados

de soluções viáveis. O conjunto de todos os pontos que satisfazem todas as restrições é conhecido como conjunto viável. Uma solução viável com o valor mais favorável da função objetivo (que maximize ou minimize a função objetivo na região viável) é uma solução ótima, que pode ou não ser única.

Andrade (2009) reforça que na etapa da avaliação final, a experiência do pessoal envolvido no estudo tem papel primordial, “é com a experiência e visão crítica que conseguimos avaliar e determinar a aplicabilidade da decisão (ANDRADE, 2009, p.11).

2.3 Modelo de Programação Linear

O termo programação pode ser entendido como sinônimo de planejamento. O termo linear indica que a função objetivo e as restrições que modelam o problema são funções lineares, ou seja, as variáveis, presentes na função objetivo e nas restrições, apresentam expoente 1, podendo ser multiplicadas por um número real (constante), que são os parâmetros.

Um modelo de programação linear “envolve o planejamento de atividades para obter um resultado ótimo, isto é, um resultado que atinja o melhor objetivo específico (de acordo com o modelo matemático) entre todas as alternativas viáveis (HILLIER; LIBERMANN, 2013, p. 20).

Como explanado, um problema de programação linear é um problema de programação matemática em que a funções objetivo e as restrições são lineares. Sua forma geral é dada por:

Otimizar $z = \sum_{j=1}^{n} c_j . x_j$ *(maximizar ou minimizar)*

Sujeito às:

Restrições funcionais:

$\sum_{j=1}^{n} a_{ij} . x_j = b_j$ *, para* $i = 1, \dots, m$ $(ou\ <,\ \leq, >, \geq)$

Restrições de não negatividade:

$x_j \geq 0$ *para todo j*

Portanto, as variáveis são x_j e os parâmetros são a_{ij} , b_j e c_j, com $i = 1, \dots, m$ e $j = 1, \dots, n$.

De acordo com Lachtermacher (2009), na resolução de todo problema de programação linear estão presentes as seguintes hipóteses:

1. proporcionalidade: o valor da função objetivo z é diretamente proporcional ao valor de cada variável de decisão x_j.

2. aditividade: a função objetivo z é a soma das contribuições individuais das respectivas variáveis de decisão x_j .

3. divisibilidade: as variáveis de decisão x_j podem assumir quaisquer valores que satisfaçam as restrições funcionais e de não negatividade, ou seja, valores fracionários. No caso de assumirem somente valores inteiros, o problema é tratado como modelo de programação linear inteira. Quando alguma variável de decisão assume valor inteiro o modelo é chamado de programação linear mista.

4. Certeza: o valor atribuído aos parâmetros do modelo é assumido como uma constante conhecida. O autor acrescenta que, em casos reais, esta hipótese nem sempre é satisfeita, sendo necessária a análise de sensibilidade dos resultados.

Em geral, emprega-se o método Simplex para a resolução algébrica desse modelo. Quando o problema envolve apenas duas variáveis de decisão, poderá ser resolvido pelo método gráfico, não abordado neste texto.

Na sequência, os modelos típicos estão apresentados em diferentes notações, que foram usadas na criação dos algoritmos implementados com auxílio do software CPLEX.

2.3.1 Problema de alocação de recursos

O problema típico de alocação de recursos é composto pelos elementos apresentados nos dados da Tabela 2.

Tabela 2 - Parâmetros – problema de alocação de recursos

Recursos i	Uso do recurso na atividade j (por unidade de atividade) Atividades j				b_i Quantidade disponível do recurso i
	1	2	...	n	

1	a_{11}	a_{12}	...	a_{1n}	b_1
2	a_{21}	a_{22}	...	a_{2n}	b_2
...	...	...	...	...	...
m	a_{m1}	a_{m2}	...	a_{mn}	b_m
Lucro Unitário c_j	c_1	c_2	...	c_n	
variáveis x_j	x_1	x_2	...	x_n	

Fonte: adaptado de HILLER E LIEBERMAN (2013, p.28)

Deseja-se maximizar a função objetivo lucro total z, em unidades monetárias, em que:

m: total dos recursos i disponíveis.

n: total das atividades j consideradas.

a_{ij}: fração do recurso i consumido por unidade de atividade j.

x_{j}: variáveis de decisão, que indicam o nível de atividade j por unidade de recurso, com $j = 1,2, \ldots, n$.

b_i : termos independentes que indicam a fração de recurso i disponível para alocação nas atividades consideradas, com $i = 1,2, \ldots, m$.

c_j : lucros unitários da atividade j, em unidades monetárias.

Nota-se que as variáveis são x_j e os parâmetros são a_{ij} , b_j e c_j, com $i = 1, \ldots, m$ e $j = 1, \ldots, n$.

Na forma-padrão ou expandida, o modelo é escrito da seguinte forma:

$$Maximizar\ z = c_1.x_1 + c_2.x_2 + \cdots + c_n.x_n$$

Sujeito às:

Restrições funcionais:

$$a_{11}.x_1 + a_{12}.x_2 + \cdots + a_{1n}.x_n \leq b_1 \quad (\text{ou} = , \geq)$$

$$a_{21}.x_1 + a_{22}.x_2 + \cdots + a_{2n}.x_n \leq b_2 \quad (\text{ou} = , \geq)$$

...

$$a_{m1}.x_1 + a_{m2}.x_2 + \cdots + a_{mn}.x_n \leq b_m \quad (\text{ou} = , \geq)$$

Restrições de não negatividade:

$$x_1, x_2 \ldots x_j \geq 0$$

Na forma de somatório, o modelo é definido por:

Maximizar $z = \sum_{j=1}^{n} c_j . x_j$

Sujeito às:

Restrições funcionais:

$\sum_{j=1}^{n} a_{ij} . x_j \leq b_i$ *, para* $i = 1, \dots, m$ (ou $=$, $\geq$)

Restrições de não negatividade:

$x_j \geq 0$ *, para todo j*

Na forma matricial, o modelo é descrito por:

Maximizar $z = C . X$

Sujeito às:

$A . X \leq B$, em que

$C = [c_1, c_2, \dots, c_n]$ vetor linha dos coeficientes da função objetivo

$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$ matriz dos coeficientes das restrições

$X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$, com $x_i >= 0$ *, i =1,2,...n.* vetor coluna dos valores das variáveis de decisão.

$B = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}$ vetor coluna dos valores do lado direito das restrições

Pode-se também minimizar a função objetivo custo total, quando os custos individuais são fornecidos em substituição aos lucros c_j.

2.3.2 Problema da mistura

Este problema consiste em combinar ingredientes para gerar novos produtos. Modela problemas de formulação de rações e adubos, ligas metálicas, composição

nutricional, entre outros. Em geral, pretende-se obter um produto pela mistura de ingredientes disponíveis, obedecendo especificações, de modo a minimizar o custo.

Os dados da Tabela 3 organizam os parâmetros do problema.

Tabela 3 - Parâmetros – problema da mistura

Componentes i	Fração do componente i (por unidade de mistura) no ingrediente j Ingredientes j				b_i Fração mínima do componente i na mistura
	1	2	...	n	
1	a_{11}	a_{12}	...	a_{1n}	b_1
2	a_{21}	a_{22}	...	a_{2n}	b_2
...	...	...	...	...	...
m	a_{m1}	a_{m2}	...	a_{mn}	b_m
Custo Unitário c_j	c_1	c_2	...	c_n	
variáveis x_j	x_1	x_2	...	x_n	

Fonte: a autora.

Deseja-se minimizar a função objetivo custo total z, em unidades monetárias, em que:

m: total de componentes.

n: total de ingredientes.

a_{ij}: fração do componente i no ingrediente j.

x_j: variáveis de decisão, que indicam as quantidades do ingrediente j por unidade da mistura, com $j = 1,2, \ldots, n$.

b_i : termos independentes que indicam a fração do componente i na mistura, com $i = 1,2, \ldots, m$.

c_j : custo unitário do ingrediente j, em unidades monetárias.

Nota-se que as constantes de entrada, ou parâmetros, para o modelo são: c_j, b_i e a_{ij}, com $i = 1, \ldots, m$ e $j = 1, \ldots, n$.

Na forma-padrão ou expandida, o modelo é escrito da seguinte forma:

$$Minimizar\ z = c_1.x_1 + c_2.x_2 + \cdots + c_n.x_n$$

Sujeito às:

Restrições funcionais:

$$a_{11}.x_1 + a_{12}.x_2 + \cdots + a_{1n}.x_n \geq b_1$$

$$a_{21}.x_1 + a_{22}.x_2 + \cdots + a_{2n}.x_n \geq b_2$$

...

$$a_{m1}.x_1 + a_{m2}.x_2 + \cdots + a_{mn}.x_n \geq b_m$$

Restrições de não negatividade:

$$x_1, x_2 \ldots x_j \geq 0$$

Na forma de somatório, o modelo é definido por:

Minimizar $z = \sum_{j=1}^{n} c_j.x_j$

Sujeito às:

Restrições funcionais:

$$\sum_{j=1}^{n} a_{ij}.x_j \geq b_i \text{ , para } i = 1, \ldots, m$$

Restrições de não negatividade:

$x_j \geq 0$ *, para todo j*

Na forma matricial, o modelo é descrito por:

Minimizar $z = C.X$

Sujeito às:

$A.X \geq B$, em que

$C = [c_1, c_2, \ldots, c_n]$

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}$$, com $x_i >= 0$ *, i =1,2,...n*

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{bmatrix}$$

2.3.3 Problema do transporte

Este modelo típico recebeu esse nome pois, inicialmente, foi aplicado na resolução de situações que envolviam a otimização no transporte de mercadorias. Um modelo genérico de transporte “refere-se a distribuir qualquer *comodity* de qualquer grupo de centros de fornecimento (origem) a qualquer grupo de centros de recepção (destino), de modo a minimizar o custo total de distribuição” (HILLER E LIEBERMAN, 2013, p.294). Cada origem apresenta determinada oferta de unidades a serem distribuídas aos destinos, e cada destino tem certa demanda pelas unidades a serem recebidas de cada origem.

Na prática, as ofertas e as demandas representam quantidades máximas e não quantidades fixas.

Os custos de transporte foram consolidados nos dados da Tabela 4.

Tabela 4 - Parâmetros (custos) – problema do transporte

Origem i	Custo de transporte c_{ij} , em unidades monetárias, para o destino j Destino j			
	1	2	...	n
1	c_{11}	c_{12}	...	c_{1n}
2	c_{21}	c_{22}	...	c_{2n}
...	...	...	...	...
m	c_{m1}	c_{m2}	...	c_{mn}

Fonte: a autora.

Os demais parâmetros foram listados nos dados da Tabela 5.

Tabela 5 - Parâmetros (quantidades) – problema do transporte

Origem i	Quantidade transportada Destino j				s_i número de unidades transportadas pela origem i (oferta)
	1	2	...	n	
1	x_{11}	x_{12}	...	x_{1n}	s_1
2	x_{21}	x_{22}	...	x_{2n}	s_2
...	...	...	...	...	...
m	x_{m1}	x_{m2}	...	x_{mn}	s_m
Demanda d_j	d_1	d_2	...	d_n	

Fonte: adaptado de HILLER E LIEBERMAN (2013, p.295)

Deseja-se minimizar a função objetivo custo total de transporte z, em unidades monetárias, em que:

m: total de locais de origem i.

n: total de locais de destino j.

c_{ij}: custo por unidade transportada da origem i ao destino j, em unidades monetárias.

x_{ij}: variáveis de decisão, que indicam a quantidade transportada da origem i para o destino j, com $i = 1, \dots, m$ e $j = 1, \dots, n$.

s_i : termos independentes que indicam o número de unidades distribuídas por origem i , com $i = 1,2, \dots, m$.

d_j : número de unidades recebidas pelo destino j, $j = 1, \dots, n$.

Nota-se que as variáveis são x_{ij} e os parâmetros são s_i, d_j e c_{ij}, com $i = 1, \dots, m$ e $j = 1, \dots, n$. As origens não produzem mais do que suas capacidades e os destinos, não recebem quantidades superiores às suas demandas. Este problema apresenta soluções viáveis se e somente se $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$, ou seja, quando as quantidades de oferta e demanda forem iguais.

Na forma de somatório, o modelo é definido por:

Minimizar $\quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}.x_{ij}$

Sujeito às:

Restrições funcionais (para o problema balanceado, as restrições são igualdades, senão são desigualdades do tipo ≥):

$$\sum_{j=1}^{n} x_{ij} = s_i \ , \text{para } i = 1,2, \dots, m$$

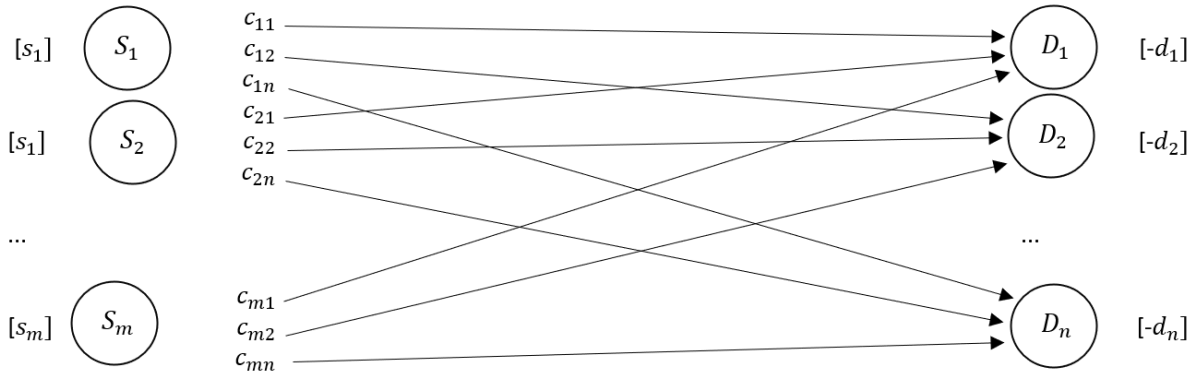
$$\sum_{i=1}^{m} x_{ij} = d_j \quad , \text{para } j = 1,2, \dots, n$$

Restrições de não negatividade:

$$x_{ij} \geq 0 \quad \text{para todo } i \text{ e } j$$

Este problema pode ser representado em rede (Figura 2):

Figura 2 – Representação em rede – problema do transporte



Fonte: adaptado de HILLER E LIEBERMAN (2013, p.296)

Nesta representação, as setas indicam possíveis rotas da origem ao destino, c_{mn} é o custo de transporte por rota, s_m é a capacidade de despacho da origem S_m, d_n é a demanda no destino D_n.

Qualquer problema que possa ser formulado por meio dos parâmetros apresentados nos dados das Tabelas 4 e 5, mesmo não sendo de transporte, pode ser resolvido por esse problema típico.

Nem sempre as quantidades demandadas e ofertadas são iguais, o que requer uma adaptação do problema, tratado no segundo exemplo do modelo de transporte. Nele, é tratado o modelo do transporte com transbordo e são considerados locais intermediários entre a origem e o destino.

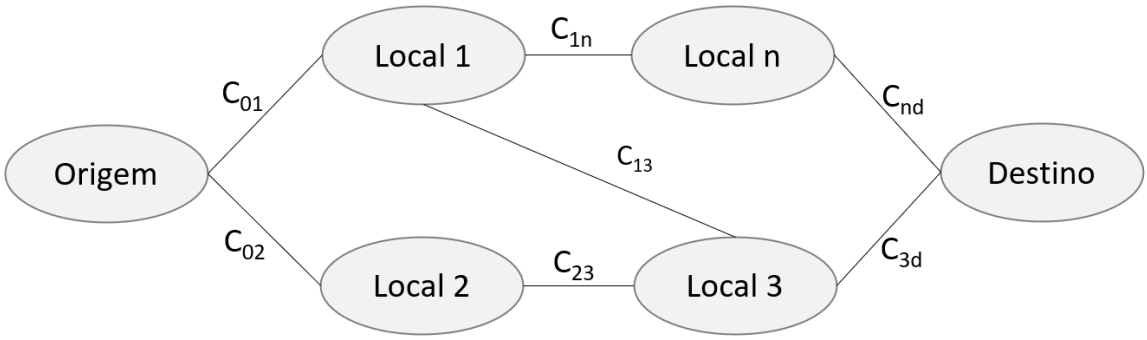
2.3.4 Problema do menor caminho

Esre modelo típico é um caso particular de problemas de rede, no qual as variáveis x_{ij} são binárias e indicam se a aresta $i \rightarrow j$ faz parte do menor caminho, considerando o sentido da cidade i para a cidade j. Quando a variável assume o valor 1, indica que o trecho deverá ser percorrido e, quando assume o valor 0, indica que o trecho não deverá ser percorrido. A função objetivo é uma função distância percorrida entre a origem e o destino, passando por lugares intermediários, a ser minimizada.

Observando a representação em rede (Figura *3*), o problema apresenta uma origem e um destino predeterminados. O parâmetro fluxo desejado em cada vértice é definido por: se o vértice for a origem, o fluxo assumirá o valor -1, indicando saída; se o vértice for o destino, o fluxo assumirá o valor 1, indicando chegada; e, nos vértices

intermediários, o fluxo assumirá o valor 0, indicando a passagem pelo vértice (chegada e saída).

Figura 3 – Representação em rede – problema do menor caminho



Fonte: a autora.

Os dados da Tabela 6 apresentam a matriz quadrada simétrica com as distâncias entre os vértices. Quando os vértices não possuem ligação, usamos o artifício do *big* M, ou seja, inserimos um valor alto de forma que a minimização da função objetivo elimine esse percurso.

Tabela 6 - Parâmetros (distâncias) – problema do menor caminho

Vértices i (origens)	Vértices j (destinos)			
	1	2	...	n
1	c_{11}	c_{12}	...	c_{1n}
2	c_{21}	c_{22}	...	c_{2n}
...	...	...	...	...
n	c_{n1}	c_{n2}	...	c_{nn}

Fonte: a autora.

Deseja-se minimizar a função distância z.

Nos dados da Tabela 7 estão as variáveis binárias x_{ij}, que se deseja conhecer. A soma de cada linha indica o fluxo de saída em cada vértice. A soma de cada coluna indica o fluxo de entrada em cada vértice. Para montar as restrições, para cada vértice, tomamos a diferença entre fluxos de entrada e saída, que é igual ao parâmetro fluxo desejado b_i.

Tabela 7 - Parâmetros (fluxos) – problema do menor caminho

						restrições	
Vértices i (origens)	Vértices j (destinos)				Saídas em cada vértice	Fluxo em cada vértice	Fluxo b_i
	1	2	...	n			
Origem 1	x_{11}	x_{12}	...	x_{1n}	$\sum_{j=1}^{n} x_{1j}$	$\sum_{i=1}^{n} x_{i1} - \sum_{j=1}^{n} x_{1j}$	-1
Intermediária 2 ... Intermediária $n-1$	x_{21}	x_{22}	...	x_{2n}	$\sum_{j=1}^{n} x_{2j}$	$\sum_{i=1}^{n} x_{i2} - \sum_{j=1}^{n} x_{2j}$	0
...	...	...	...	...	...	...	...
Destino n	x_{n1}	x_{n2}	...	x_{nn}	$\sum_{j=1}^{n} x_{nj}$	$\sum_{i=1}^{n} x_{in} - \sum_{j=1}^{n} x_{nj}$	1
Entradas em cada vértice	$\sum_{i=1}^{n} x_{i1}$	$\sum_{i=1}^{n} x_{i2}$	...	$\sum_{i=1}^{n} x_{in}$			

Fonte: a autora.

Na forma de somatório, o modelo é definido por:

Minimizar $z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}.x_{ij}$

Sujeito às:

Restrições funcionais:

$$\sum_{i=1}^{n} x_{ij} (para\ j = 1, \dots, n) - \sum_{j=1}^{n} x_{ij} = b_i\ (para\ i = 1, \dots, m)$$

$$x_{ij} \in \{0,1\}$$

Deseja-se minimizar a função objetivo distância z, em unidade de medida linear, em que:

n: número de vértices (nós).

i: origens, $i = 1 \dots, n$.

j: destinos, $j = 1, \dots, n$.

c_{ij}: distância entre a origem i e o destino j.

b_i: fluxo no vértice i, sendo: $b_i := -1$ (origem), $b_i := 0$ (vértices intermediários) e $b_i := 1$ (destino).

x_{ij}: variáveis de decisão que indicam se a aresta $i \to j$ faz parte da decisão. Se $x_{\mathrm{ij}} = 1$, então a aresta faz parte do caminho mínimo e, se $x_{\mathrm{ij}} = 0$, a aresta não faz parte da solução.

Nota-se que as constantes de entrada, ou parâmetros, para o modelo são: c_{ij} e b_i .

2.3.5 Problema do planejamento agregado

O modelo de planejamento agregado procura atender a demanda, por períodos, usando diferentes recursos de mão de obra (regime normal, regime de horas extras ou terceirização), buscando balancear a produção de modo a minimizar os custos com pessoal. Pode-se considerar, também, o estoque em cada período.

Os custos unitários, por tipo de mão de obra e por períodos são consolidados nos dados da Tabela 8.

Tabela 8 - Parâmetros (custos) – exemplo problema do planejamento agregado

Custos Unitários por regime de contratação i e estoque	Custos unitários por regime i no período j. períodos j				
		$mês1$	$mês2$	...	$mês\ n$
$Custo\ Regime\ 1 \quad C_1$		C_{11}	C_{12}	...	C_{1n}
$Custo\ Regime\ 2 \quad C_2$		C_{21}	C_{22}	...	C_{2n}
...		...	...	...	...
$Custo\ Regime\ m \quad C_m$		C_{m1}	C_{m2}	...	C_{mn}
	Custos unitários $custoE_j$ períodos j				
	$anterior$	$mês1$	$mês2$	...	$mês\ n$
$Estoque\ E_j$	$custoE_0$	$custoE_1$	$custoE_2$	...	$custoE_n$

Fonte: a autora.

Os dados da Tabela 9 apresentam as ofertas, por tipo de mão de obra e a demanda em cada período.

Tabela 9 - Parâmetros – exemplo problema do planejamento agregado

Oferta por tipo de regime de mão de obra i e demanda	Oferta por regime i e períodos j			
	$mês1$	$mês2$	...	$mês\ n$
$Oferta\ Regime\ 1\ \ O_1$	O_{11}	O_{12}	...	O_{1n}
...	...	...	...	...
$Oferta\ Regime\ m\ \ O_m$	O_{m1}	O_{m2}	...	O_{mn}
$Demanda_j$	$Demanda_1$	$Demanda_2$	...	$Demanda_n$

Fonte: a autora.

Os dados da Tabela 10 apresentam as variáveis de decisão, em cada período, indicando a produção a ser realizada por tipo de regime de mão de obra.

Tabela 10 - Variáveis de decisão–exemplo problema do planejamento agregado

Produção por tipo i e estoque	Quantidade produzida do regime i no período j				
		$mês1$	$mês2$	...	$mês\ n$
$Produção\ Regime\ 1\ \ P_1$		P_{11}	P_{12}	...	P_{1m}
...		...	...	...	...
$Produção\ Regime\ m\ \ P_m$		P_{m1}	P_{m2}	...	P_{mn}
	Quantidade armazenada no período j				
	anterior	$mês1$	$mês2$	...	$mês\ n$
$Estoque\ E_j$	$E_0 = 0$	E_1	E_2	...	E_n

Fonte: a autora.

Deseja-se minimizar a função objetivo custo total de produção z, em unidades monetárias, em que:

m: total de regimes de mão de obra, variando de $i = 1, \dots, m$.

n: total de períodos, variando de $j = 1, \dots, n$.

C_{ij}: custo por unidade produzida pelo tipo de mão de obra i, no período j, $i = 1,2, \dots m$ e $j = 1, \dots, n$.

$custoE_j$: custo por unidade estocada no período j. Matriz com $n + 1$ colunas, para informação do custo de estoque no período 0 (estoque inicial).

O_{ij} : oferta ou capacidade de produção por tipo de regime i, no período j.

$Demanda_j$: demanda de produção do produto, no período j.

P_{ij}: variáveis de decisão, que indicam a quantidade produzida no regime de horas i, no período j.

E_j: variáveis de decisão, que indicam a quantidade estocada, no período j, após produção e atendimento da demanda. O estoque inicial é um parâmetro a ser informado, no caso, $E_0 = 0$. Para cada período j, tem-se:

$E_j = E_{j-1} + \sum_{i=0}^{m} P_{ij} - demanda_j$

Na forma de somatório, o modelo é definido por:

Minimizar

$z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{ij}.C_{ij} + \sum_{j=0}^{n} custoE_j.E_j$

Sujeito às:

$E_0 = 0$ (estoque no período $j - 1$)

Restrições funcionais:

$P_{ij} \leq O_{ij}$, para todo i e j

$E_{j-1} + \sum_{i=0}^{m} P_{ij} - demanda_j = E_j$, para todo i e j

Restrições de não negatividade:

$P_{11} + \cdots + P_{mn}, E_j \geq 0$, para todo i e j

2.3.6 Problema do corte de barras

O problema do corte de barras é uma aplicação do problema de cobertura de conjuntos. De acordo com Pizzolato e Gandolpho (2009), as decisões tomadas devem garantir a satisfação de um conjunto de restrições expressas como elementos de um conjunto. Acrescentam que outra aplicação desse problema é a alocação de tripulação de aeronaves em diversos voos.

Este é um dos tipos de problemas de programação inteira, quando as variáveis de decisão assumem valores inteiros.

Considerando o corte de barras (unidimensional), deseja-se cortar barras de tamanho conhecido em peças menores, também de tamanhos conhecidos, buscando minimizar as perdas ou minimizar o total de barras necessárias para atender à

demanda. Resumindo, são conhecidos o tamanho da barra, o tamanho das peças, os padrões de cortes e as demandas por tamanho da peça.

Os dados da tabela 11 apresentam os possíveis padrões de corte, considerando os tamanhos das peças que se deseja cortar e as perdas decorrentes de cada padrão de corte. Nota-se que $\sum_{j=i}^{n} c_{ij} \leq L$, com $i = 1,..,m$. A perda é calculada no caso em que se deseja minimizar a perda obtida em cada padrão de corte.

Tabela 11 - Parâmetros (padrões de corte) – problema do corte de barras

Dimensões desejadas i das peças, em unidade linear	Quantidade de peças em cada padrão de corte n com $j = 1,2,\ldots,n$			
	$Padrão\ 1$	$Padrão\ 2$	...	$Padrão\ n$
l_1	c_{11}	c_{12}	...	c_{1n}
l_2	c_{21}	c_{22}	...	c_{2n}
...	...	...	...	...
l_m	c_{m1}	c_{m2}	...	c_{mn}
Perda p_j	$L - (l_1.c_{11} + l_2.c_{21} + \cdots + l_m.c_{m1})$	...		$L - (l_1.c_{1n} + l_2.c_{2n} + \cdots + l_m.c_{mn})$

Fonte: a autora.

Os dados da Tabela 12 apresentam as variáveis de decisão que se deseja calcular com o objetivo de minimizar a quantidade de barras.

Tabela 12 - Variáveis de decisão – problema do corte de barras

variáveis	Quantidades de barras cortadas, conforme os padrões de corte j, $j = 1,\ldots,n$			
	$Padrão\ 1$	$Padrão\ 2$	...	$Padrão\ n$
x_j	x_1	x_2	...	x_n

Fonte: a autora.

Os dados da Tabela 13 apresentam as restrições de demanda.

Tabela 13 - Restrições – problema do corte de barras

Dimensões desejadas l_i das peças, em unidade linear	Quantidades por padrão j				d_i Demanda das i peças de dimensão l_i
	$Padrão\ 1$	$Padrão\ 2$	...	$Padrão\ n$	
l_1	$c_{11}.x_1$	$c_{12}.x_2$	...	$c_{1n}.x_n$	d_1
l_2	$c_{21}.x_1$	$c_{22}.x_2$	...	$c_{2n}.x_n$	d_2
...	...	...	...	...	...

m	$c_{m1}.x_1$	c_{m2},x_2	...	$c_{mn}.x_n$	d_m

Fonte: a autora.

Em que:

L: tamanho original da barra.

l_i : dimensões de peças menores que se deseja obter cortando a barra de tamanho L, com $i = 1,\ldots.m$, em unidade linear.

m: total de tipos de peças menores que a barra, $i = 1,\ldots.m$.

n: total de padrões de corte, com $j = 1,\ldots,n$.

c_{ij}: quantidade de peças de dimensão l_i em cada padrão de corte j.

p_j : perda no padrão de corte j.

d_i : demanda de peça com dimensão l_i .

x_j: quantidade de barras cortadas por padrão de corte j.

Considerando que se deseja minimizar a função objetivo número de barras z, o modelo na forma de somatório é dado por:

$Minimizar\ z = \sum_{j=1}^{n} x_j$

Sujeito às:

Restrições funcionais:

$\sum_{j=1}^{n} c_{ij}x_j = d_i$, para $i = 1,2,\ldots m.$

Restrições de não negatividade:

x_j inteiro maior que zero, para todo j.

Considerando que se deseja minimizar a função objetivo perda z, o modelo na forma de somatório é dado por:

$p_j = L - (l_1.c_{11} + l_2.c_{21} + \cdots + l_m.c_{m1})$

$Minimizar\ z = \sum_{j=1}^{n} p_j x_j$

Sujeito às:

Restrições funcionais:

$\sum_{j=1}^{n} c_{ij} x_j = d_i$, para $i = 1,2, \ldots m.$

Restrições de não negatividade:

x_j inteiro e maior que zero para todo i e j

2.3.7 Problema da mochila

O problema da mochila consiste em selecionar n itens ($j = 1, \ldots, n$) para colocar em uma mochila, de tamanho definido b, e são conhecidos os pesos a_j destes itens e seus valores c_j. Busca-se maximizar o valor total dos itens inseridos na mochila. As variáveis de decisão x_j são binárias e definem se o item será ou não inserido na mochila.

De acordo com Pizzolato e Gandolpho (2009), este modelo é aplicado em situações em que são necessárias a escolha de itens com algum tipo de valor ou prioridade.

Os dados da Tabela 14 apresentam os parâmetros do modelo.

Tabela 14 - Parâmetros – problema da mochila

			$Mochila$
Itens j	$Peso\ a_j$	$Valor\ c_j$	x_j
1	a_1	c_1	x_1
2	a_2	c_2	x_2
...	...	...	...
n	a_n	c_n	x_n
Peso mochila			b

Fonte: a autora.

O modelo na forma de somatório é dado por:

$Maximizar\ z = \sum_{j=1}^{n} c_j x_j$

Sujeito às:

Restrições funcionais:

$$\sum_{j=1}^{n} a_j x_j \leq b$$

Restrições de domínio:

$$x_j = \{0,1\}, \text{para todo } j$$

Em que:

n: número de itens, com $j = 1, \ldots, n$.

a_j: peso do item j.

c_j: valor do item j.

b : tamanho ou capacidade da mochila.

x_j : variável de decisão binária, sendo igual a 1, se o item for selecionado e igual a 0, se o item não for selecionado.

3. DESENVOLVIMENTO

Neste item, é apresentada a implementação dos modelos típicos mostrados em precedência, por meio de exemplos. Estes exemplos também foram desenvolvidos com o solver, presente no Excel (Microsoft) e encontram-se no Apêndice B.

3.1 Problema de alocação de recursos

Exemplo adaptado de Hillier e Lieberman (2013, p.21): Uma empresa fabrica produtos de vidro e possui uma planta com três fábricas. A Fábrica 1 produz esquadrias de alumínio e ferragens, a Fábrica 2 produz esquadrias de madeira e a Fábrica 3, vidro e monta os produtos. Na modernização da empresa, optou-se pelo lançamento de dois novos produtos:

- Produto 1: porta de vidro de 2,5 cm com esquadria de alumínio
- Produto 2: janela duplamente adornada com esquadrias de madeira, de 1,20m x 1,80m

O produto 1 é feito nas Fábricas 1 e 3 e o produto 2, nas Fábricas 2 e 3. Buscando otimizar o lucro total, definiu-se o seguinte problema:

Determinar quais as taxas de produção para os produtos 1 e 2, buscando maximizar o lucro total, Sujeito àss restrições de capacidade produtiva: cada produto será fabricado em lotes de 20 e a taxa de produção é definida como o número de lotes produzidos por semana.

Os parâmetros quantidade de horas disponíveis pelas fábricas (b_i), lucro por lote de cada produto (c_j), e o uso dos recursos na produção dos produtos 1 e 2 (a_{ij}), são conhecidos (Tabela 15).

Após o levantamento dos tempos de produção para cada produto, os dados foram listados na Tabela 15:

Tabela 15 - Parâmetros – exemplo do problema de alocação de recursos

Fábrica i	Tempo de produção na atividade j (por lote, em horas/semana) do produtos j		b_i Tempo disponível da fábrica i (horas/semana)
	Produto 1	Produto 2 ($n = 2$)	
1	1	0	4

2	0	2	12
3 (m=3)	3	2	18
Lucro por lote c_j	3	5	
variáveis x_j $(x_j \geq 0)$	x_1	x_2	

Fonte: adaptado de Hillier e Lieberman (2013, p.22)

Deseja-se maximizar a função objetivo z, que é a função lucro total, em milhares de dólares, considerando:

m: total de fábricas i disponíveis ($m = 3$).

n: total produtos j considerados ($n = 2$).

a_{ij}: frações da fábrica i consumida para produzir o produto j, em horas por semana.

x_{j}: variáveis de decisão, número de lotes a ser produzido por semana do produto j, com $j = 1{,}2$.

b_i : termos independentes que indicam a fração da fábrica i disponível para alocação na produção dos produtos j, com $i = 1{,}2{,}3$, em horas por semana.

c_i : lucros por lote do produto, em unidades monetárias, com $j = 1{,}2$.

Na forma-padrão ou expandida, o modelo é escrito da seguinte forma:

$Maximizar\ z = 3x_1 + 5x_2$

Sujeito às:

Restrições funcionais:

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

Restrições de não negatividade:

$$x_1, x_2 \geq 0$$

Na forma de somatório, o modelo é definido por:

$Maximizar \qquad z = \sum_{j=1}^{2} c_j . x_j$

Sujeito às:

Restrições funcionais: $\sum_{j=1}^{2} a_{ij}.x_j \leq b_i$, $para\, i = 1,2,3$

Restrições de não negatividade: $x_1, x_2 \geq 0$

Na forma matricial, o modelo é descrito por:

$Maximizar\ z = C.X$

Sujeito às:

$A.X \leq B$, em que

$C = [3,5]$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix}$$

$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, com $x_1, x_2 \geq 0$

$$B = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

O CPLEX permite o registro do modelo nas diferentes formas. As formas matricial e de somatório são mais indicadas quando há muitas variáveis de decisão.

Para o exemplo dado, usando a notação padrão são declaradas as variáveis de decisão, a função objetivo e as restrições (Figura 4).

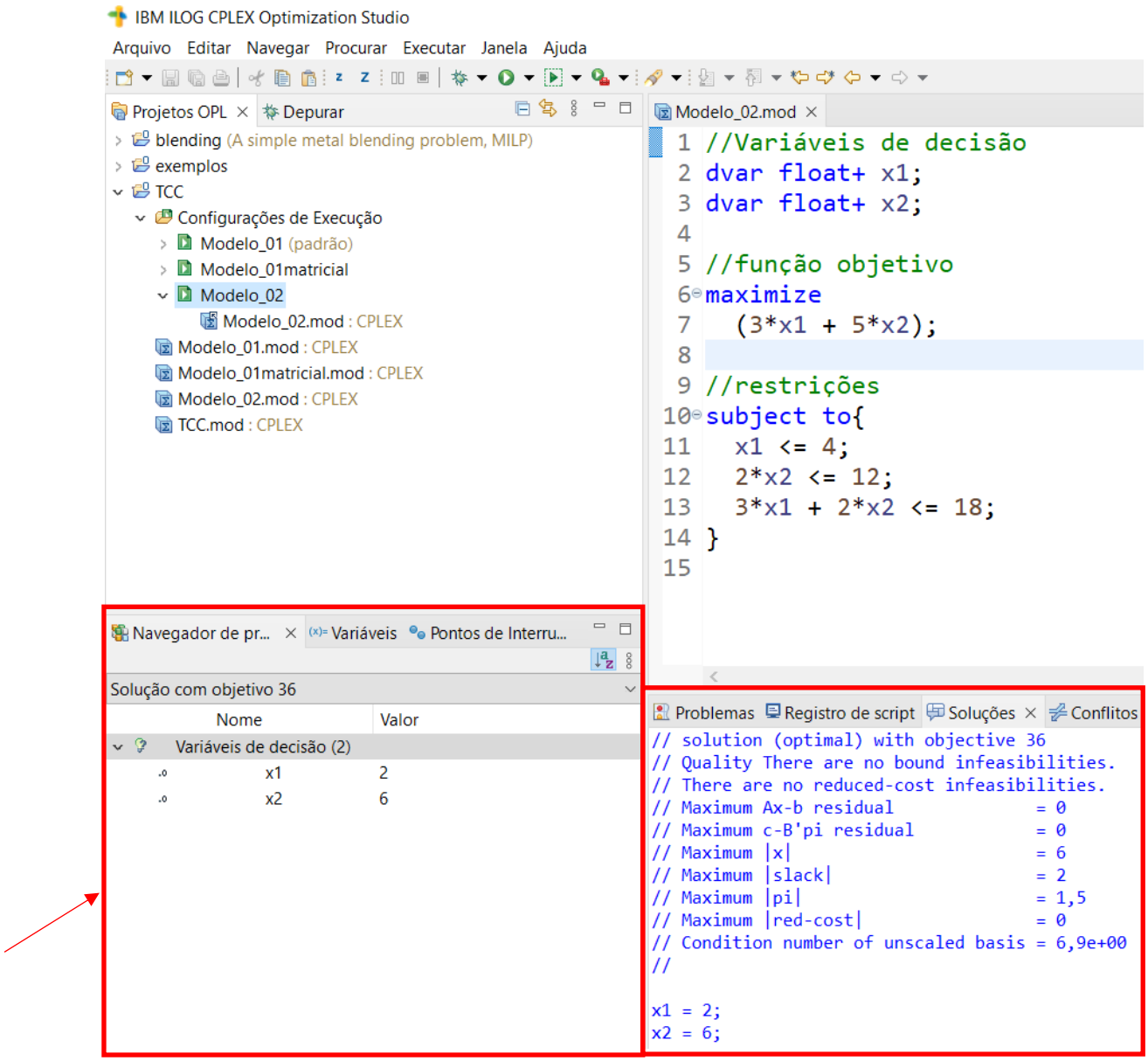
Figura 4 - CPLEX – código do problema de alocação de recursos

```
1  //Variáveis de decisão
2  dvar float+ x1;
3  dvar float+ x2;
4
5  //função objetivo
6  maximize
7    (3*x1 + 5*x2);
8
9  //restrições
10 subject to{
11   x1 <= 4;
12   2*x2 <= 12;
13   3*x1 + 2*x2 <= 18;
14 }
```

Fonte: a autora.

Na sequência, o software retorna o valor ótimo e os valores das variáveis de decisão que maximizam a função objetivo (Figura 5).

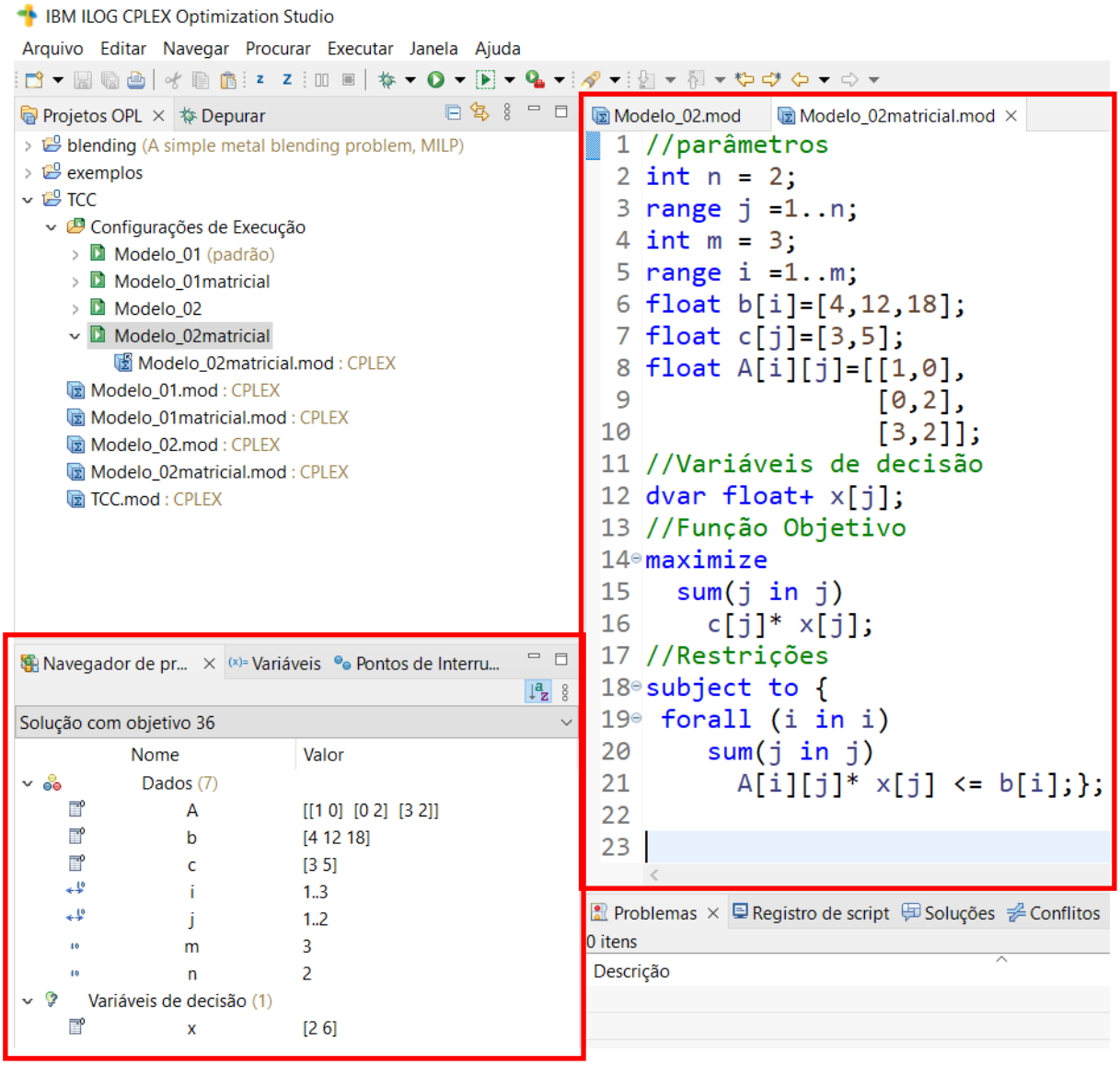
Figura 5 - CPLEX – resultado do problema de alocação de recursos



Fonte: a autora.

Usando a forma matricial, obtemos o mesmo resultado (Figura 6). Ressalta-se que a função objetivo e as restrições são construídas por laços.

Figura 6 - CPLEX – código (matricial) do problema de alocação de recursos

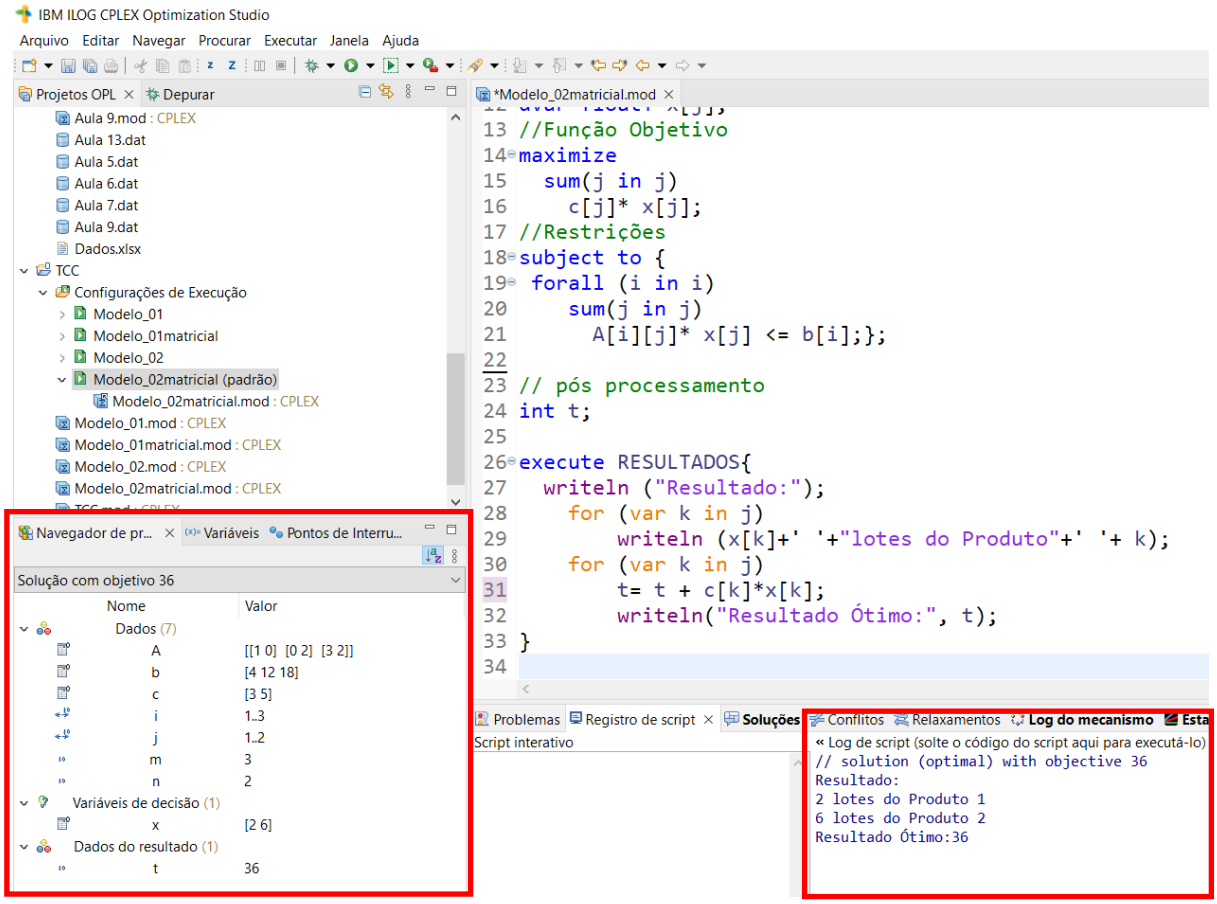


Fonte: a autora.

Após o processamento, as saídas foram formatadas usando recursos da linguagem *Javascript* (Figura 7).

Este exemplo foi montado, usando as formas expandida e matricial para mostrar as possibilidades do CPLEX. Na resolução dos próximos problemas, optou-se pela forma mais conveniente.

Figura 7 - CPLEX – resultados do problema de alocação de recursos



Fonte: a autora.

3.2 Problema da mistura

Exemplo (adaptado de Hillier e Lieberman (2013, p.75): Edmundo adora bifes e batatas e decidiu entrar em uma dieta rígida, usando somente esses alimentos em todas as suas refeições, complementadas por líquidos e suplementos vitamínicos. Ele deseja certificar-se de que se alimenta das quantidades corretas destes dois tipos de alimentos, a fim de atender a determinados requisitos nutricionais. Assim, obteve as informações nutricionais e de custo mostradas nos dados da Tabela 16. Edmundo quer determinar o número de refeições diárias (pode ser fracionada) com bifes e batatas que atenderá a estas exigências a um custo mínimo.

Tabela 16 - Parâmetros – exemplo do problema da mistura

Componentes nutricionais i	Fração do componente i (em gramas) no ingrediente j Ingredientes j	b_i

	$Bifes$	$Batatas\ (n=2)$	Fração do componente i na mistura (exigência diária)
$Carboidratos$	5	15	≥ 50
$Proteína$	20	5	≥ 40
$Gordura\ (m=3)$	15	2	≤ 60
Custo Unitário c_j	4	2	
variáveis x_j	x_1	x_2	

Fonte: a autora.

Deseja-se minimizar a função objetivo custo total z, em dólares, em que:

m: total de componentes nutricionais ($m = 3$).

n: total de ingredientes j ($n = 2$).

a_{ij}: fração do componente i no ingrediente j, com $i = 1,2,3$ e $j = 1,2$.

x_j: variáveis de decisão, que indicam as quantidades do ingrediente j, em gramas, com $j = 1,2$.

b_i : termos independentes que indicam a fração do componente i na mistura, com $i = 1,2,3$.

c_j : custo unitário do componente, em dólares, com $j = 1,2$.

Ressalta-se que esse exemplo é uma variação do problema da mistura com restrições com sinais $\geq$ ou $\leq$.

Na forma-padrão ou expandida, o modelo é escrito da seguinte forma:

$Minimizar\ z = 4x_1 + 2x_2$

Sujeito às:

Restrições funcionais:

$$5x_1 + 15x_2 \geq 50$$

$$20x_1 + 5x_2 \geq 40$$

$$15x_1 + 2x_2 \leq 60$$

Restrições de não negatividade:

$$x_1, x_2 \geq 0$$

Na forma de somatório, o modelo é definido por:

$Minimizar \qquad z = \sum_{j=1}^{2} c_j . x_j$

Sujeito às:

Restrições funcionais:

$$\sum_{j=1}^{2} a_{ij} . x_j \geq b_i \ , para \ i = 1,2$$

$$\sum_{j=1}^{2} a_{ij} . x_j \leq b_i \ , para \ i = 3$$

Restrições de não negatividade:

$$x_1, x_2 \geq 0$$

Na forma matricial, o modelo é descrito por:

$Minimizar\, z = C . X$

Sujeito às:

$A . X \geq B$, em que

$C = [4,2]$

$$A = \begin{bmatrix} 5 & 15 \\ 20 & 5 \\ 15 & 2 \end{bmatrix}$$

$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, com $x_1, x_2 \geq 0$

$$B = \begin{bmatrix} 50 \\ 40 \\ 60 \end{bmatrix}$$

Para o exemplo dado, são declaradas no CPLEX as variáveis de decisão, a função objetivo, as restrições e uma saída dos resultados (Figura 8).

Figura 8 - CPLEX – código do problema de mistura

```
1 //parâmetros
2 int n = 2;
3 range j =1..n;
4 int m = 3;
5 range i =1..m;
6
7 int o = 2;
8 range k =1..o;
9 int p = 3;
```

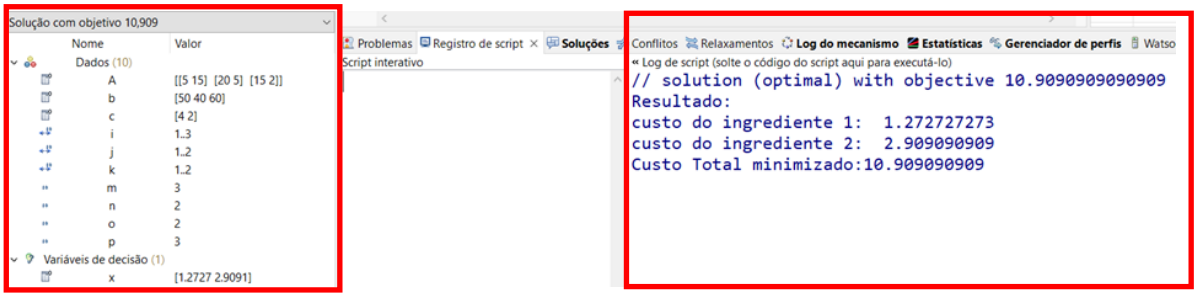
```
10
11 float b[i]=[50,40,60];
12 float c[j]=[4,2];
13 float A[i][j]=[[5,15],
14                [20,5],
15                [15,2]];
16
17 //Variáveis de decisão
18 dvar float+ x[j];
19
20 //Função Objetivo
21 minimize
22   sum(n in j)
23     c[n]* x[n];
24
25 //Restrições
26 subject to {
27  forall (o in k)
28     sum(n in j)
29       A[o][n]* x[n] >= b[o];
30
31  sum(n in j)
32     A[p][n]* x[n] <= b[p];
33
34 };
35
36 // JavaScript
37 execute RESULTADOS{
38   writeln ("Resultado:");
39     for (var n in j)
40         writeln ("custo do ingrediente"+ ' '+ n +": "+ ' '+ x[n]);
41   writeln ("Custo Total minimizado:", cplex.getObjValue());
42
43 }
44
```

Fonte: a autora.

No pós-processamento (*Javascript*), o comando cplex.getObjValue() retorna o resultado da função objetivo.

Após o processamento, são apresentados os resultados e a saída solicitada com o auxílio do Javascript (Figura 9).

Figura 9 - CPLEX – resultado do problema de mistura



Fonte: a autora.

3.3 Problema do transporte 1

Exemplo (adaptado de Hillier e Lieberman (2013, p.291): Uma empresa possui três fábricas de enlatados e transporta o produto por caminhão para quatro depósitos.

Os custos de transporte impactam consideravelmente o preço do produto e a gerência estuda reduzi-los tanto quanto possível. Estimou-se o volume proveniente de cada fábrica e destinou-se a cada depósito certa quantidade de suprimento. Essas informações (em unidades de carretas), em conjunto com o custo de transporte por carreta para cada combinação fábrica-depósito foram divulgadas nos dados da Tabela 17. Identifica-se que há uma carga total a ser remetida de 300 carretas. Busca-se determinar qual plano de destinação destas remessas às diversas combinações fábrica-depósito minimizaria o custo total de remessa desse produto.

Tabela 17 - Parâmetros (custos) – exemplo do problema do transporte 1

Fábrica i (Origem)	Custo de transporte c_{ij}, em US$, por caminhão carregado até o depósito j Depósitos j (destino)			
	1	2	3	$n = 4$
1	464	513	654	867
2	352	416	690	791
3 ($m = 3$)	995	682	388	685

Fonte: adaptado de HILLER E LIEBERMAN (2013, p.293)

Os dados da Tabela 18 concentram as quantidades a serem transportadas, que se deseja encontrar, a partir das restrições de demanda e oferta.

Tabela 18 - Parâmetros (quantidades) – exemplo do problema do transporte 1

Fábrica i (Origem)	Quantidade transportada (por carretas) Depósitos j (destino)				s_i número de unidades distribuídas pela origem i (oferta)
	1	2	3	$n = 4$	
1	x_{11}	x_{12}	x_{13}	x_{14}	75
2	x_{21}	x_{22}	x_{23}	x_{24}	125
$m = 3$	x_{31}	x_{32}	x_{33}	x_{34}	100
d_j: número de unidades recebidas pelo destino j	80	65	70	85	300 / 300

Fonte: adaptado de HILLER E LIEBERMAN (2013, p.293)

Deseja-se minimizar a função objetivo custo total de transporte z, em dólares, em que:

m: total de fábricas j ($m = 3$).

n: total de depósitos i ($n = 4$).

c_{ij}: custo por unidade transportada da fábrica i ao depósito j, em dólares.

x_{ij}: variáveis de decisão, que representam o número de carretas a ser despachado da fábrica i para o depósito j, com $i = 1,2,3$ e $j = 1,2,3,4$.

s_i : termos independentes que indicam o número de unidades distribuídas pela origem i, com $i = 1,2,3$.

d_j : número de unidades recebidas pelo depósito $j, \text{com } j = 1,2,3,4$.

Na forma de expandida, o modelo é definido por:

Minimizar $z = 464x_{11} + 513x_{12} + 654x_{13} + 867x_{14} + 352x_{21} + 416x_{22} + 690x_{23} + 791x_{24} + 995x_{31} + 682x_{32} + 388x_{33} + 685x_{34}$

Sujeito àss restrições:

De oferta:
$$x_{11} + x_{12} + x_{13} + x_{14} \leq 75$$
$$x_{21} + x_{22} + x_{23} + x_{24} \leq 125$$
$$x_{31} + x_{32} + x_{33} + x_{34} \leq 100$$

De demanda:
$$x_{11} + x_{21} + x_{31} \geq 80$$
$$x_{12} + x_{22} + x_{32} \geq 65$$
$$x_{13} + x_{23} + x_{33} \geq 70$$
$$x_{14} + x_{24} + x_{34} \geq 85$$

De não negatividade: $x_{ij} \geq 0 \quad \text{para todo } i = 1,2,3 \text{ e } j = 1,2,3,4$

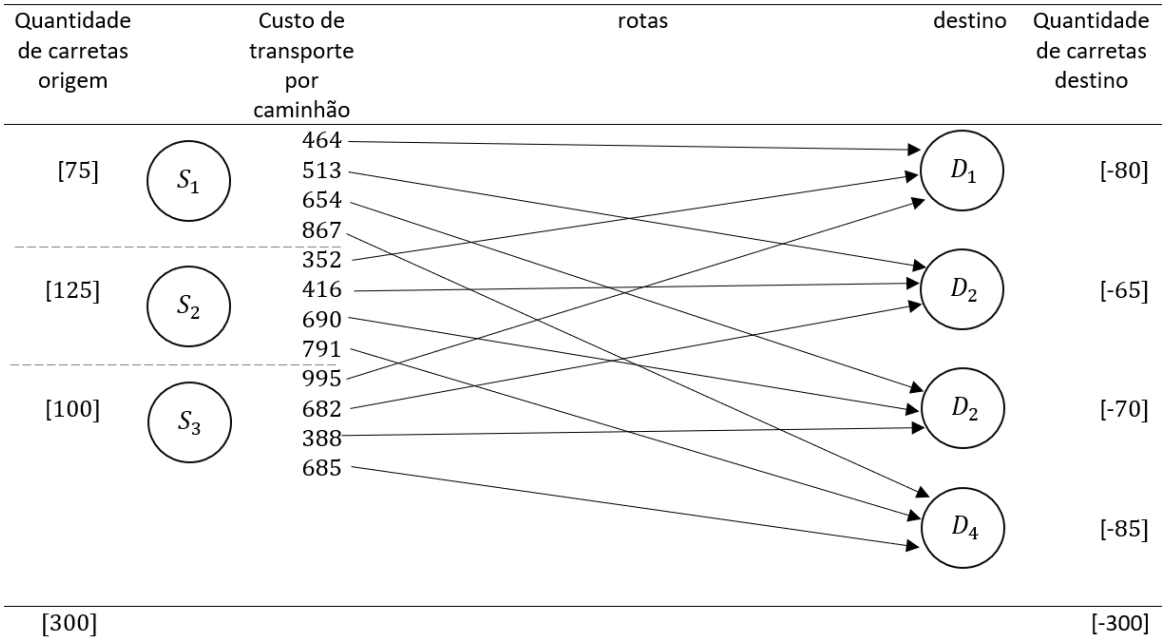
Na forma de somatório, o modelo é definido por:

Minimizar $z = \sum_{i=1}^{3} \sum_{j=1}^{4} c_{ij}.x_{ij}$

Sujeito às:
$$\sum_{j=1}^{4} x_{ij} \leq s_i, \text{ para } i = 1,2,3$$
$$\sum_{i=1}^{3} x_{ij} \geq d_j \text{ para } j = 1,2,3,4$$
$$x_{ij} \geq 0 \quad \text{para todo } i \text{ e } j$$

A Figura 10 apresenta esse problema representado na forma de rede.

Figura 10 – Rede de transporte – problema do transporte 1



Fonte: adaptado de HILLER E LIEBERMAN, 2013, p.293

Note que o problema tem soluções viáveis, pois está balanceado, ou seja,

$\sum_{i=1}^{3} s_i = \sum_{j=1}^{4} d_j \rightarrow 75 + 125 + 100 = 80 + 65 + 70 + 85 \rightarrow 300 = 300$

Na sequência, são apresentados o input no CPLEX, incluindo uma formatação do resultado (Figura 11).

Figura 11 - CPLEX – código do problema do transporte 1

```
1 //parametros
2 int nFabricas = 3;
3 range Fabricas =1..nFabricas;
4 int nDepositos = 4;
5 range Depositos = 1..nDepositos;
6
7 float Capacidade[Fabricas]=[75,125,100];  //oferta de cada fábrica
8 float Demanda[Depositos] = [80,65,70,85]; //demanda de cada depósito
9                                    //d1  d2  d3  d3
10 float Custo[Fabricas][Depositos]=[[464,513,654,867],  //fábrica 1
11                                  [352,416,690,791],  //fábrica 2
12                                  [995,682,388,685]   //fábrica 3
13                                 ];
14 //Variáveis de decisão
15 dvar float+ Transportado[Fabricas][Depositos];
16
17 //Função Objetivo
18 minimize
19   sum(f in Fabricas, d in Depositos) Custo[f][d]*Transportado[f][d];
20
```

```
21 //Restrições
22 subject to {
23  forall (f in Fabricas)
24     sum(d in Depositos)
25       Transportado[f][d] <= Capacidade[f];
26
27  forall (d in Depositos)
28    sum (f in Fabricas)
29      Transportado [f][d]  >= Demanda[d];
30 };
31
32 //javascript
33 execute RESULTADOS{
34   writeln (" ");
35    for (var f in Fabricas)
36     for (var d in Depositos)
37         if (Transportado[f][d]>0){  //não imprime 0 milhares...
38         writeln (Transportado[f][d]+' '+"carretas sai da fabrica"+
39         ' '+ f +' '+"e são enviadas para o destino"+' '+ d);
40 }
41 }
```

Fonte: a autora.

Os resultados obtidos estão listados na Figura 12.

Figura 12 - CPLEX – resultado do problema do transporte 1

Solução com objetivo 152.535

Nome	Valor
Dados (7)	
Capacidade	[75 125 100]
Custo	[[464 513 654 867] [352 416 690 791] [995 682 388 685]]
Demanda	[80 65 70 85]
Depositos	1..4
Fabricas	1..3
nDepositos	4
nFabricas	3
Variáveis de decisão (1)	
Transportado	[[0 20 0 55] [80 45 0 0] [0 0 70 30]]

« Log de script (solte o código do script aqui para executá-lo)

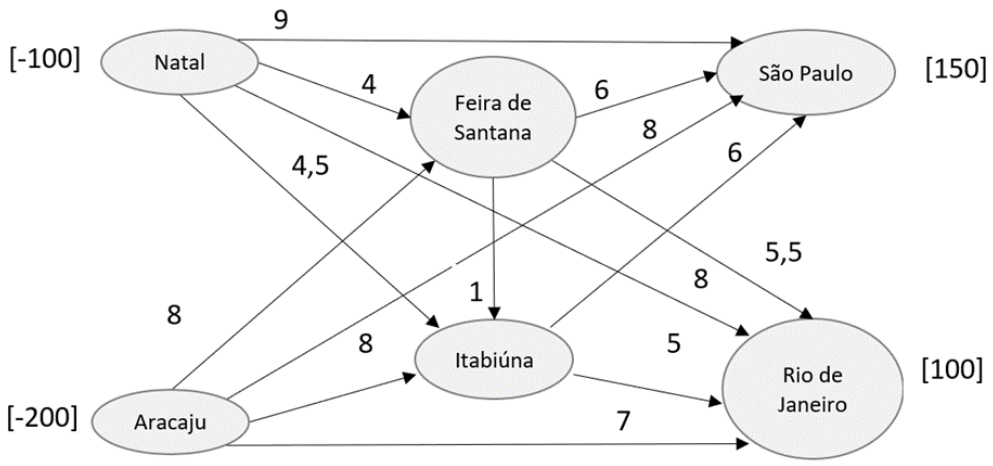
```
// solution (optimal) with objective 152535

20 carretas saem da fabrica 1 e são enviadas para o destino 2
55 carretas saem da fabrica 1 e são enviadas para o destino 4
80 carretas saem da fabrica 2 e são enviadas para o destino 1
45 carretas saem da fabrica 2 e são enviadas para o destino 2
70 carretas saem da fabrica 3 e são enviadas para o destino 3
30 carretas saem da fabrica 3 e são enviadas para o destino 4
```

Fonte: a autora.

3.4 Problema do transporte 2

Exemplo (adaptado de Lachtermacher (2009, p.151): considere o modelo de rede de transporte (Figura 13) em que Natal e Aracaju são fábricas de tratores e São Paulo e Rio de Janeiro são centros consumidores. As capacidades de produção e quantidades demandadas estão indicadas entre colchetes. As setas indicam possíveis rotas e, a cada seta, está associado o respectivo custo unitário de transporte, em Reais. Estabeleça um modelo para determinar as rotas pelas cidades de Feira de Santana e Itabuna para atender aos consumidores.

Figura 13 - Rede de Transporte – problema do transporte 2



Fonte: adaptado de LACHTERMACHER (2009, p.151)

Este exemplo é uma adaptação do exemplo do transporte, conhecido como problema do transporte com transbordo. Neste caso, as quantidades transportadas passam por pontos intermediários, ou de transbordo, entre a origem e o destino. Neste exemplo, Natal e Aracaju são pontos de fornecimento (remetem insumos), Feira de Santana e Itabiúna são pontos de transbordo (recebem e remetem insumos) e São Paulo e Rio de Janeiro são pontos de demanda (recebem insumos).

Verifica-se que o total das quantidades produzidas nas origens ($100 + 200 = 300$) é diferente do total das quantidades requeridas nos destinos ($150 + 100 = 250$). Para modelar este tipo de problema, recorre-se a criação de uma variável fantasma (*dummy*), nesse caso, um destino que receberá a diferença entre demanda e oferta ($300 - 250 = 50$), com custo zero.

Os dados da Tabela 19 concentram as quantidades a serem transportadas, extraídas da Figura *13*, acrescida da variável *dummy*, de forma e igualar a oferta e a demanda. Apresenta, também, a estruturação do problema, considerando as origens

e os transbordos. Nota-se que foi utilizado o recurso do *big* M, ou seja, atribuímos um custo muito alto (999) entre os locais de transbordo, tornando esta rota impeditiva, uma vez que pretendemos minimizar os custos. Esta é uma das formas de modelar o problema, adaptando-o ao modelo do transporte.

Tabela 19 - Parâmetros (custos) – exemplo do problema do transporte 2

Fábrica i (Origem)		Custo de transporte c_{ij}, em Reais, até o depósito j				
		Depósitos j (destino)				
		Transbordos		Destinos		
		Feira de Santana	Itabuna	São Paulo	Rio de Janeiro	*Dummy*
Origens	Natal	4	4,5	9	8	0
	Aracaju	8	8	8	7	0
Transbordos	Feira de Santana	0	1	6	5,5	0
	Itabuna	999	0	6	5,0	0

Fonte: a autora.

Os dados da Tabela 20 concentram as quantidades a ser transportadas, que se deseja encontrar, a partir das restrições de demanda e oferta. Atribui-se ao destino dummy a quantidade 50, para que a oferta seja igual à demanda. O total de unidades dos transbordos (demanda e oferta) é igual a soma das ofertas das origens, de Natal e Aracaju.

Tabela 20 - Parâmetros (quantidades) – exemplo do problema do transporte 2

Fábrica i (Origem)		Quantidade Remetida (por carretas)					s_i número de unidades distribuídas pela origem i (oferta)
		Depósitos j (destino)					
		Transbordos		Destinos			
		Feira de Santana	Itabuna	São Paulo	Rio de Janeiro	*Dummy*	
Origens	Natal	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	100
	Aracaju	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	200
Transbordos	Feira de Santana	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	300
	Itabuna	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	300
d_j: número de unidades recebidas pelo destino j		300	300	150	100	50	

Fonte: a autora.

Deseja-se minimizar a função objetivo custo total de transporte z, em Reais, em que:

m: total de origens (fábricas e transbordo), com $m = 4$.

n: total de destinos (depósitos, transbordo e *dummy*), com $n = 5$.

c_{ij}: custo por unidade distribuída da fábrica i ao depósito j, em Reais.

x_{ij}: variáveis de decisão, que representam a quantidade a ser transportada da fábrica i para o depósito j, com $i = 1,2,3,4$ e $j = 1,2,3,4,5$.

s_i : termos independentes que indicam número de unidades distribuídas pela fábrica i, com $i = 1,2,3,4$.

d_j : número de unidades recebidas pelo depósito $j, \mathrm{com}\ j = 1,2,3,4,5$.

Na forma expandida, o modelo é definido por:

Minimizar $z = 4x_{11} + 4{,}5x_{12} + 9x_{13} + 8x_{14} + 0x_{15} + 8x_{21} + 8x_{22} + 8x_{23} + 7x_{24} + 0x_{25} + 0x_{31} + 1x_{32} + 6x_{33} + 5{,}5x_{34} + 0x_{35} + 999x_{41} + 0x_{42} + 6x_{43} + 5x_{44} + 0x_{45}$

Ou seja,

Minimizar $z = 4x_{11} + 4{,}5x_{12} + 9x_{13} + 8x_{14} + 8x_{21} + 8x_{22} + 8x_{23} + 7x_{24} + +x_{32} + 6x_{33} + 5{,}5x_{34} + 999x_{41} + 6x_{43} + 5x_{44}$

Sujeito àss restrições:

De oferta:

$$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 100$$

$$x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 200$$

$$x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 300$$

$$x_{41} + x_{42} + x_{43} + x_{44} + x_{45} = 300$$

De demanda:

$$x_{11} + x_{21} + x_{31} + x_{41} = 300$$

$$x_{12} + x_{22} + x_{32} + x_{42} = 300$$

$$x_{13} + x_{23} + x_{33} + x_{43} = 150$$

$$x_{14} + x_{24} + x_{34} + x_{44} = 100$$

$$x_{15} + x_{25} + x_{35} + x_{45} = 50$$

De Não negatividade: $x_{ij} \geq 0$ para todo $i = 1,2,3,4$ e $j = 1,2,3,4,5$

Na forma de somatório, o modelo é definido por:

$Minimizar\ z = \sum_{i=1}^{4} \sum_{j=1}^{5} c_{ij}.x_{ij}$

Sujeito às:

$\sum_{j=1}^{5} x_{ij} = s_i$, para $i = 1,2,3,4$

$\sum_{i=1}^{4} x_{ij} = d_j$ para $j = 1,2,3,4,5$

$x_{ij} \geq 0$ para todo i e j

O código digitado no CPLEX consta na Figura 14.

Figura 14 - CPLEX – código do problema do transporte 2

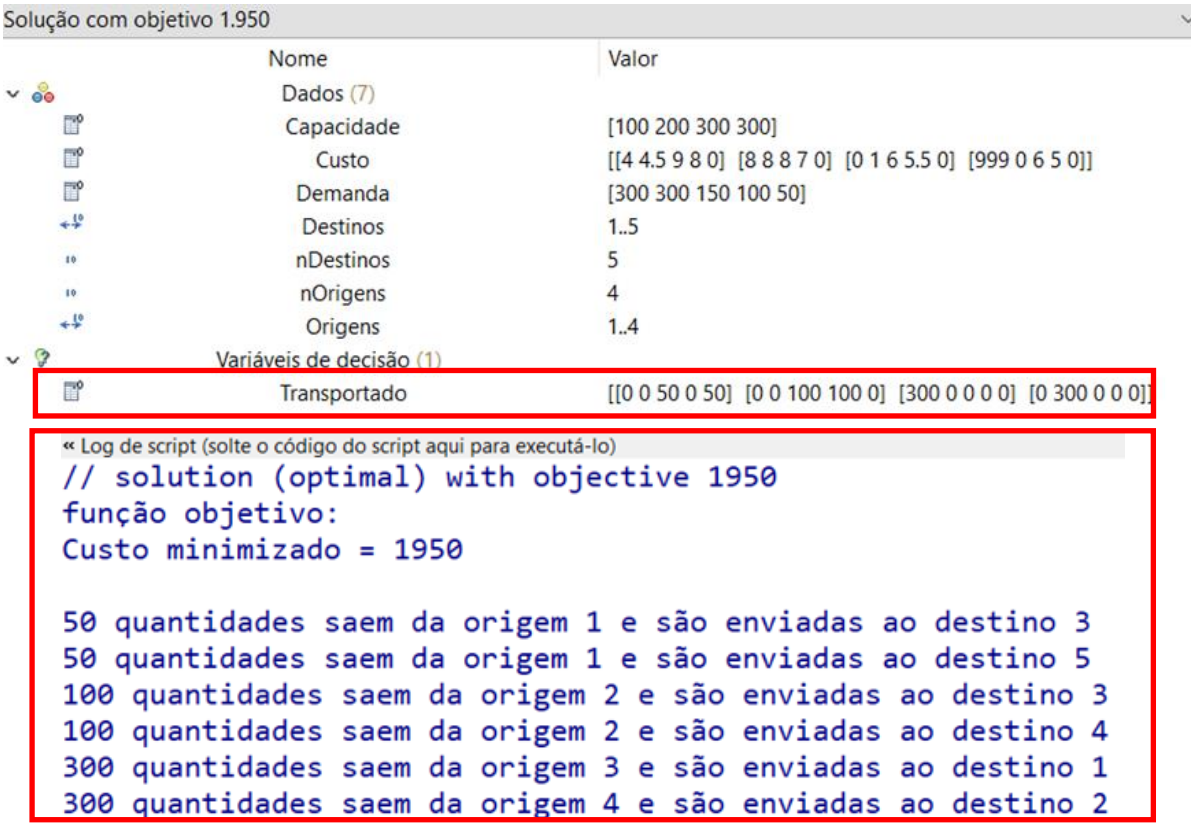
```
//parametros
int nOrigens = 4;
range Origens =1..nOrigens;
int nDestinos = 5;
range Destinos = 1..nDestinos;

float Capacidade[Origens]=[100,200,300,300];        //ofertas das origens
float Demanda[Destinos] = [300,300,150,100,50];     //demanda dos destinos
                                  // d1   d2 d3 d4 d5
float Custo[Origens][Destinos]=[  [4  ,4.5,9, 8  ,0],    //origem 1
                                  [8  ,8  ,8, 7  ,0],    //origem 2
                                  [0  ,1  ,6, 5.5,0],    //origem 3
                                  [999,0  ,6, 5  ,0]     //origem 4
                                 ];
//Variáveis de decisão
dvar float+ Transportado[Origens][Destinos];

//Função Objetivo
minimize
  sum(o in Origens, d in Destinos) Custo[o][d]*Transportado[o][d];

//Restrições
subject to {
 forall (o in Origens)
    sum(d in Destinos)
      Transportado[o][d] == Capacidade[o];

 forall (d in Destinos)
   sum (o in Origens)
     Transportado [o][d] == Demanda[d];
};

//javascript
execute RESULTADOS{
  writeln ("função objetivo:");
  writeln ("Custo minimizado = ", cplex.getObjValue());
  writeln (" ");
   for (var o in Origens)
    for (var d in Destinos)
        if (Transportado[o][d]>0){  //não imprime 0
        writeln (Transportado[o][d]+' '+"quantidades saem da origem"+
        ' '+ o +' '+"e são enviadas ao destino"+' '+ d);
```

```
43
44 }
45 }
```

Fonte: a autora.

Os resultados estão apresentados na Figura 15.

Figura 15 - CPLEX – resultado do problema do transporte 2



Fonte: a autora.

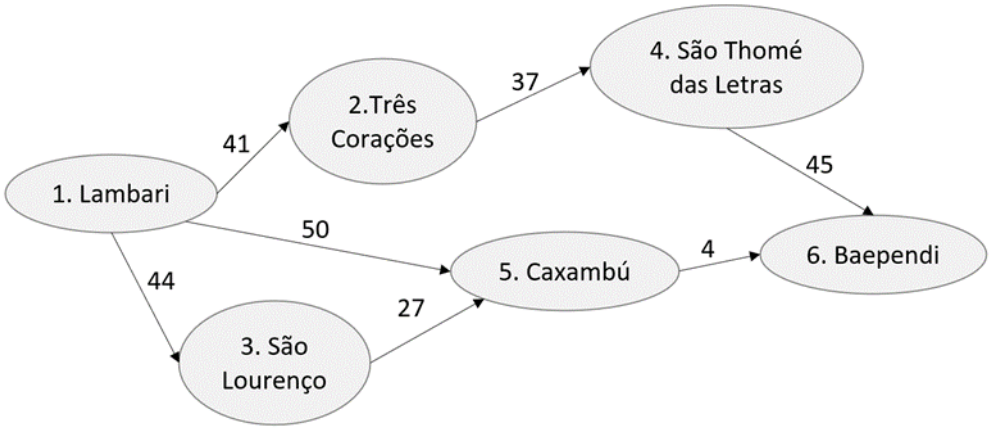
3.5 Problema do menor caminho

O problema do menor caminho é um caso particular do problema de rede, no qual os arcos apresentam as distâncias entre dois pontos (nós). Há mais de uma maneira de partir da origem e chegar ao destino, passando por pontos intermediários, e busca-se uma rota com a menor distância.

Exemplo (adaptado de Lachtermacher (2009, p.140): A fábrica de artigos de decoração, localizada em Lambari (MG), entrega quantidades de peças na cidade de Baependi (MG). A empresa procura um caminho, para que o caminhão de entregas possa fazer de modo a minimizar a distância total percorrida. As distâncias entre as

cidades, em km, que constituem possíveis caminhos estão apresentadas na Figura 16.

Figura 16 - Rede de Transporte – problema do menor caminho



Fonte: adaptado de LACHTERMACHER, 2009, p.141

Neste caso, as variáveis são binárias do tipo x_{ij}, e indicam o sentido da cidade i para a cidade j. Quando a variável assume o valor 1, indica que o trecho deverá ser percorrido e, quando assume o valor 0, indica que o trecho não deverá ser percorrido. A função objetivo é uma função distância percorrida pelo caminhão, em km, entre a origem (Lambari) e o destino (Baependi), a ser minimizada.

Os dados da Tabela 21 apresentam a matriz quadrada de ordem $n = 6$ simétrica com as distâncias entre os vértices, extraídas da Figura 16. Para vértices sem ligação, foi atribuído o valor 999 (*big M*).

Tabela 21 - Parâmetros (distâncias) – exemplo do problema do menor caminho

Cidades i (origem)	Cidades j (destino)					
	1	2	3	4	5	6
1	999	41	44	999	50	999
2	41	999	999	37	999	999
3	44	999	999	999	27	999
4	999	37	999	999	999	45
5	50	999	27	999	999	4
$n = 6$	999	999	999	45	4	999

Fonte: a autora.

Nos dados da Tabela 22, estão as variáveis binárias x_{ij} que se deseja conhecer. A soma de cada linha indica o fluxo de saída no respectivo vértice. A soma de cada

coluna indica o fluxo de entrada no respectivo vértice. Para montar as restrições, para cada vértice, tomamos a diferença entre fluxos de entradas e saídas, que é igualada ao parâmetro fluxo desejado b_i. Para a origem, b_i é igual a -1, para o destino, b_i é igual a 0 e para os vértices intermediários, b_i é igual a 0.

Tabela 22 - Parâmetros (fluxos) – exemplo do problema do menor caminho

								restrições	
Vértices i (origem)	Vértices j (destinos) 1	2	3	4	5	6	Saídas em cada vértice	Fluxo em cada vértice	Fluxo desejado b_i
1	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	$\sum_{j=1}^{6} x_{1j}$	$\sum_{j=1}^{6} x_{i1} - \sum_{j=1}^{6} x_{1j}$	-1
2	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	$\sum_{j=1}^{6} x_{2j}$	$\sum_{j=1}^{6} x_{i2} - \sum_{j=1}^{6} x_{2j}$	0
3	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	$\sum_{j=1}^{6} x_{3j}$	$\sum_{i=1}^{6} x_{i3} - \sum_{j=1}^{6} x_{3j}$	0
4	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	$\sum_{j=1}^{6} x_{4j}$	$\sum_{i=1}^{6} x_{i4} - \sum_{j=1}^{6} x_{4j}$	0
5	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	$\sum_{j=1}^{6} x_{5j}$	$\sum_{i=1}^{6} x_{i5} - \sum_{j=1}^{6} x_{5j}$	0
6	x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	$\sum_{j=1}^{6} x_{6j}$	$\sum_{i=1}^{6} x_{i6} - \sum_{j=1}^{6} x_{6j}$	1
Entradas em cada vértice	$\sum_{i=1}^{6} x_{i1}$	$\sum_{i=1}^{6} x_{i2}$		...		$\sum_{i=1}^{6} x_{i6}$			

Fonte: a autora.

Na forma expandida, o modelo é definido por:

$$\begin{aligned} \textit{Minimizar } z = {} & 999x_{11} + 41x_{12} + 44x_{13} + 999x_{14} + 50x_{15} + 999x_{16} + \\ & 41x_{21} + 999x_{22} + 999x_{23} + 37x_{24} + 999x_{25} + 999x_{26} + \\ & 44x_{31} + 999x_{32} + 999x_{33} + 999x_{34} + 27x_{35} + 999x_{36} + \\ & 999x_{41} + 37x_{42} + 999x_{43} + 999x_{44} + 999x_{45} + 45x_{46} + \\ & 50x_{51} + 999x_{52} + 27x_{53} + 999x_{54} + 999x_{55} + 4x_{56} + 999x_{61} + \end{aligned}$$

$$999x_{62} + 999x_{63} + 45x_{64} + 4x_{65} + 999x_{66}$$

Sujeito às restrições:

$$x_{11} + x_{21} + x_{31} + x_{41} + x_{51} + x_{61} - (x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16}) = -1$$

$$x_{12} + x_{22} + x_{32} + x_{42} + x_{52} + x_{62} - (x_{21} + x_{22} + x_{23} + x_{24} + x_{25} + x_{26}) = 0$$

$$x_{13} + x_{23} + x_{33} + x_{43} + x_{53} + x_{63} - (x_{31} + x_{32} + x_{33} + x_{34} + x_{35} + x_{35} + x_{36}) = 0$$

$$x_{14} + x_{24} + x_{34} + x_{44} + x_{54} + x_{64} - (x_{41} + x_{42} + x_{43} + x_{44} + x_{45} + x_{46}) = 0$$

$$x_{15} + x_{25} + x_{35} + x_{45} + x_{55} + x_{65} - (x_{51} + x_{52} + x_{53} + x_{54} + x_{55} + x_{56}) = 0$$

$$x_{16} + x_{26} + x_{36} + x_{46} + x_{56} + x_{66} - (x_{61} + x_{62} + x_{63} + x_{64} + x_{65} + x_{66}) = 1$$

$x_{i,j} = \begin{cases} 0 \\ 1 \end{cases}$, para todo $i = 1,2,3,4,5,6$ e $j = 1,2,3,4,5,6$

Na forma de somatório, o modelo é definido por:

Minimizar $z = \sum_{i=1}^{6} \sum_{j=1}^{6} c_{ij}.x_{ij}$

Sujeito às:

Restrições funcionais:

$$\sum_{i=1}^{n=6} x_{ij} (para\ j = 1, \dots, 6) - \sum_{j=1}^{n=6} x_{ij} = b_i\ (para\ i = 1, \dots, 6)$$

$x_{ij} \in \{0,1\}$

Deseja-se minimizar a função objetivo distância z, em km, em que:

n: número de vértices (cidades)

i: origens, $i = 1\dots,6$

j: destinos, $j = 1,\dots,6$

c_{ij}: distância entre a origem i e o destino j, em km.

b_i: fluxo no vértice i, sendo: $b_i := -1$ (origem), $b_i := 0$ (vértices intermediários) e $b_i := 1$ (destino).

x_{ij}: variáveis de decisão que indicam se a aresta $i \to j$ fará parte da decisão. Se $x_{\mathrm{ij}} = 1$, então a aresta faz parte do caminho mínimo e, se $x_{\mathrm{ij}} = 0$, então a aresta não faz parte da solução.

O código inserido no CPLEX está apresentado na Figura *17*.

Figura 17 - CPLEX – código do problema do menor caminho

```
//parâmetros
int nNos = 6;
range Nos = 1..nNos;
float b[Nos] = [-1,0,0,0,0,1];  //origem: -1, destino: 1 e demais vértices:0
                  // Nós 1    2   3  4   5   6
float Dist[Nos][Nos]=[[999, 41, 44,999, 50,999], // Nó 1
                      [ 41,999,999, 37,999,999], // Nó 2
                      [ 44,999,999,999, 27,999], // Nó 3
                      [999, 37,999,999,999, 45], // Nó 4
                      [ 50,999, 27,999,999,  4], // Nó 5
                      [999,999,999, 45,  4,999]  // Nó 6
                     ];

//Variáveis de decisão
dvar boolean x[Nos][Nos];

//Função Objetivo
minimize
  sum(i in Nos, j in Nos) Dist[i][j]*x[i][j];

//Restrições
subject to{
    forall (i in Nos)
      sum(j in Nos) x[j][i] - sum(j in Nos) x[i][j] == b[i];
}

//javascript
execute RESULTADOS{
  writeln ("função objetivo:");
  writeln ("Distância minimizada = ", cplex.getObjValue());
  writeln (" ");
  for (var i in Nos)
   for (var j in Nos){
      if (x[i][j]==1)
        writeln("x["+ i + "]["+ j +"] resultado: "+x[i][j]);
    }
}
```

Fonte: a autora.

Os resultados estão apresentados na Figura 18. Observando a matriz x_{ij}, verifica-se que o menor caminho é feito por 1 – 5 – 6 (variável binária = 1).

Figura 18 - CPLEX – resultado do problema do menor caminho

Nome	Valor
Dados (4)	
b	[-1 0 0 0 0 1]
Dist	[[999 41 44 999 50 999] [41 999 999 37 999 999] [44 999 999 999 27 999] [99...
nNos	6
Nos	1..6
Variáveis de decisão (1)	
x	[[0 0 0 0 1 0] [0 0 0 0 0 0] [0 0 0 0 0 0] [0 0 0 0 0 0] [0 0 0 0 0 1] [0 0 0 0 0 0]]

Fonte: a autora.

Os resultados podem ser formatados com auxílio do Javascript, permitindo melhor visualização (Figura 19).

Figura 19 - CPLEX – resultado do problema do menor caminho

```
// solution (optimal) with objective 54
função objetivo:
Distância minimizada = 54

x[1][5] resultado: 1
x[5][6] resultado: 1
```

Fonte: a autora.

3.6 Problema do Planejamento Agregado

Exemplo adaptado de Lachtermacher (2009, p.146): Uma fábrica de eletrodomésticos deseja realizar o escalonamento de sua produção de liquidificadores para os próximos quatro meses. A fábrica pode produzir, mensalmente, em jornada normal, 150.000 unidades a um custo unitário de R$ 15,00. Com o pagamento de horas extras, a capacidade mensal de produção da fábrica pode ser aumentada em 50.000 liquidificadores a um custo de produção unitário de R$ 22,00 (somente os adicionais). Há a possibilidade de armazenagem (ilimitada) de unidades de um mês para outro a um custo unitário mensal de R$ 3,00. Sabendo-se que as demandas de liquidificadores para os próximos quatro meses são de 120.000, 200.000, 120.000 e 180.000, minimize a função custo total de produtos produzidos (produção e armazenagem). Os custos unitários, por tipo de mão de obra e por períodos, são consolidados nos dados da Tabela 23.

Tabela 23 - Parâmetros (custos)–exemplo problema do planejamento agregado

Custos Unitários por regime de contratação i e estoque	Custos unitários por regime i no período j. períodos j				
		$mês1$	$mês2$	$mês3$	$mês4$
$Regime\ normal\ C_1$		22,00	22,00	22,00	22,00
$Regime\ de\ hora\ extra\ C_2$		15,00	15,00	15,00	15,00
	Custos unitários $custoE_j$				
	$anterior$	$mês1$	$mês2$	$mês3$	$mês4$
$Estoque\ E_j$	3,00	3,00	3,00	3,00	3,00

Fonte: a autora.

Os dados da Tabela 24 apresentam as ofertas, por tipo de mão de obra e a demanda em cada período.

Tabela 24 - Parâmetros – exemplo problema do planejamento agregado

Oferta/Demanda por tipo de mão de obra i	Oferta por regime i e períodos j			
	$mês1$	$mês2$	$mês3$	$mês4$
$Oferta\ Regime\ normal\ \ O_1$	150000	150000	150000	150000
$Oferta\ Regime\ hora\ extra\ \ O_2$	50000	50000	50000	50000
$Demanda_j$	120000	200000	120000	180000

Fonte: a autora.

Os dados da Tabela 25 apresentam as variáveis de decisão, em cada período, indicando a produção a ser realizada em regime normal, de mão de obra e estoque.

Tabela 25 - Variáveis de decisão–exemplo problema do planejamento agregado

Produção por tipo i e estoque	Quantidade produzida do regime i no período j				
		$mês1$	$mês2$	$mês3$	$mês4$
$Produção\ Regime\ normal\ \ P_1$		P_{11}	P_{12}	P_{13}	P_{14}
$Produção\ Regime\ hora\ extra\ \ P_2$		P_{21}	P_{22}	P_{23}	P_{24}
	Quantidade armazenada no período j				
	anterior	$mês1$	$mês2$	$mês3$	$mês4$
$Estoque\ E_j$	$E_0 = 0$	E_1	E_2	E_3	E_4

Fonte: a autora.

Deseja-se minimizar a função objetivo custo total de produção de liquidificadores z, em Reais, em que:

m: total de regimes de horas, com $m = 2$ (horas normais e horas extras), $i = 1,2$.

n: total de períodos, com $n = 4$ meses, variando de $j = 1,2,3,4$.

C_{1j}: custo por unidade produzida por mão de obra normal no período j.

C_{2j}: custo por unidade produzida por mão de obra hora extra no período j.

$custoE_j$: custo por unidade estocada no período j. Matriz com $j + 1$ colunas, para informação do custo de estoque no período $j - 1$ (estoque inicial).

O_{1j} : oferta ou capacidade de produção no regime de horas normal, no período j.

O_{2j} oferta ou capacidade de produção dos liquidificadores no regime de horas extras, no período j.

$Demanda_j$: demanda de produção dos liquidificadores, no período j.

P_{1j}: variáveis de decisão, que indicam a quantidade produzida no regime de horas normal, no período j.

P_{2j} : variáveis de decisão, que indicam a quantidade produzida no regime de horas extras, no período j.

E_j: variáveis de decisão, que indicam a quantidade estocada, no período j, após produção e atendimento da demanda. O estoque inicial é um parâmetro a ser informado, no caso, $E_0 = 0$. Para cada período j, tem-se:

$E_j = E_{j-1} + P_{1j} + P_{2j} - demanda_J$

Na forma expandida, tem-se:

Minimizar

$22P_{11} + 22P_{12} + 22P_{13} + 22P_{14} + 15P_{21} + 15P_{22} + 15P_{23} + 15P_{24} + 3E_0 + 3E_1 + 3E_2 + 3E_3 + 3E_4 + 3E_5$

Com $E_0 = 0$

Sujeito às:

Restrições funcionais:

$P_{11} \leq 150000$

$P_{12} \leq 150000$

$P_{13} \leq 150000$

$P_{14} \leq 150000$

$P_{21} \leq 50000$

$P_{22} \leq 50000$

$P_{23} \leq 50000$

$P_{24} \leq 50000$

$E_0 + P_{11} + P_{21} - 120000 = E_1$

$E_1 + P_{12} + P_{22} - 200000 = E_2$

$$E_2 + P_{13} + P_{23} - 120000 = E_3$$

$$E_3 + P_{14} + P_{24} - 180000 = E_4$$

Restrições de não negatividade:

$$P_{11}, P_{12}, P_{13}, P_{14} \geq 0$$

$$P_{21}, P_{22}, P_{23}, P_{24} \geq 0$$

$$E_0, E_1, E_2, E_3, E_4, E_5 \geq 0$$

Na forma de somatório, o modelo é definido por:

$Minimizar\ z = \sum_{i=1}^{2} \sum_{j=1}^{4} P_{ij}.C_{ij} + \sum_{j=0}^{4} custoE_j.E_j$

Sujeito às:

$E_0 = 0$ (estoque no período $j-1$)

Restrições funcionais:

$P_{ij} \leq O_{ij}$, para todo i e j

$E_{j-1} + \sum_{i=0}^{m} P_{ij} - demanda_j = E_j$, para todo i e j

Restrições de não negatividade:

$P_{11}, \dots, P_{14}, P_{21}, \dots, P_{24}, E_j \geq 0$ para todo j

O código foi inserido no CPLEX e apresentado na Figura *20*.

Figura 20 - CPLEX – código do problema do menor caminho

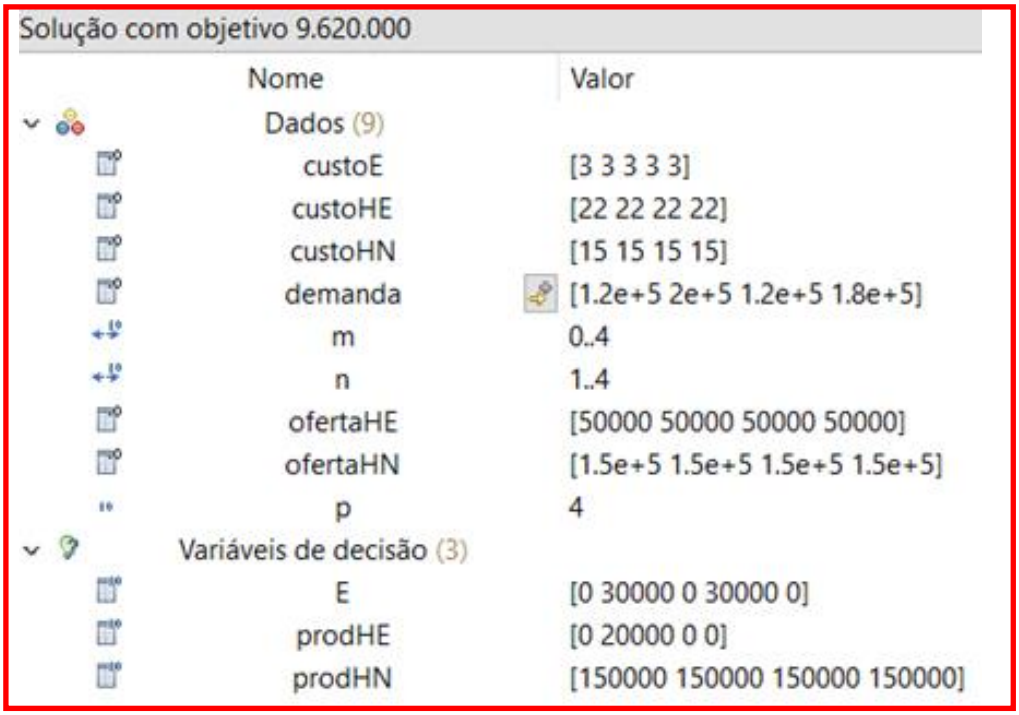
```
1 //parâmetros
2 int p = 4;    //períodos
3 range n = 1..p;
4 range m = 0..p;                                    //estoque e0=0 + períodos
5 float custoHN[n]=[15,15,15,15];                    //custo HN
6 float custoHE[n]=[22,22,22,22];                    //custo HE
7 float custoE[m]=[3,3,3,3,3];                       //custo estoque
8 float ofertaHN[n]=[150000,150000,150000,150000];   //oferta HN
9 float ofertaHE[n]=[50000,50000,50000,50000];       //oferta HE
10 float demanda[n]=[120000,200000,120000,180000];   //demanda no período
11
12 //variáveis de decisão
13 dvar int+ prodHN[n];  //produção em regime Normal
14 dvar int+ prodHE[n];  //produção em regime HoraExtra
15 dvar int+ E[m];   //produção estocada (estoque anterior: E0=0)
16
```

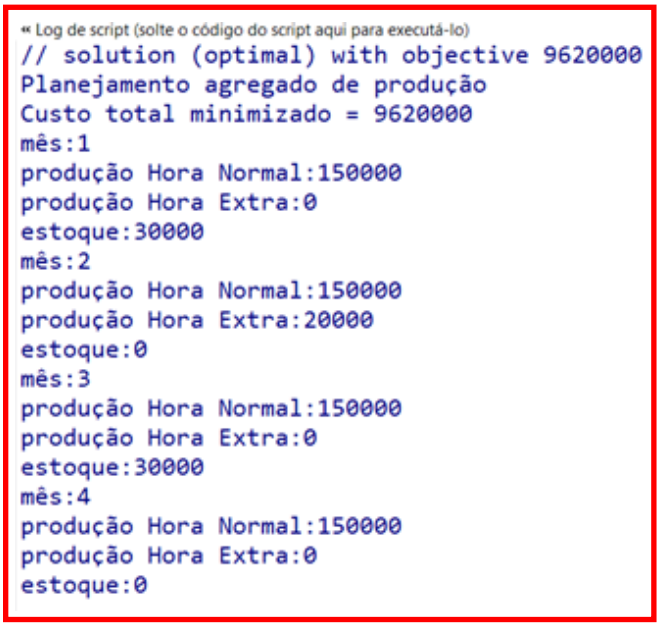
```
17 //função objetivo
18 minimize
19   sum(j in n)(custoHN[j]*prodHN[j] + custoHE[j]*prodHE[j]) + sum(k in m) (custoE[k]*E[k]
20
21 //restrições
22 subject to {
23   E[0]==0;  //estoque inicial e0=0
24   forall (j in n)
25     E[j-1] + prodHN[j] + prodHE[j] - E[j]  == demanda[j];
26   forall (j in n)
27     prodHN[j]<=ofertaHN[j];
28   forall (j in n)
29     prodHE[j]<=ofertaHE[j];
30 }
31
32 //javascript
33 execute RESULTADOS{
34   writeln("Planejamento agregado de produção");
35   writeln ("Custo total minimizado = ", cplex.getObjValue());
36   for(var j in n){
37     writeln("mês:",j);
38     writeln("produção Hora Normal:", prodHN[j]);
39     writeln("produção Hora Extra:", prodHE[j]);
40     writeln("estoque:", E[j]);
41   }
42 }
```

Fonte: a autora.

Os resultados estão apresentados na Figura 21.

Figura 21 - CPLEX – resultado do problema do menor caminho





```
« Log de script (solte o código do script aqui para executá-lo)
// solution (optimal) with objective 9620000
Planejamento agregado de produção
Custo total minimizado = 9620000
mês:1
produção Hora Normal:150000
produção Hora Extra:0
estoque:30000
mês:2
produção Hora Normal:150000
produção Hora Extra:20000
estoque:0
mês:3
produção Hora Normal:150000
produção Hora Extra:0
estoque:30000
mês:4
produção Hora Normal:150000
produção Hora Extra:0
estoque:0
```

Fonte: a autora.

3.7 Problema do corte de barras 1

Exemplo adaptado de Prado (2016, p.131): supondo que barras de 6 m de comprimento devam ser convenientemente cortadas em barras menores. Deseja-se cortar 50 barras de 2 m, 60 barras de 3 m e 90 barras de 4 m. O corte deve ser feito de tal forma que se minimizem as perdas, ou seja, as sobras do corte.

Inicialmente são definidos os possíveis padrões de corte das barras de comprimento 6 m ($L = 6$), considerando as dimensões das peças ($l_1 = 2$, $l_2 = 3$ e $l_3 = 4$, em metros) e as respectivas perdas, conforme os dados da Tabela 26. As perdas são as sobras da barra original, após realizar os cortes das peças.

Tabela 26 - Parâmetros (padrões de corte) – problema do corte de barras

Dimensões l_i das peças, em m	Quantidade de peças c_{ij} de dimensão l_i, referente ao padrão de corte j							
	$Padrão$ 1	$Padrão$ 2	$Padrão$ 3	$Padrão$ 4	$Padrão$ 5	$Padrão$ 6	$Padrão$ 7	$Padrão$ 8
2	0	0	1	1	1	0	3	2
3	0	1	0	0	1	2	0	0
4	1	0	0	1	0	0	0	0
Perda p_j	2	3	4	0	1	0	0	2

Fonte: a autora.

Os dados da Tabela 27 apresentam as variáveis de decisão que representam as quantidades de barras utilizadas em cada padrão de corte.

Tabela 27 - Variáveis de decisão – problema do corte de barras

variáveis	Quantidades de barras cortadas em cada padrão de corte j							
	$Padrão$ 1	$Padrão$ 2	$Padrão$ 3	$Padrão$ 4	$Padrão$ 5	$Padrão$ 6	$Padrão$ 7	$Padrão$ 8
x_j	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8

Fonte: a autora.

Os dados da Tabela 28 apresentam as restrições de demanda.

Tabela 28 - Restrições – problema do corte de barras

Dimensões l_i das peças, em metros	Quantidades por padrão j ($n = 8$)								d_i demanda das peças de dimensão l_i
	$Padrão$ 1	$Padrão$ 2	$Padrão$ 3	$Padrão$ 4	$Padrão$ 5	$Padrão$ 6	$Padrão$ 7	$Padrão$ 8	
2	$0x_1$	$0x_2$	$1x_3$	$1x_4$	$1x_5$	$0x_6$	$3x_7$	$2x_8$	50
3	$0x_1$	$1x_2$	$0x_3$	$0x_4$	$1x_5$	$2x_6$	$0x_7$	$0x_8$	60
4 ($m = 3$)	$1x_1$	$0x_2$	$0x_3$	$1x_4$	$0x_5$	$0x_6$	$0x_7$	$0x_8$	90

Fonte: a autora.

Considerando que se deseja minimizar a função objetivo perda total z, no modelo expandido, tem-se:

$$Minimizar\ z = p_1x_1 + p_2x_2 + p_3x_3 + p_4x_4 + p_5x_5 + p_6x_6 + p_7x_7 + p_8x_8$$

$$z = 2x_1 + 3x_2 + 4x_3 + 0x_4 + 1x_5 + 0x_6 + 0x_7 + 2x_8$$

$$z = 2x_1 + 3x_2 + 4x_3 + x_5 + 2x_8$$

Sujeito às:

Restrições funcionais:

$$0x_1 + 0x_2 + 1x_3 + 1x_4 + 1x_5 + 0x_6 + 3x_7 + 2x_8 = 50 \rightarrow x_3 + x_4 + x_5 + 3x_7 + 2x_8 = 50$$

$$0x_1 + 1x_2 + 0x_3 + 0x_4 + 1x_5 + 2x_6 + 0x_7 + 0x_8 = 60 \rightarrow x_2 + x_5 + 2x_6 = 60$$

$$1x_1 + 0x_2 + 0x_3 + 1x_4 + 0x_5 + 0x_6 + 0x_7 + 0x_8 = 90 \rightarrow x_1 + x_4 = 90$$

Restrições de não negatividade:

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7\ e\ x_8 \text{ inteiro e maior que zero}$$

Em que:

L: tamanho original da barra, igual a 6 m.

l_i : dimensões de peças menores cortadas da barra de tamanho $L = 6$, ou seja, $l_1 = 2m$, $l_2 = 3m$, e $l_3 = 4m$.

m: total de tipos de peças menores, ou seja, $m = 3$.

n: total de padrões de corte, ou seja, $n = 8$.

c_{ij}: quantidade de peças de dimensão l_i em cada padrão de corte j.

p_j : perda no padrão de corte j.

d_i : demanda de peça com dimensão l_i .

x_i : quantidade de barras cortadas por padrão de corte j.

o modelo na forma de somatório é dado por:

$Minimizar\ z = \sum_{j=1}^{8} p_j x_j$

Sujeito às:

Restrições funcionais:

$\sum_{j=1}^{8} c_{ij} x_j = d_i$, para $i = 1,2,3$

Restrições de não negatividade:

x_j inteiro e maior que zero para todo j

O código foi inserido no CPLEX e apresentado na Figura 22.

Figura 22 - CPLEX – código do problema do corte de barras

```
1  //mininizar perda
2  //parâmetros
3  int pc = 8;       //padrões de corte
4  range n = 1..pc;
5  int itens = 3;    //tipos de peças
6  range m = 1..itens;
7  int L = 6; //tamanho da barra
8   //padrões de cortes(cij) 1 2 3 4 5 6 7 8
9  int c[m][n]=            [[0,0,1,1,1,0,3,2], //peça 2m
10                          [0,1,0,0,1,2,0,0], //peça 3m
11                          [1,0,0,1,0,0,0,0]  //peça 4m
12                         ];
13
14 int l[m]=[2,3,4];           //dimensões das peças
15 int d[m]=[50,60,90];        //demanda das respctivas peças
16 int p[n]=[2,3,4,0,1,0,0,2]; //perdas por padrão de corte
17
```

```
18 //parâmetros (para javascript)
19 int qtde;
20 int metros;
21
22 //variáveis de decisão
23 dvar int+ x[n];
24
25 //função objetivo
26 minimize
27   sum(j in n) p[j]*x[j];
28
29 //restrições
30 subject to {
31   forall (i in m)
32     sum(j in n) c[i][j]*x[j] == d[i];
33 }
34
35 //javascript
36 execute RESULTADOS{
37   writeln("Perdas (em metros):");
38   writeln ("Perda = ", cplex.getObjValue());
39   writeln("Quantidade total de barras cortadas:");
40   for (var j in n)
41     qtde = qtde + x[j];
42     writeln(qtde);
43   writeln("Metros necessários (qtde * L):");
44   for (var j in n)
45     metros = qtde*L;
46     writeln(metros);
47   writeln("Quantidade de barras por padrão de corte:");
48   for(var j in n){
49     writeln("padrão:",j);
50     writeln("x["+ j + "]:", x[j]+ " unidades");
51   }
52   writeln("dimensões das peças:");
53     for (var i in m)
54     writeln("tamanho (metros):",l[i]);
55 }
```

Fonte: a autora.

Os resultados estão apresentados na Figura *23*.

Figura 23 - CPLEX – resultado do problema do corte de barras

Solução com objetivo 80

Nome	Valor
Dados (11)	
c	[[0 0 1 1 1 0 3 2] [0 1 0 0 1 2 0 0] [1 0 0 1 0 0 0 0]]
d	[50 60 90]
itens	3
L	6
l	[2 3 4]
m	1..3
metros	720
n	1..8
p	[2 3 4 0 1 0 0 2]
padraoc	8
qtde	120
Variáveis de decisão (1)	
x	[40 0 0 50 0 30 0 0]

```
Perdas (em metros):
Perda = 80
Quantidade total de barras cortadas:
120
Metros necessários (qtde * L):
720
Quantidade de barras por padrão de corte:
padrão:1
x[1]:40 unidades
padrão:2
x[2]:0 unidades
padrão:3
x[3]:0 unidades
padrão:4
x[4]:50 unidades
padrão:5
x[5]:0 unidades
padrão:6
x[6]:30 unidades
padrão:7
x[7]:0 unidades
padrão:8
x[8]:0 unidades
dimensões das peças:
tamanho (metros):2
tamanho (metros):3
tamanho (metros):4
```

Fonte: a autora.

3.8 Problema do corte de barras 2

Considerando o exemplo anterior, adaptado de Prado (2016, p.131), pretende-se minimizar a função objetivo quantidade de barras z,usadas para atender a demanda.

O modelo na forma expandida é dado por:

$Minimizar\ z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$

As restrições funcionais e de não-negatividade são as mesmas apresentadas no item anterior, ou seja,

Restrições funcionais:

$$0x_1 + 0x_2 + 1x_3 + 1x_4 + 1x_5 + 0x_6 + 3x_7 + 2x_8 = 50 \rightarrow x_3 + x_4 + x_5 + 3x_7 + 2x_8 = 50$$

$$0x_1 + 1x_2 + 0x_3 + 0x_4 + 1x_5 + 2x_6 + 0x_7 + 0x_8 = 60 \rightarrow x_2 + x_5 + 2x_6 = 60$$

$$1x_1 + 0x_2 + 0x_3 + 1x_4 + 0x_5 + 0x_6 + 0x_7 + 0x_8 = 90 \rightarrow x_1 + x_4 = 90$$

Restrições de não negatividade:

$x_1, x_2, x_3, x_4, x_5, x_6, x_7\ e\ x_8$ inteiro e maior que zero.

O modelo em forma de somatório é dado por:

$Minimizar\ z = \sum_{j=1}^{8} x_j$

Sujeito às:

Restrições funcionais:

$\sum_{j=1}^{8} c_{ij} x_j = d_i$, para $i = 1,2,3$

Restrições de não negatividade:

x_j inteiro maior que zero, para todo j

O código foi inserido no CPLEX e apresentado na Figura 24.

Figura 24 - CPLEX – código do problema do corte de barras

```
//mininizar quantidades de barras
//parâmetros
int padraoc = 8;          //padrões de corte
range n = 1..padraoc;
int itens = 3;            //tipos de peças
range m = 1..itens;

int L = 6; //tamanho da barra
 //padrões de cortes (cij) 1 2 3 4 5 6 7 8
int c[m][n]=            [[0,0,1,1,1,0,3,2], //peça 2m
                         [0,1,0,0,1,2,0,0], //peça 3m
                         [1,0,0,1,0,0,0,0]  //peça 4m
                        ];
int l[m]=[2,3,4];      //dimensões das peças
int d[m]=[50,60,90];  //demanda das respectivas peças
int p[n]=[2,3,4,0,1,0,0,2]; //perdas por padrão de corte

//parâmetros (para javascript)
int qtde;
int metros;
int total;

//variáveis de decisão
dvar int+ x[n];

//função objetivo
minimize
  sum(j in n) x[j];

//restrições
subject to {
  forall (i in m)
    sum(j in n) c[i][j]*x[j] == d[i];
}
```

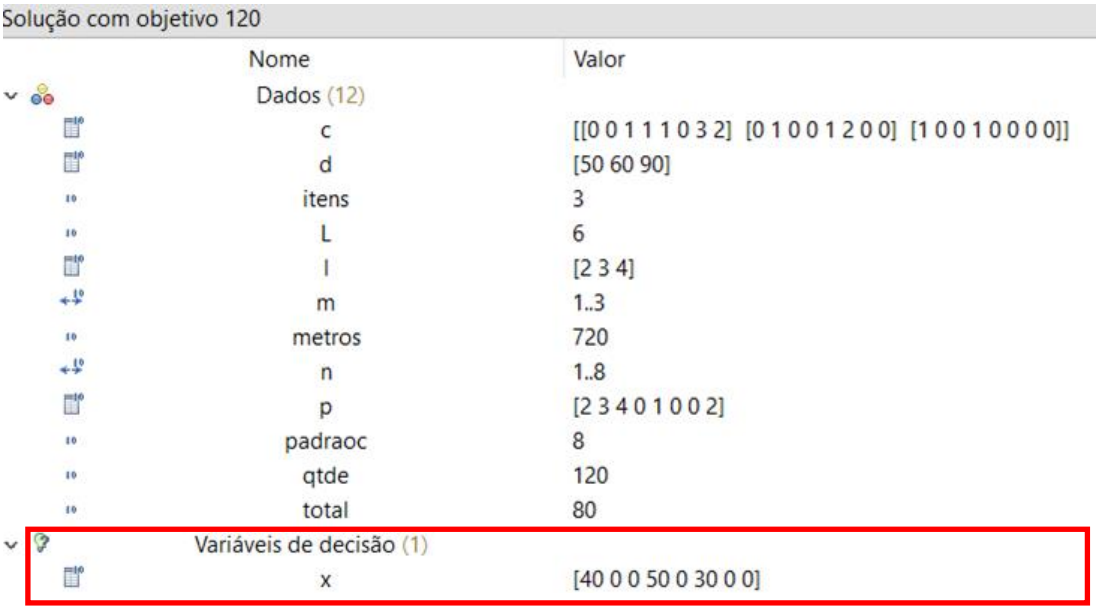
```
36 //javascript
37 execute RESULTADOS{
38   writeln("Quantidade de barras:");
39   writeln ("Qtde = ", cplex.getObjValue());
40   writeln("Metros necessários (qtde * L):");
41   for (var j in n)
42     qtde = qtde + x[j];
43     metros = qtde*L;
44     writeln(metros);
45   writeln("Total de perda (em metros):");
46   for (var j in n)
47     total = total + p[j]*x[j];
48     writeln(total);
49   writeln("Quantidade de barras por padrão de corte:");
50   for(var j in n){
51     writeln("padrão:",j);
52     writeln("x["+ j + "]:", x[j]+ " unidades");
53   }
54   writeln("dimensões das peças:");
55    for (var i in m)
56     writeln("tamanho (metros):",l[i]);
57 }
```

Fonte: a autora.

Os resultados estão apresentados na Figura 25.

Nota-se que o total de perdas, minimizando a quantidade de barras, é o mesmo obtido na otimização anterior (80 metros).

Figura 25 - CPLEX – resultado do problema do corte de barras



```
Quantidade de barras:
Qtde = 120
Metros necessários (qtde * L):
720
Total de perda (em metros):
80
Quantidade de barras por padrão de corte:
padrão:1
x[1]:40 unidades
padrão:2
x[2]:0 unidades
padrão:3
x[3]:0 unidades
padrão:4
x[4]:50 unidades
padrão:5
x[5]:0 unidades
padrão:6
x[6]:30 unidades
padrão:7
x[7]:0 unidades
padrão:8
x[8]:0 unidades
dimensões das peças:
tamanho (metros):2
tamanho (metros):3
tamanho (metros):4
```

Fonte: a autora.

3.9 Problema da mochila 1

Exemplo: Um viajante deseja escolher para levar em sua mochila, com capacidade de 20 kg, alguns dos cinco itens listados, de modo a maximizar o valor dos itens escolhidos. Informe quais itens o viajante deverá escolher.

Os dados da Tabela 29 apresentam os parâmetros do exemplo.

Tabela 29 - Parâmetros – problema da mochila 1

Itens j	$Peso\ a_j$	$Valor\ c_j$	$Mochila$
			x_j
1	7,5	112	x_1
2	2,6	98	x_2
3	9,5	67	x_3
4	3,2	135	x_4
5 $(n = 5)$	5,8	22	x_5
Peso Mochila			b = 20

Fonte: a autora.

O modelo na forma expandida é dado por:

$$Maximizar\ \ z = 112x_1 + 98x_2 + 67x_3 + 135x_4 + 22x_5$$

Sujeito às:

Restrições funcionais:

$$7{,}5x_1 + 2{,}6x_2 + 9{,}5x_3 + 3{,}2x_4 + 5{,}8x_5 \leq 20$$

Restrições de domínio:

$$x_1, x_2, x_3, x_4, x_5 = \{0{,}1\}$$

Em que:

n: número de itens, com $n = 5$.

a_j: peso do item j.

c_j: valor do item j.

b : tamanho ou capacidade da mochila, com $b = 20\ kg$.

x_j : variável de decisão binária, sendo igual a 1, se o item for selecionado e igual a 0, se o item não for selecionado.

O modelo na forma de somatório é dado por:

$$Maximizar\ \ z = \sum_{j=1}^{5} c_j x_j$$

Sujeito às:

Restrições funcionais:

$$\sum_{j=1}^{5} a_j x_j \leq 20$$

Restrições de domínio:

$$x_j = \{0{,}1\}, \text{para}\ \ j = 1, \ldots, 5$$

O algoritmo para resolução deste problema está listado na Figura 26.

Figura 26 - CPLEX – código do problema da mochila 1

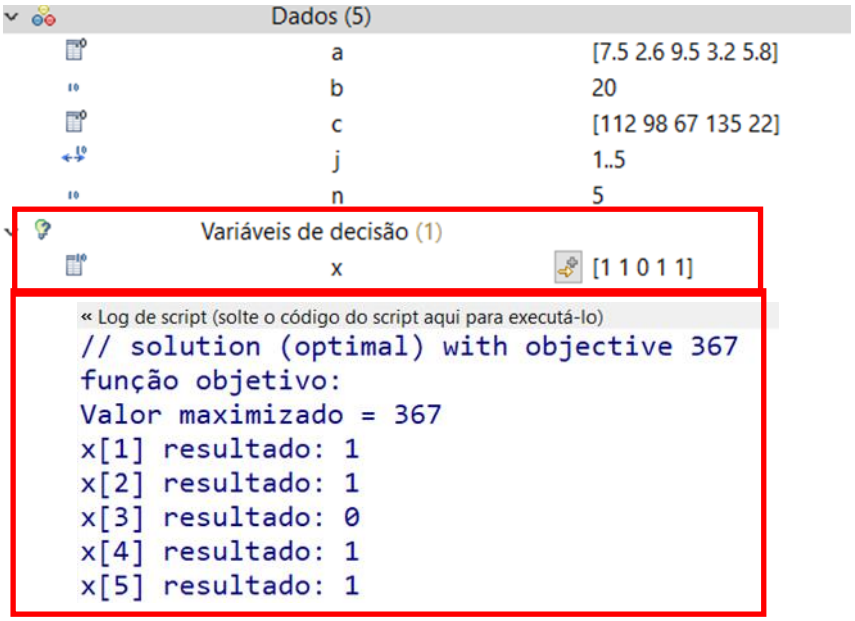
```
1  //parâmetros
2  int n = 5;       //número de itens
3  range j = 1..n;
4  int b = 20;      //capacidade da mochila (kg)
5  float a[j]=[7.5,2.6,9.5,3.2,5.8]; //peso  de cada item
6  float c[j]=[112,98,67,135,22];    //valor de cada item
7
8  //Variáveis de decisão (0 ou 1)
9  dvar boolean x[j];
10
```

```
11 //Função Objetivo
12 maximize
13   sum(j in j) c[j]*x[j];
14
15 //Restrições
16 subject to {
17   forall(k in j)
18     sum(k in j)
19      a[k]*x[k] <= b;
20 };
21
22 //javascript
23 execute RESULTADOS{
24   writeln ("função objetivo:");
25   writeln ("Valor maximizado = ", cplex.getObjValue());
26   for (var k in j)
27      writeln("x["+ k +"]"+ " resultado: "+x[k]);
28
29 }
30
```

Fonte: a autora.

Os resultados estão apresentados na Figura 27.

Figura 27 - CPLEX – resultado do problema da mochila



Fonte: a autora.

3.10 Problema da mochila 2

Este exemplo é uma variação do problema da mochila, mas considerando que os objetos podem ser inseridos em duas mochilas.

Exemplo: Um viajante deseja escolher para levar em suas duas mochilas, com capacidades 20 kg e 8kg, alguns dos oito itens listados, de modo a maximizar o valor dos itens escolhidos. Informe quais itens o viajante deve escolher.

Os dados da Tabela 30 apresentam os parâmetros do exemplo.

Tabela 30 - Parâmetros – problema da mochila 2

Itens j	$Peso\ a_j$	$Valor\ c_j$	$Mochila\ 1$	$Mochila\ 2\ (m=2)$
			x_{ji}	x_{ji}
1	2,5	112	x_{11}	x_{12}
2	4,5	98	x_{21}	x_{22}
3	9,5	132	x_{31}	x_{32}
4	3,2	135	x_{41}	x_{42}
5	5,8	22	x_{51}	x_{52}
6	4,0	175	x_{61}	x_{62}
7	6,0	90	x_{71}	x_{72}
8 $(n=8)$	1,3	145	x_{81}	x_{82}
Peso Mochila i			$b_1 = 20$	$b_2 = 8$

Fonte: a autora.

O modelo na forma de expandida é dado por:

Maximizar

$$z = c_1x_{11} + c_2x_{21} + c_3x_{31} + c_4x_{41} + c_5x_{51} + c_6x_{61} + c_7x_{71} + c_8x_{81} + c_1x_{12} + c_2x_{22} + c_3x_{32} + c_4x_{42} + c_5x_{52} + c_6x_{62} + c_7x_{72} + c_8x_{82}$$

ou

$$z = 112x_{11} + 98x_{21} + 132x_{31} + 135x_{41} + 22x_{51} + 175x_{61} + 90x_{71} + 145x_{81} + 112x_{12} + 98x_{22} + 132x_{32} + 135x_{42} + 22x_{52} + 175x_{62} + 90x_{72} + 135x_{82}$$

Sujeito às:

Restrições funcionais:

1ª restrição:

$$a_1x_{11} + a_2x_{21} + a_3x_{31} + a_4x_{41} + a_5x_{51} + a_6x_{61} + a_7x_{71} + a_8x_{81} \leq b_1$$

$$a_1x_{12} + a_2x_{22} + a_3x_{32} + a_4x_{42} + a_5x_{52} + a_6x_{62} + a_7x_{72} + a_8x_{82} \leq b_2$$

ou

$$2{,}5x_{11} + 4{,}5x_{21} + 9{,}5x_{31} + 3{,}2x_{41} + 5{,}8x_{51} + 4x_{61} + 6x_{71} + 1{,}3x_{81} \leq 20$$

$$2{,}5x_{12} + 4{,}5x_{22} + 9{,}5x_{32} + 3{,}2x_{42} + 5{,}8x_{52} + 4x_{62} + 6x_{72} + 1{,}3x_{82} \leq 8$$

2ª restrição (garante a inserção do item em uma única mochila):

$$x_{11} + x_{12} \leq 1$$

$$x_{21} + x_{22} \leq 1$$

$$x_{31} + x_{32} \leq 1$$

$$x_{41} + x_{42} \leq 1$$

$$x_{51} + x_{52} \leq 1$$

$$x_{61} + x_{62} \leq 1$$

$$x_{71} + x_{72} \leq 1$$

$$x_{81} + x_{82} \leq 1$$

Restrições de domínio:

$$x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32}, x_{41}, x_{42}, x_{51}, x_{52}, x_{61}, x_{62}, x_{71}, x_{72}, x_{81}, x_{82} = \{0{,}1\}$$

Em que:

n: número de itens, com $n = 8$.

a_j: peso do item j.

c_j: valor do item j.

b_i : tamanho ou capacidade das mochilas, com $b_1 = 20\ kg$ e $b_2 = 8\ kg$

x_j : variável de decisão binária, sendo igual a 1, se o item for selecionado e igual a 0, se o item não for selecionado.

O modelo na forma de somatório é dado por:

$$Maximizar\ z = \sum_{i=1}^{2} \sum_{j=1}^{8} c_j x_{ji}$$

Sujeito às:

Restrições funcionais:

$\sum_{i=1}^{2} a_j x_{ji} \leq b_i$, para todo $j = 1{,}2{,}3{,}4{,}5{,}6{,}7{,}8$

$\sum_{j=1}^{8} x_{ji} \leq 1$, para todo $i = 1,2$

Restrições de domínio:

$x_{ji} = \{0,1\}$, para todo $j = 1, \ldots, n$ e $i = 1, \ldots, m$

O código está listado na Figura *28*.

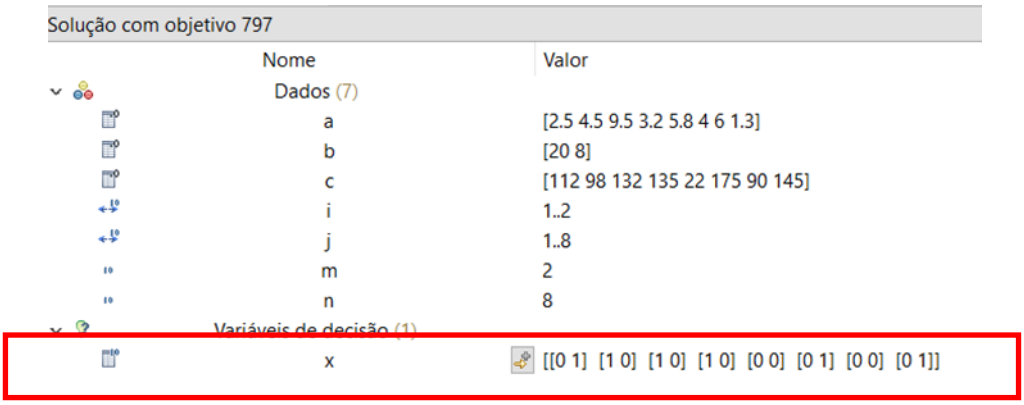
Figura 28 - CPLEX – código do problema da mochila 2

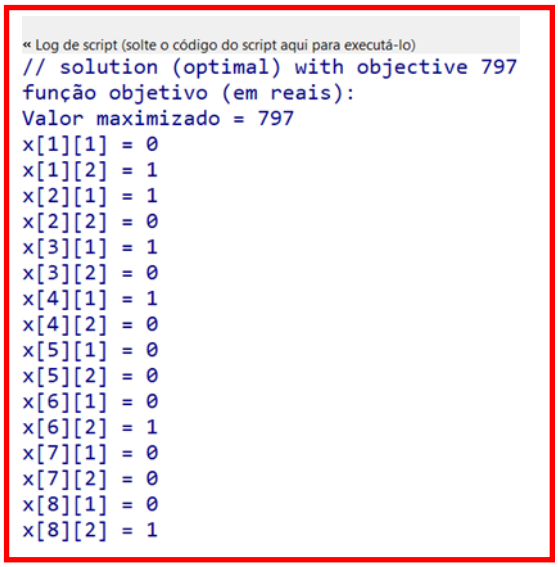
```
1  //parâmetros
2  int n = 8;            //número de itens
3  range j = 1..n;
4  int m = 2;            //número de mochilas
5  range i = 1..m;
6  float b[i]=[20,8];    //capacidade das mochilas (kg)
7  float a[j]=[2.5,4.5,9.5,3.2,5.8,4.0,6.0,1.3]; //peso  de cada item
8  float c[j]=[112,98,132,135,22,175,90,145];    //valor de cada item
9
10 //Variáveis de decisão (0 ou 1)
11 dvar boolean x[j][i];
12
13 //Função Objetivo
14 maximize
15   sum(m in i,n in j) c[n]*x[n][m];
16
17 //Restrições
18 subject to {
19   forall(m in i)
20     sum(n in j) a[n]*x[n][m] <= b[m];
21
22   forall(n in j)
23    sum(m in i)
24      x[n][m] <=1;
25
26 };
27
28 //javascript
29 execute RESULTADOS{
30   writeln ("função objetivo (em reais):");
31   writeln ("Valor maximizado = ", cplex.getObjValue());
32   for (var n in j)
33     for (var m in i)
34      writeln("x["+ n +"]"+"["+ m +"]"+ " = "+x[n][m]);
35 }
```

Fonte: a autora.

Os resultados estão apresentados na Figura 29.

Figura 29 - CPLEX – resultado do problema da mochila 2





Fonte: a autora.

3 CONCLUSÃO

Este estudo propôs o uso do CPLEX como alternativa para aproximar as disciplinas de programação à disciplina de Pesquisa Operacional, no curso de Análise e Desenvolvimento de Sistemas. O cadastro no site da IBM para obtenção de licença e a apresentação inicial do software foram apresentados no Apêndice A.

Inicialmente, foi feito um histórico do desenvolvimento da pesquisa operacional, a partir da 2ª Guerra Mundial. Foram apresentados os elementos de um problema de PO, em particular, da programação linear e sete modelos típicos amplamente discutidos na literatura, seguidos da implementação de dez exemplos com o auxílio do CPLEX.

O Apêndice B apresentou os exemplos realizados com o solver, do Excel. Procurou-se realizar uma validação para os resultados obtidos.

Os algoritmos desenvolvidos nos exemplos permitiram a análise dos principais recursos do CPLEX. Esta análise afirma que ele permite o uso dos conceitos de programação (definição de variáveis, criação de funções, uso de laços, formatação de resultados) e, também, contribui para uma discussão prévia sobre a implementação do modelo, especialmente, quando a função objetivo é definida na forma de somatório. Entende-se que esta abordagem permite momentos de discussão sobre a modelagem do problema.

Como proposta de estudos futuros, propõe-se a realização de modelos mais complexos e a implementação da análise de sensibilidade, não tratada nesta pesquisa.

REFERÊNCIAS

ANDRADE, E.L. **Introdução à Pesquisa Operacional**. Rio de Janeiro: LTC, 2009.

BELFIORE, P.; FÁVERO, L. P. **Pesquisa operacional**: para cursos de engenharia. Rio de Janeiro: Elsevier, 2013.

FERREIRA, W.P. **Como usar o CPLEX**? 13 abr. 2017. Disponível em: https://www.youtube.com/watch?v=3Yap4UVbt2s. Acesso em: 04 jul. 2021.

HILLIER, F.S.; LIEBERMAN, G.J. **Introdução à Pesquisa Operacional**. Porto Alegre: AMGH, 2013.

IBM. **IBM CPLEX Optimizer**. Disponível em: https://www.ibm.com/br-pt/analytics/cplex-optimizer. Acesso em 29 jul. 2022.

LACHTERMACHER. **Pesquisa Operacional na tomada de decisões**. São Paulo: Pearson Prentice Hall, 2009.

PIZZOLATO, N. D; GANDOLPHO, A.A. **Técnicas de Otimização**. Rio de Janeiro: LTC, 2009.

PRADO, D.S. **Programação Linear**. Nova Lima: Falconi Editora, 2016. Série Pesquisa Operacional, v.1.

SILVA, Ermes Medeiros *et al*. **Pesquisa Operacional**: programação linear. São Paulo: Atlas, 1998.

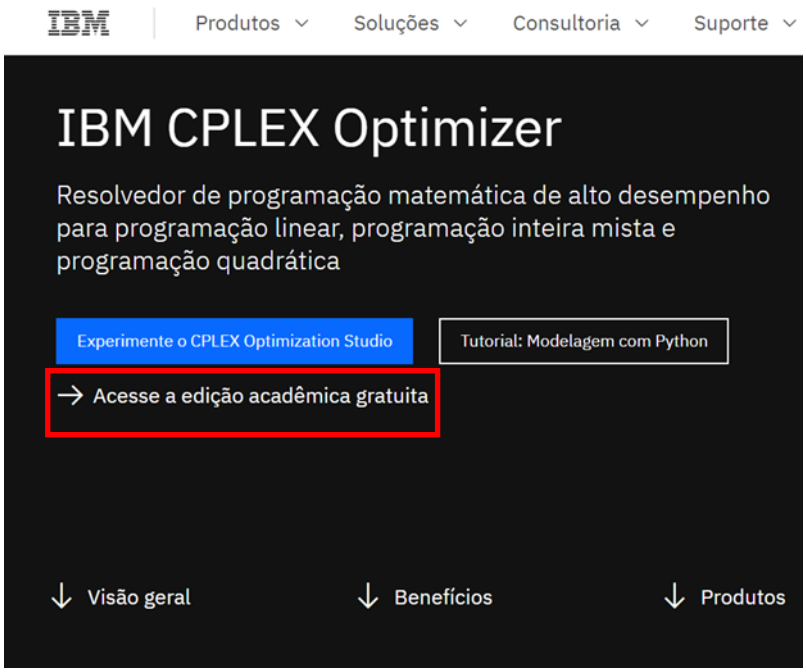
APÊNDICE

APÊNDICE A - Software CPLEX

A.1. Licença

A licença é obtida pelo acesso ao endereço https://www.ibm.com/br-pt/analytics/cplex-optimizer e seleção da edição acadêmica gratuita (Figura 30).

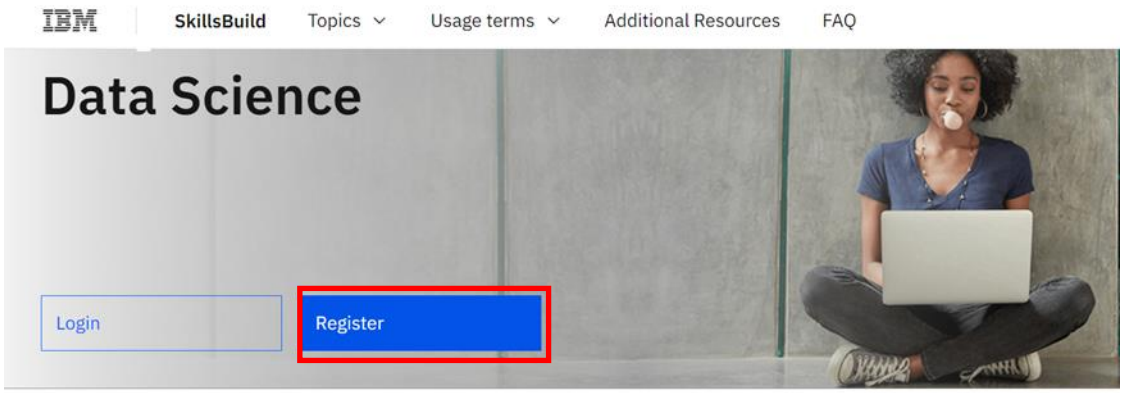
Figura 30 - Página IBM: tela inicial



Fonte: https://www.ibm.com/br-pt/analytics/cplex-optimizer

O registro é feio por meio de e-mail institucional (Figura 31).

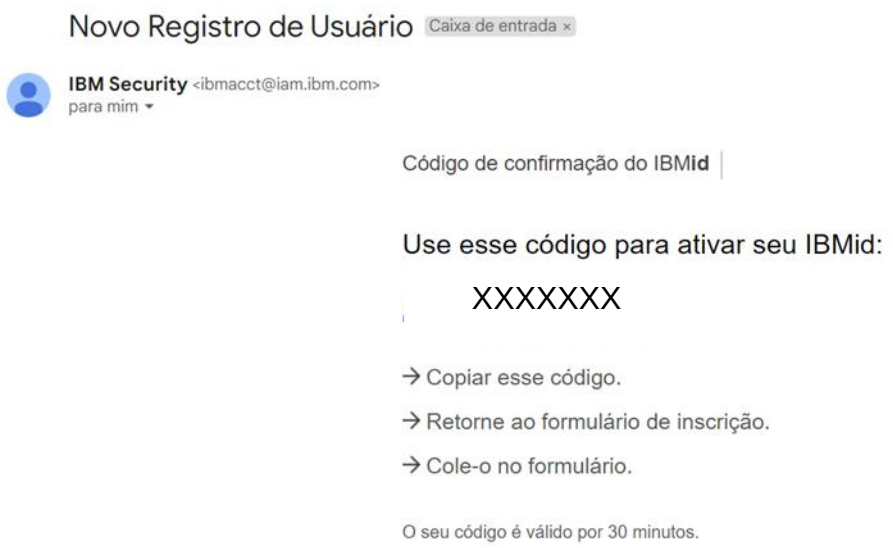
Figura 31 - Página IBM - registro



Fonte: https://www.ibm.com/academic/topic/data-science

Após o registro, será enviado, ao e-mail informado no cadastro, um código para ativação do IBMid (Figura 32).

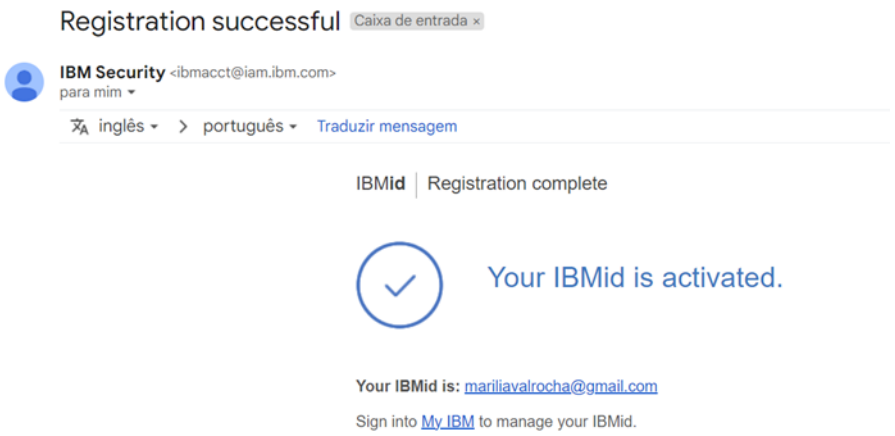
Figura 32 – E-mail IBM – código de ativação



Fonte: a autora.

Com o código informado e a ativação do id realizada, é enviado ao usuário a confirmação do registro (Figura 33).

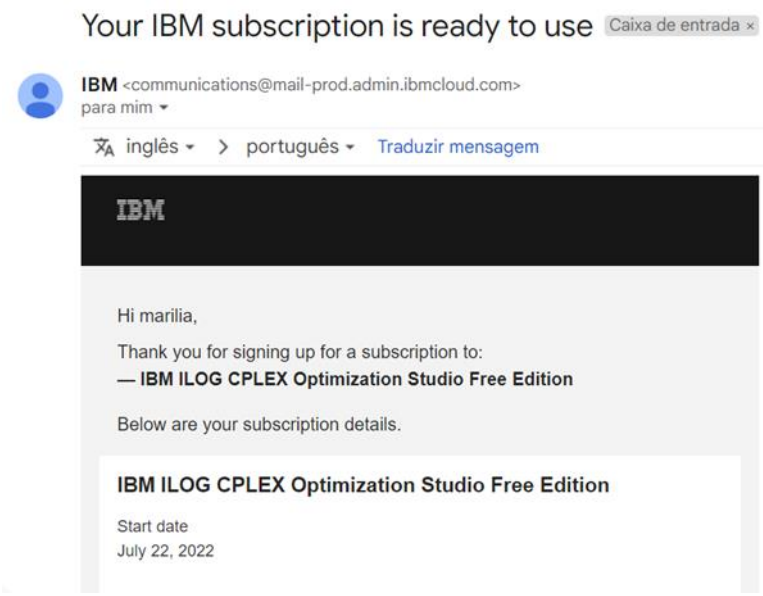
Figura 33 – E-mail IBM – confirmação de registro



Fonte: a autora.

Finalmente, é feita a confirmação de que a assinatura está pronta para uso, via e-mail (Figura 34).

Figura 34 – E-mail IBM – liberação de uso



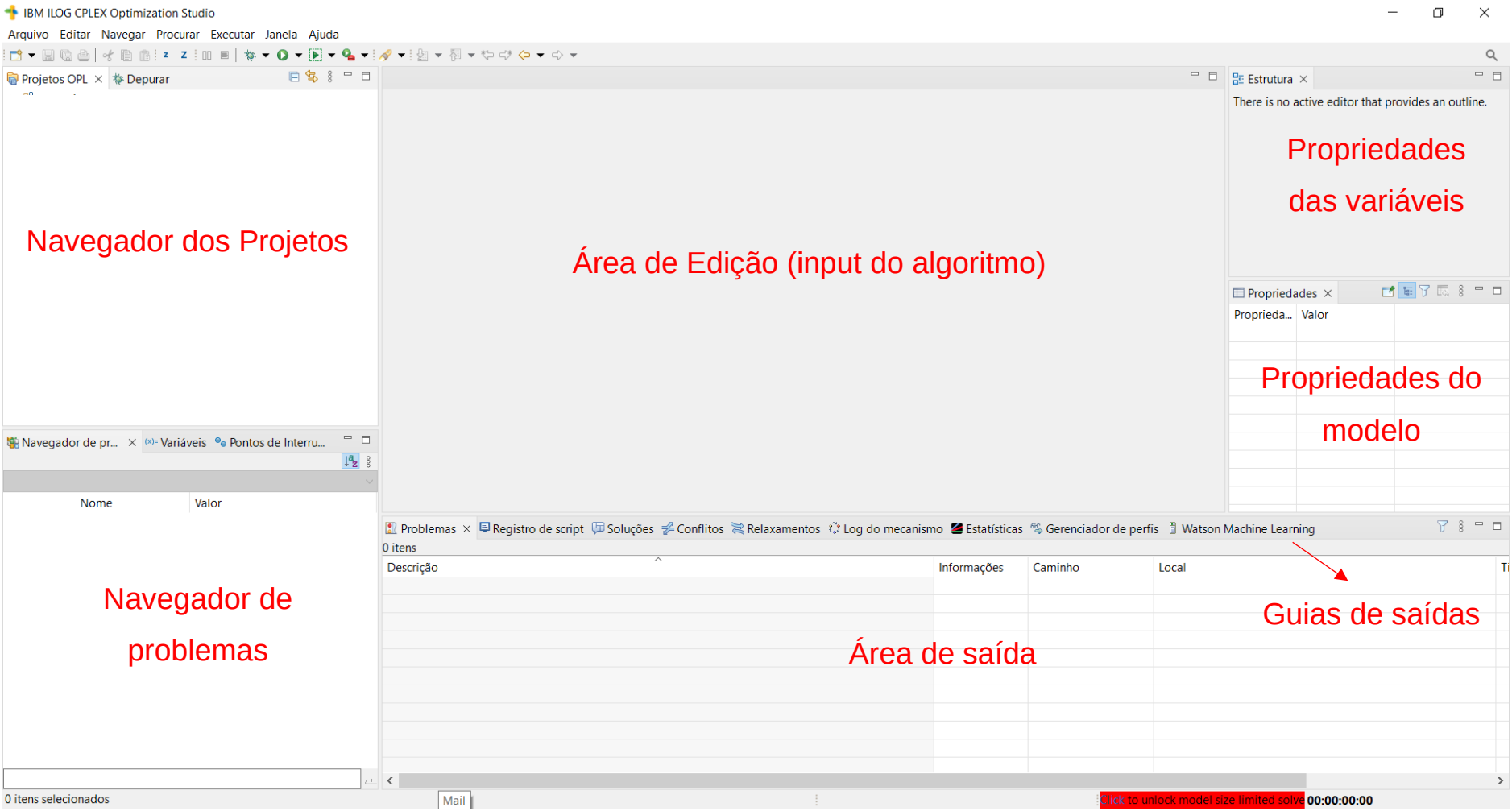
Fonte: a autora.

A.2. Interface

A Figura *35* apresenta a área de edição (para input do algoritmo), o navegador de projetos (lista os modelos cadastrados em árvore) e o navegador de problemas (lista os resultados obtidos ou eventuais problemas). As Guias de Saída são:

- Problemas: apresenta eventuais erros no processamento do modelo.
- Registro de Script: apresenta o resultado obtido no pós-processamento.
- Soluções: apresenta a solução final do modelo (variáveis e função objetivo).
- Conflitos: aponta locais prováveis para viabilizar o modelo.
- Relaxamentos: aponta locais que podem ser relaxados para viabilizar o modelo.
- Log do mecanismo: apresenta informações do mecanismo de resolução feita pelo CPLEX.
- Estatísticas: exige total de restrições, de variáveis e de coeficientes e o número de iterações necessárias para obter a solução final.
- Gerenciador de perfis: apresenta o tempo e a memória necessária em cada etapa de resolução.
- *Watson Machine Learning*: para uso em nuvem.

Figura 35 – CPLEX – interface

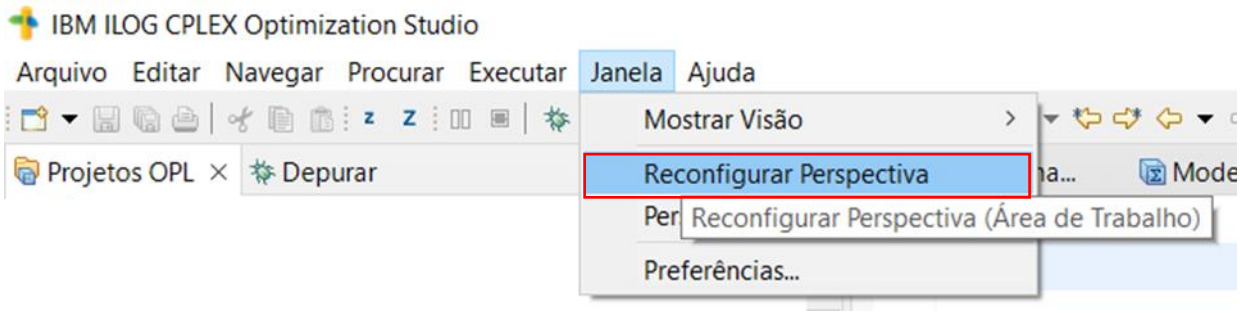


Fonte: a autora.

Em Estrutura (Figura *35*), estão listadas as características das variáveis do modelo. Em Propriedades, estão listados os dados do modelo. A Figura 46, exemplifica as respectivas saídas.

Em janela/reconfigurar perspectiva, há a possibilidade de restabelecer o layout padrão (Figura 36).

Figura 36 – CPLEX – reconfigurar perspectiva



Fonte: a autora.

A.3. Comandos

As principais informações para escrever os códigos estão resumidas na sequência e são ilustradas nos exemplos apresentados no item 3. Outras informações são obtidas nos manuais citados no item A.5.

- Tipos de variáveis: int (inteiros), boolean (booleanos), float (decimais), string (textos).
  Inteiros positivos: int+
  Float positivos: float+
- Declaração de variável: dvar
- Declaração da função objetivo: maximize ou minimize.
- Declaração de restrições: subject to {}
- Inserção de comentários: \\ ou /*...*/

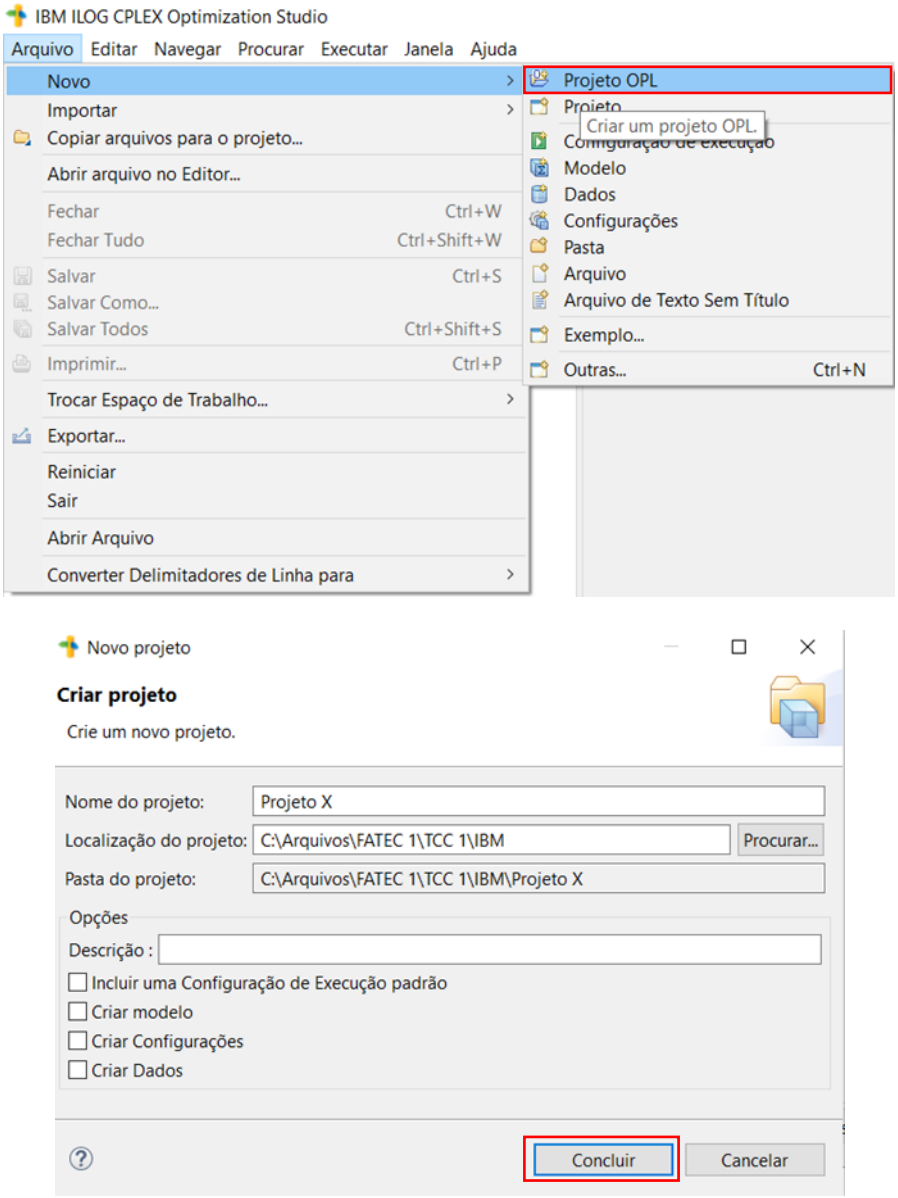
Foi incorporada uma implementação ao CPLEX no JavaScriptTM que permite o uso nos pré e pós-processamentos no arquivo de modelo (.mod), no processamento do arquivo de dados (.dat), e na definição de parâmetros. De acordo com Ferreira (2017), o pré-processamento permite a preparação dos dados (para solução do modelo), a pré-resolução (para redução do tamanho do problema) e agregação (para eliminar variáveis e linhas por substituição). O pós-processamento é usado para manipulação das soluções, controle das saídas e exibição de dados.

Neste estudo, foram exploradas algumas possibilidades no pós-processamento para exibição de dados.

A.4. Inclusão do modelo, execução e resultados

Foram listados o procedimento para inclusão de um modelo. Inicialmente, cadastra-se um projeto, em Arquivo/Novo/Projeto OPL (Figura 37).

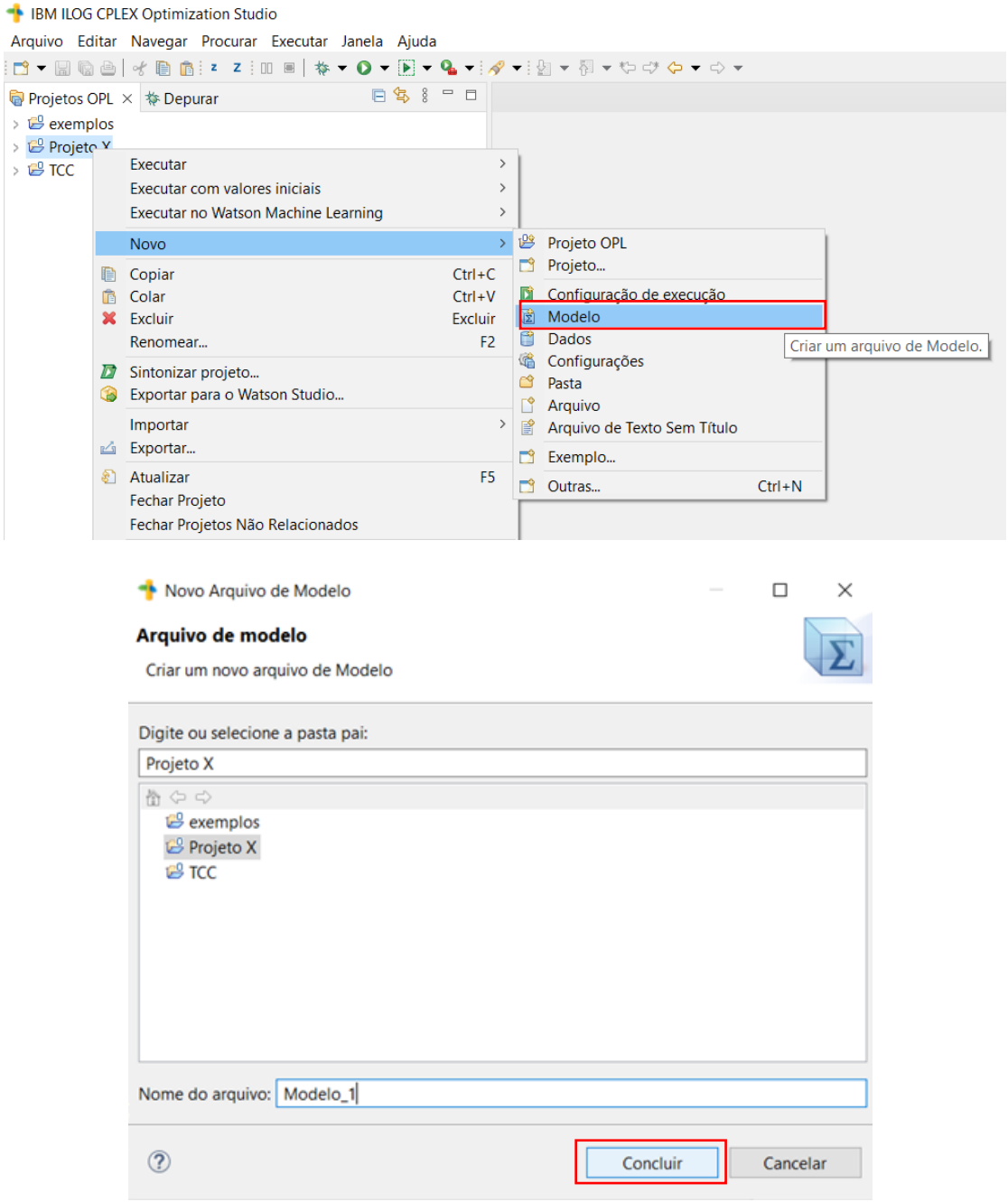
Figura 37 – CPLEX – cadastro projeto



Fonte: a autora.

Com o botão direito do mouse sobre o nome do projeto criado, seleciona-se Novo/Modelo (Figura 38).

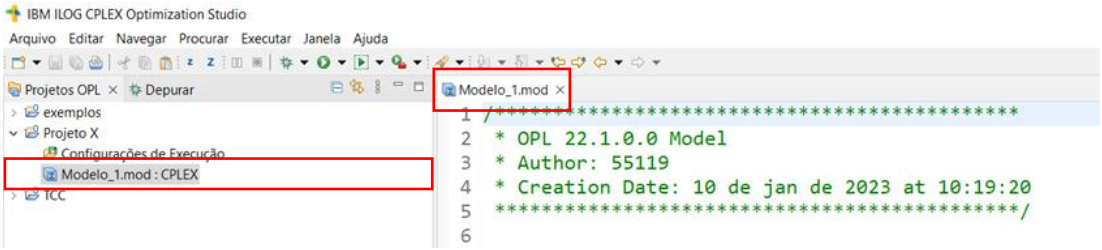
Figura 38 – CPLEX – cadastro modelo



Fonte: a autora.

Após a conclusão do cadastro, o arquivo Modelo_1.mod é criado (Figura 39).

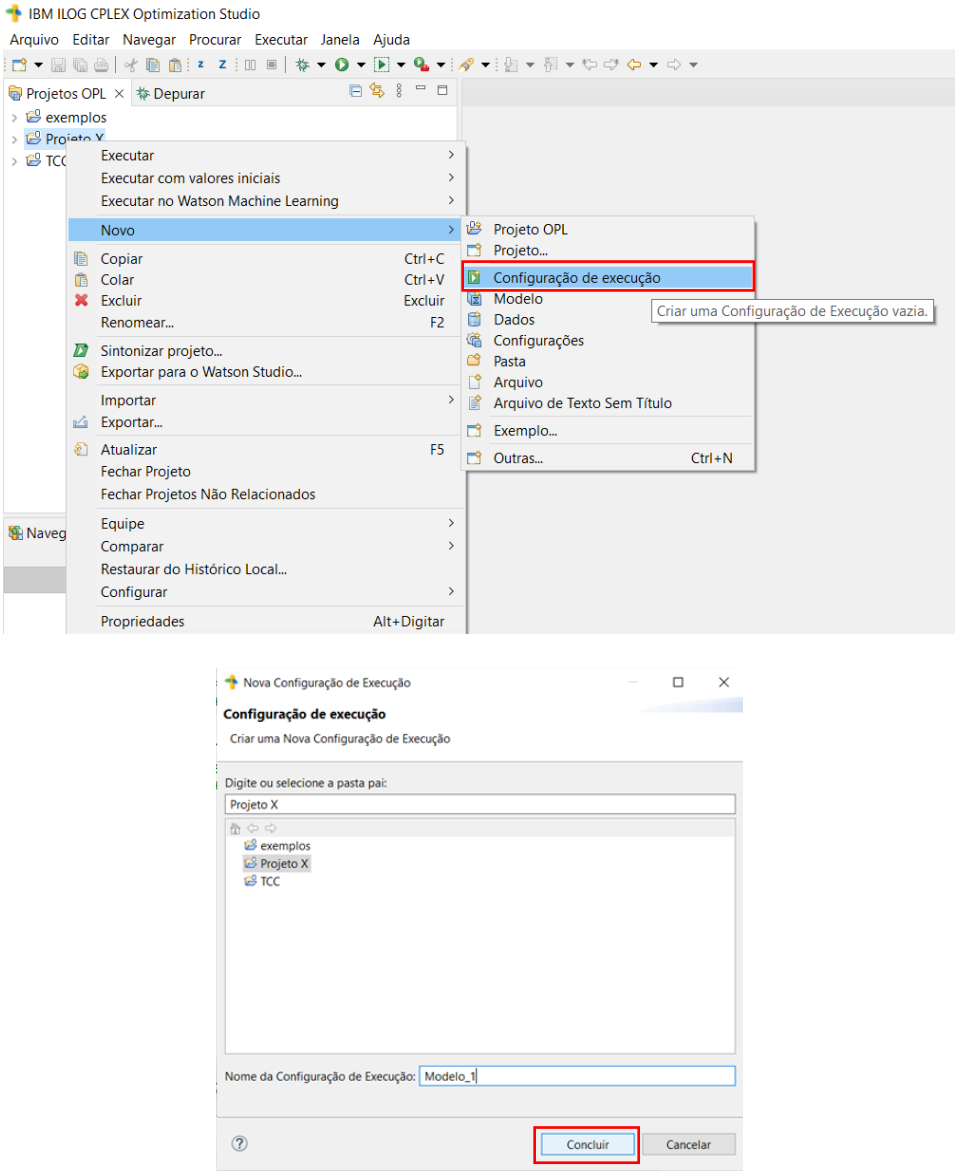
Figura 39 – CPLEX – resultado do cadastro modelo



Fonte: a autora.

Com o botão direito do mouse sobre o nome do projeto, selecionar Novo/Configurações de execução (Figura 40).

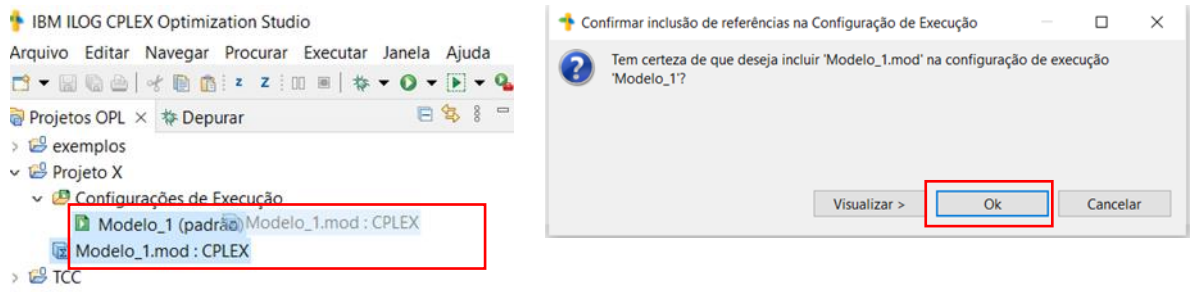
Figura 40 – CPLEX – cadastro configuração de execução



Fonte: a autora.

Para finalizar, o modelo deve ser arrastado para a respectiva configuração de execução (Figura 41).

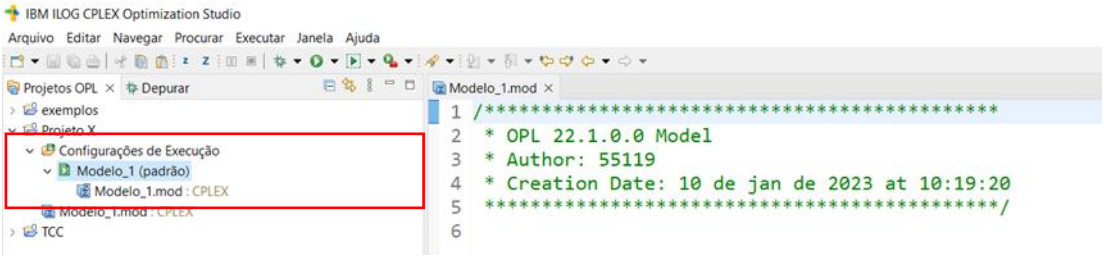
Figura 41 – CPLEX – cadastro configuração de execução com modelo



Fonte: a autora.

Como resultado, no navegador de projetos, visualiza-se o Modelo_1.mod no interior da respectiva configuração de execução (Figura 42).

Figura 42 – CPLEX – resultado do cadastro do modelo

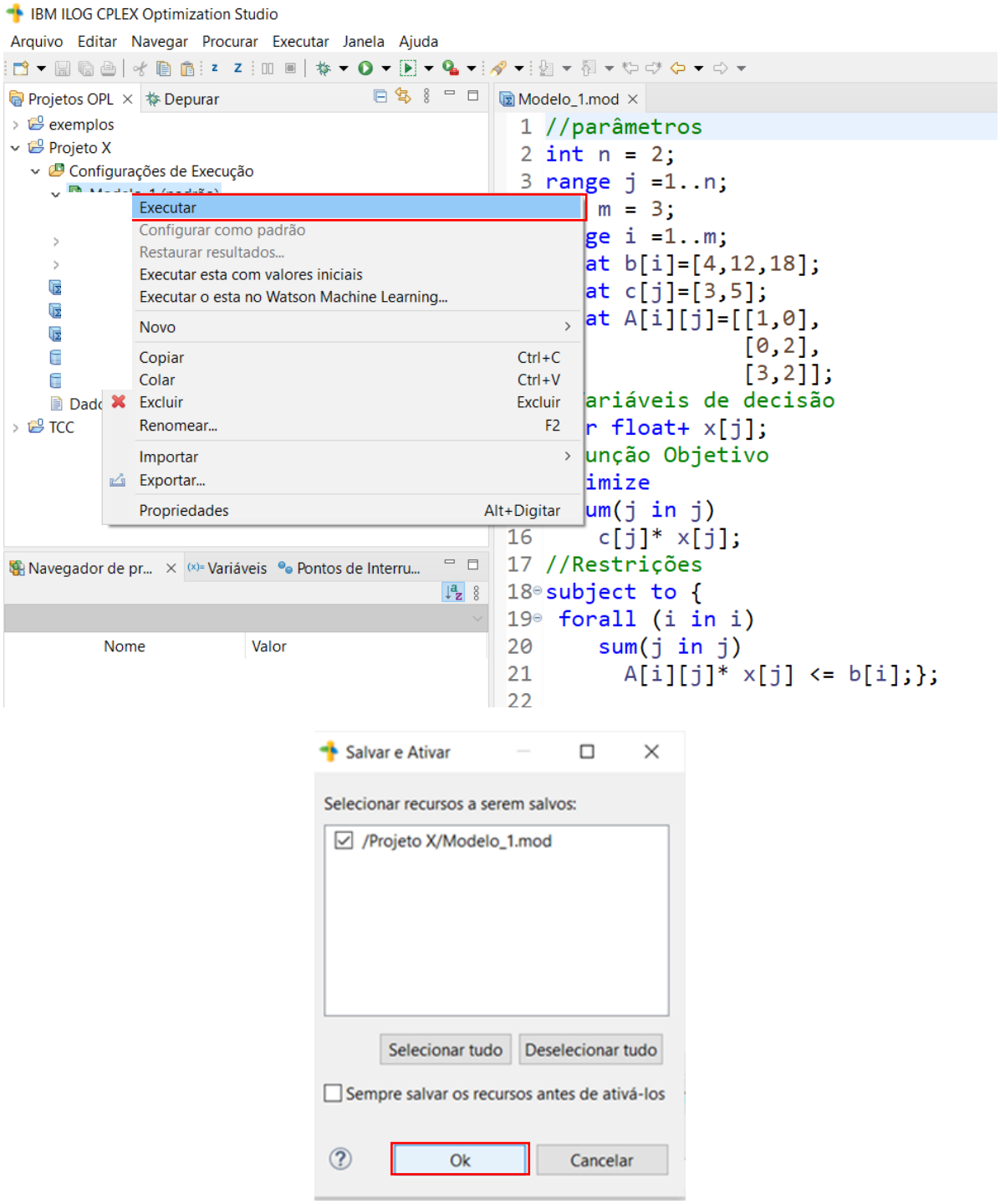


Fonte: a autora.

Um projeto pode conter mais de um modelo e, para cada modelo, é necessário criar uma configuração de execução. O modelo e a configuração de execução podem ter nomes diferentes.

O algoritmo é inserido no Modelo_1.mod. Neste caso, usou-se o exemplo do modelo de alocação de recursos, apresentado no item 3.1. Com o botão direito do mouse sobre as configurações de execução do modelo, seleciona-se ‘Executar’ (Figura 43).

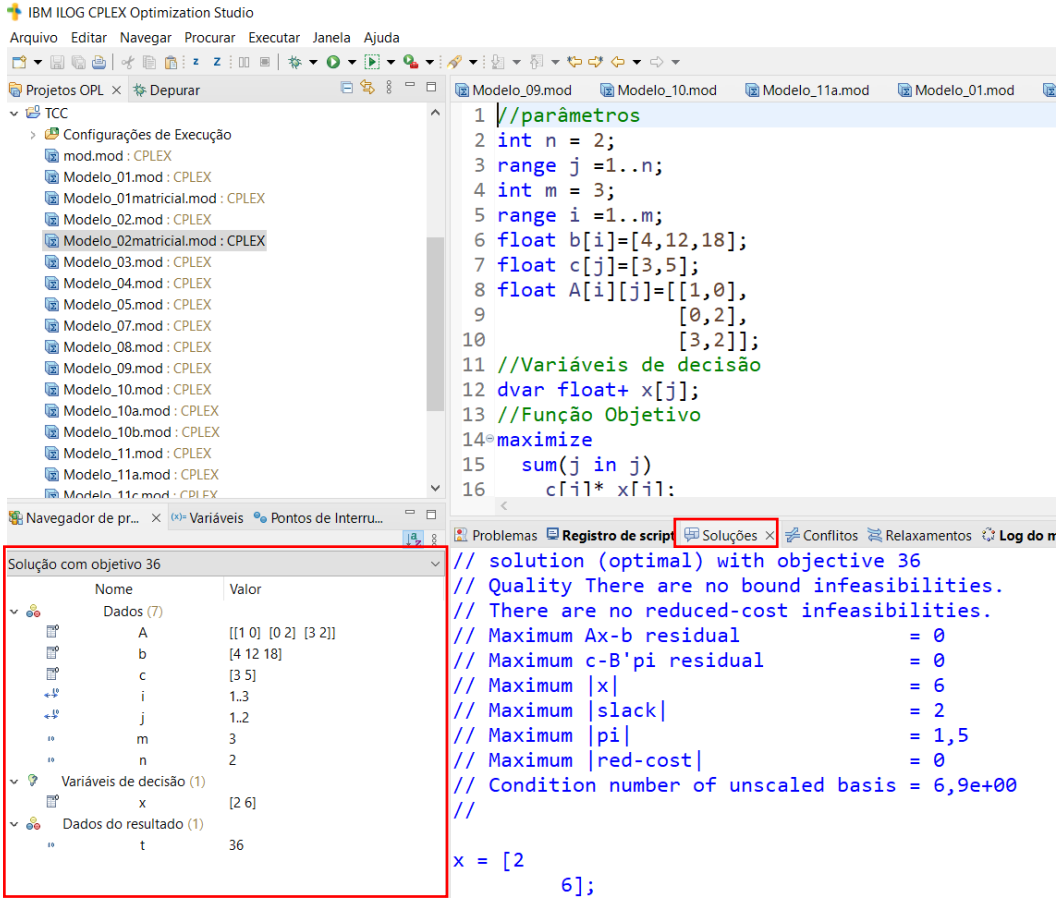
Figura 43 – CPLEX – execução do modelo



Fonte: a autora.

Os resultados são apresentados na área Navegador de problemas e, também, na guia Soluções (Figura 44).

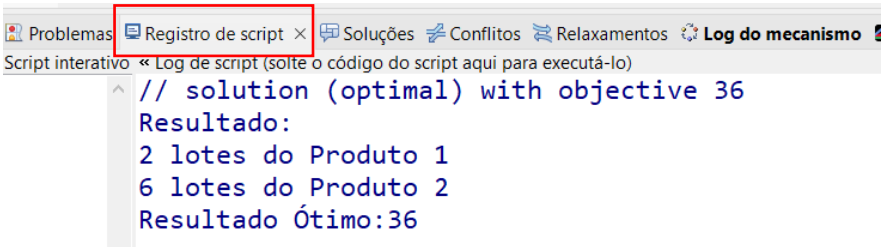
Figura 44 – CPLEX – resultado da execução do modelo



Fonte: a autora.

O resultado da execução do pós-processamento é apresentado na pasta ‘Registro de script’.

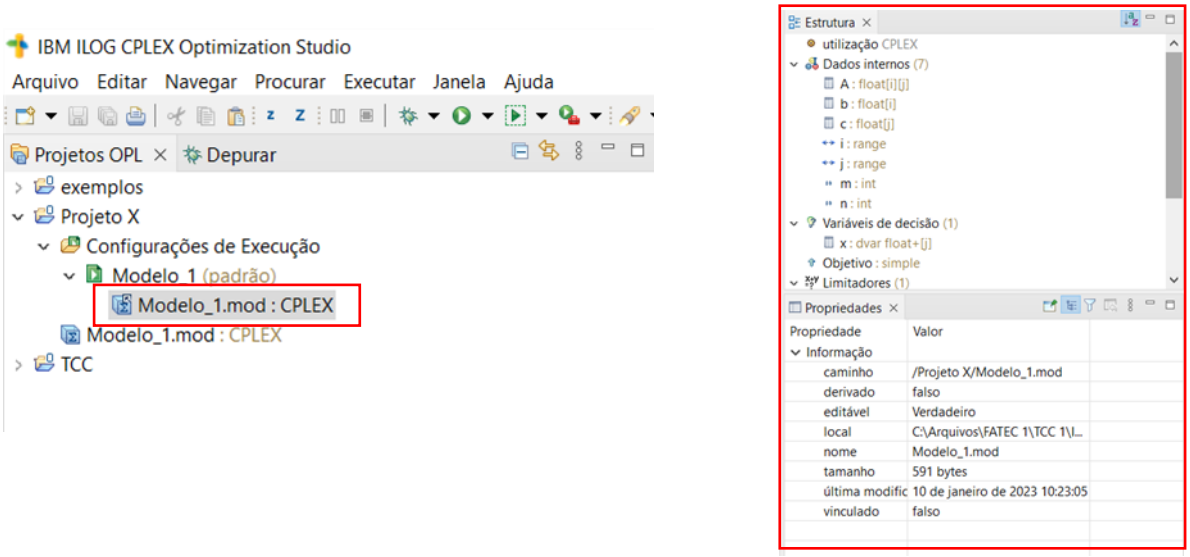
Figura 45 – CPLEX – registro de script



Fonte: a autora.

Um duplo clique sobre o modelo lista as suas propriedades, entre elas, a data de modificação, local de salvamento e tamanho. Em Estrutura, temos os tipos das variáveis definidas (Figura 46).

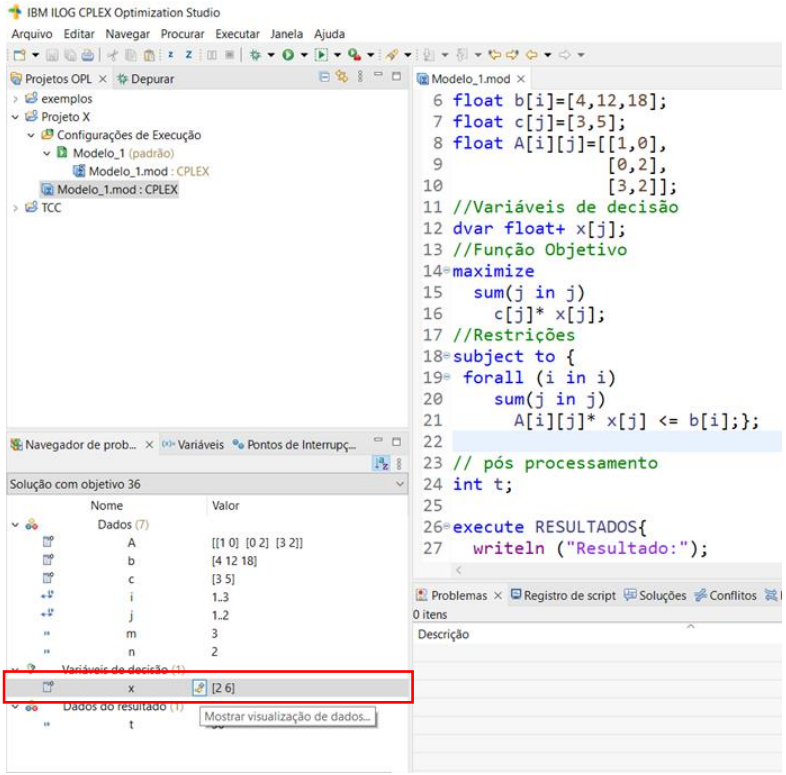
Figura 46 – CPLEX – propriedades do modelo



Fonte: a autora.

Após o processamento, é possível selecionar o ícone ‘mostrar visualização de dados’ (Figura 47).

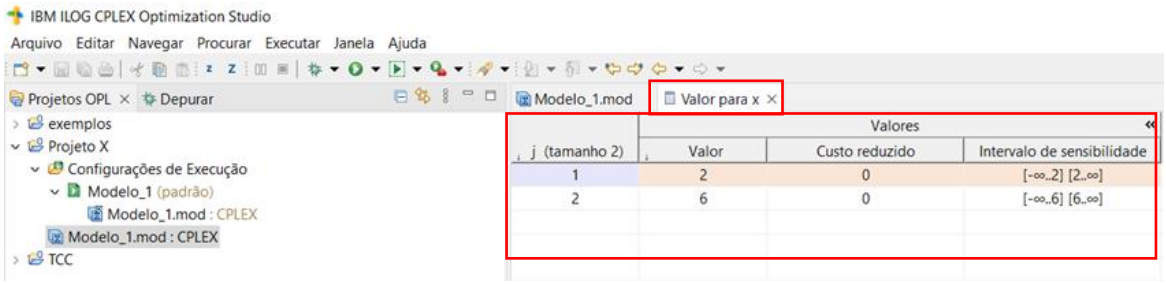
Figura 47 – CPLEX – mostrar visualização dos dados



Fonte: a autora.

Na sequência, a pasta ‘valor para x’, é apresentada. Esta janela permite a realização da análise de sensibilidade, não discutida neste estudo (Figura 48).

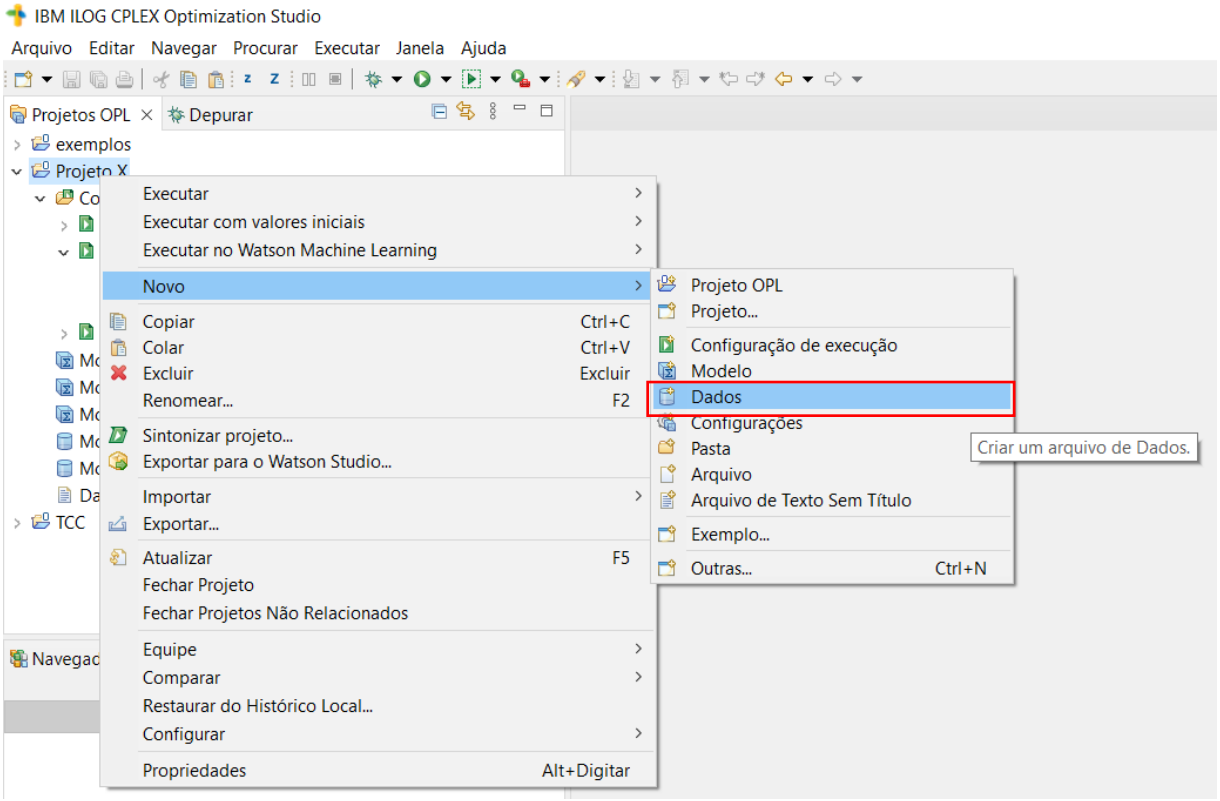
Figura 48 – CPLEX – análise de sensibilidade

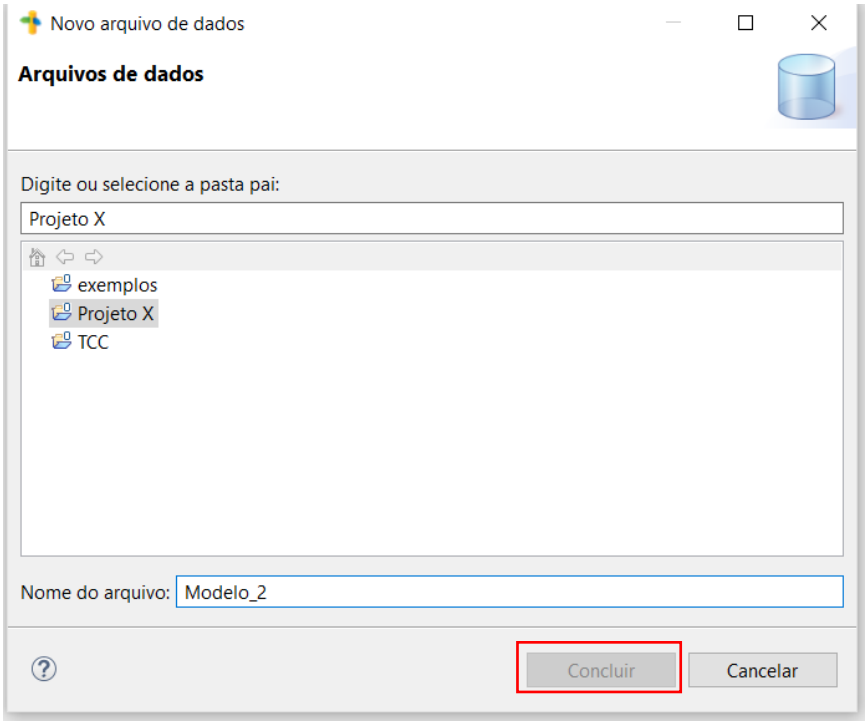


Fonte: a autora.

É possível separar os dados de entrada (parâmetros do modelo) em um banco de dados separadamente do algoritmo. Para ilustrar, foi usado o mesmo exemplo. Foi criado o Modelo_2 e a respectiva configuração de execução. Para apresentação deste recurso, criou-se um arquivo.dat, com o botão direito do mouse sobre o nome do projeto, em Novo/Dados (Figura 49).

Figura 49 – CPLEX – cadastro de dados

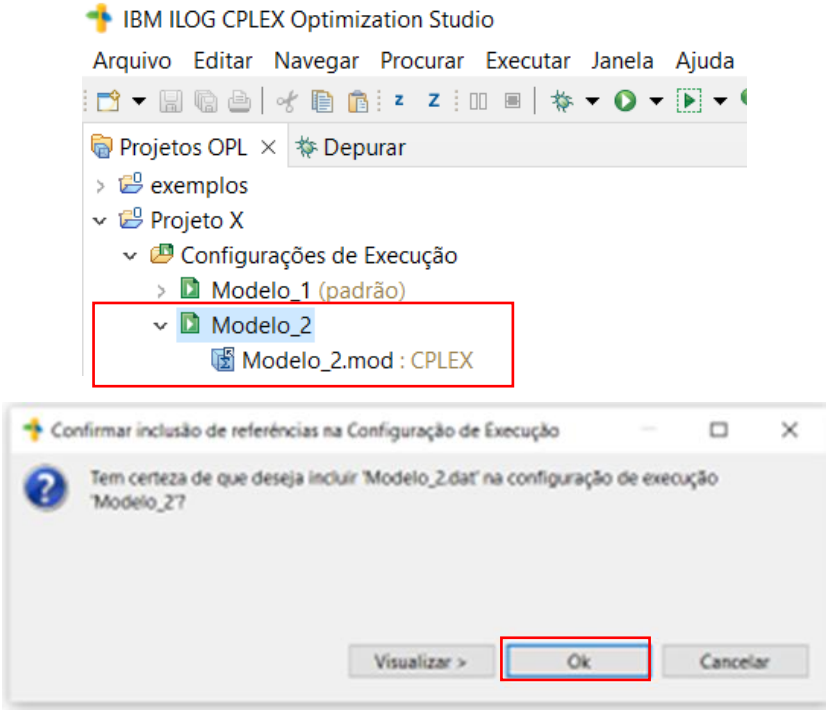




Fonte: a autora.

Este arquivo foi arrastado para a configuração de execução do Modelo_2 (Figura 50).

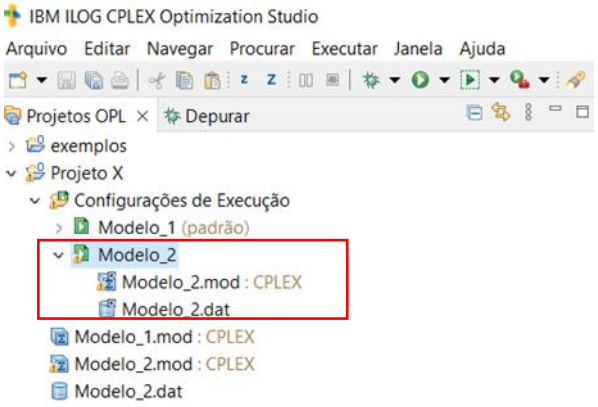
Figura 50 – CPLEX – cadastro de dados



Fonte: a autora.

Observa-se na árvore, em Projetos OPL, que no interior da configuração de Execução do Modelo_2 estão os arquivos Modelo_2.mod e Modelo_2.dat (Figura 51).

Figura 51 – CPLEX – cadastro de dados



Fonte: a autora.

O arquivo Modelo_2.dat contém os parâmetros do modelo. O arquivo Modelo_2.mod contém o algoritmo apresentado no exemplo anterior, com os inputs substituídos por ..., indicando que os dados serão buscados no arquivo Modelo_2.dat (Figura 52).

Figura 52 – CPLEX – códigos arquivo .mod e arquivo.dat

Modelo_02.mod

```
1  //parâmetros
2  int n = ...;
3  range j =1..n;
4  int m = ...;
5  range i =1..m;
6
7  float b[i]=...;
8  float c[j]=...;
9  float A[i][j]=...;
10
11 //Variáveis de decisão
12 dvar float+ x[j];
13 //Função Objetivo
14 maximize
15   sum(j in j)
16     c[j]* x[j];
17 //Restrições
18 subject to {
19  forall (i in i)
20     sum(j in j)
21       A[i][j]* x[j] <= b[i];
22 };
```

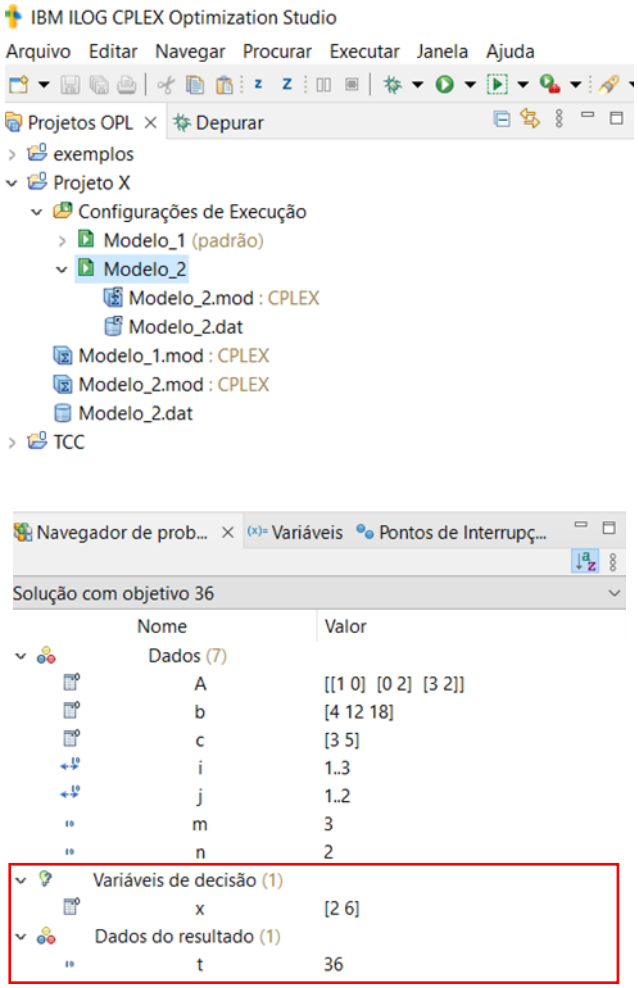
Modelo_02.dat

```
1 //parâmetros
2 n = 2;
3 m = 3;
4 b=[4,12,18];
5 c=[3,5];
6 A=[[1,0],
7    [0,2],
8    [3,2]];
```

Fonte: a autora.

Após a execução, o CPLEX retorna o mesmo resultado do exemplo anterior (Figura 53).

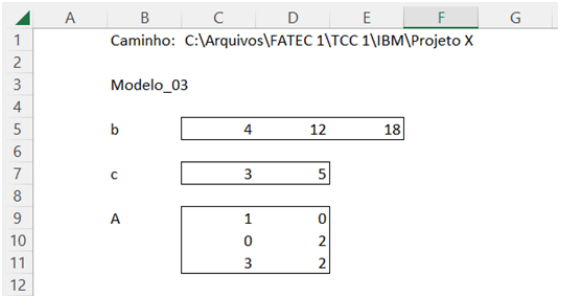
Figura 53 – CPLEX – resultado



Fonte: a autora.

O CPLEX permite que os dados sejam lidos de uma planilha Excel (Microsoft). Para exemplificar, foram criados o Modelo_03.mod, o Modelo_03.dat e a respectiva configuração de execução. Será repetido o exemplo anterior, com os dados inseridos na planilha Dados_Modelo_3.xlsx (Figura 54).

Figura 54 – Excel – input dos parâmetros



Fonte: a autora.

O arquivo Modelo_3.mod não necessita de alteração, em relação ao exemplo anterior. O Modelo_3.dat realiza a conexão com a planilha e a leitura dos intervalos de cada variável (Figura 55).

Figura 55 – CPLEX – Modelo_03.mod e Modelo_3.dat

Modelo_3.mod

```
//parâmetros
int n = ...;
range j =1..n;
int m = ...;
range i =1..m;

float b[i]=...;
float c[j]=...;
float A[i][j]=...;

//Variáveis de decisão
dvar float+ x[j];
//Função Objetivo
maximize
  sum(j in j)
    c[j]* x[j];
//Restrições
subject to {
 forall (i in i)
    sum(j in j)
      A[i][j]* x[j] <= b[i];
};
```

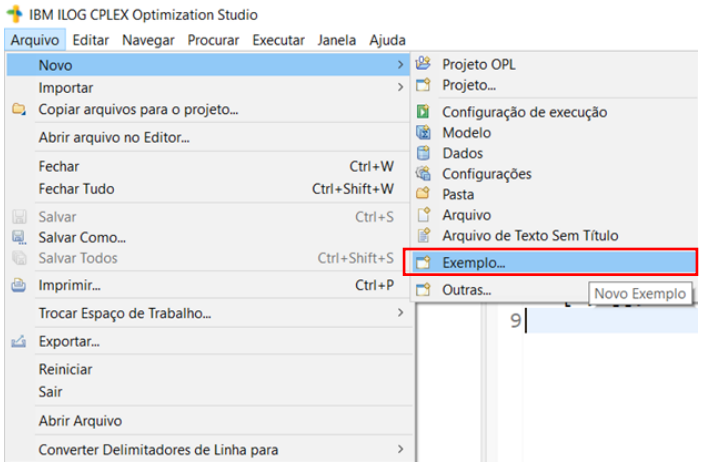
O Modelo_3.dat

```
//parâmetros
n = 2;
m = 3;

//estabelecer conexão
SheetsConnection sheet("Dados_Modelo_3.xlsx")

//colher dados dos intervalos
b from SheetsRead(sheet, "'Planilha1'C5:E5");
c from SheetsRead(sheet, "'Planilha1'C7:D7");
A from SheetsRead(sheet, "'Planilha1'C9:D11");

//obs: salvar o arquivo excel
// no mesmo diretório dos modelos,
// ou indicar o caminho completo
```

Fonte: a autora.

O CPLEX apresenta exemplos que podem auxiliar na criação de modelos, em Arquivo/Novo/Exemplo (Figura 56).

Figura 56 – CPLEX – exemplos nativos



Fonte: a autora.

A.5. Manuais IBM

Para saber mais sobre os recursos do CPLEX, os manuais da IBM sobre o produto estão disponíveis nos seguintes links:

https://www.ibm.com/docs/en/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/opl_languser.pdf

https://www.ibm.com/docs/en/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/opl_langref.pdf

https://www.ibm.com/docs/en/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/gsoplide.pdf

APÊNDICE B – Resolução dos exemplos com o Solver (Excel)

Considerando a letra azul como input, as células com fundo azul preenchidas pelo solver estão apresentadas na planilha Excel e a solicitação do solver, para os 10 exemplos.

Exemplo 1 (Figura 57):

Figura 57 – Excel – exemplo 1

ALOCAÇÃO DE RECURSOS

Fábricas	Uso do Recurso na atividade		Qte disponível da fábrica (horas/semana)			Qte disponível da fábrica (horas/semana)
	Produto 1	Produto 2				
1	1	0	2	I	<=	4
2	0	2	12	II	<=	12
3	3	2	18	III	<=	18
Lucro por lote (Milhares de US$)	3	5				

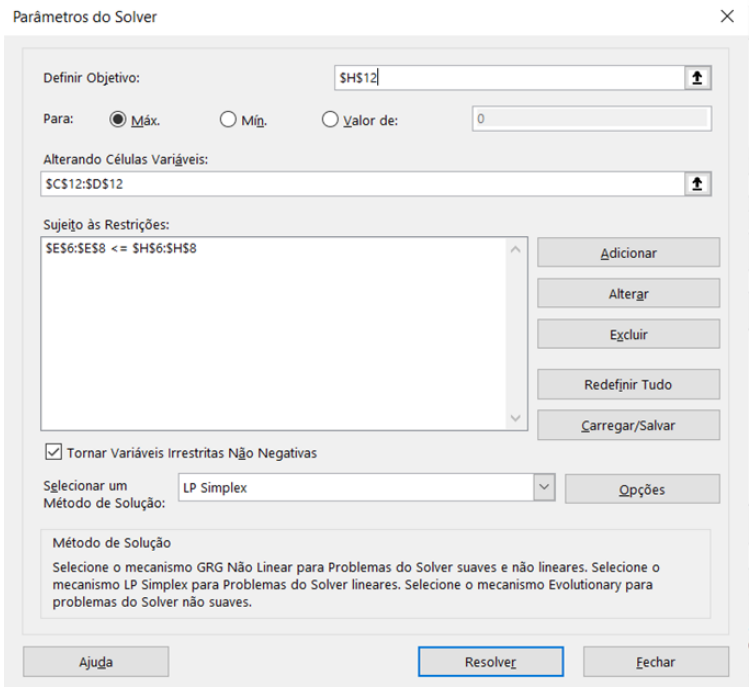
número de lotes por semana	2,0	6,0

Lucro Total (Milhares de US$)	
36,00	IV

Maximizar Z

Fórmulas:

I	=C6*C12+D6*D12
II	=C7*C12+D7*D12
III	=C8*C12+D8*D12
IV	=C9*C12+D9*D12



Fonte: a autora.

Exemplo 2 (Figura 58):

Figura 58 – Excel – exemplo 2

MISTURA

Ingredientes	Exigências diárias do ingrediente (em gramas)		Exigência diária (em gramas)			Exigência diária (em gramas)
	Bife	Batata				
Carboidrato	5	15	50	I	>=	50
Proteína	20	5	40	II	>=	40
Gordura	15	2	25	III	<=	60
Custo por refeição (US$)	4	2				

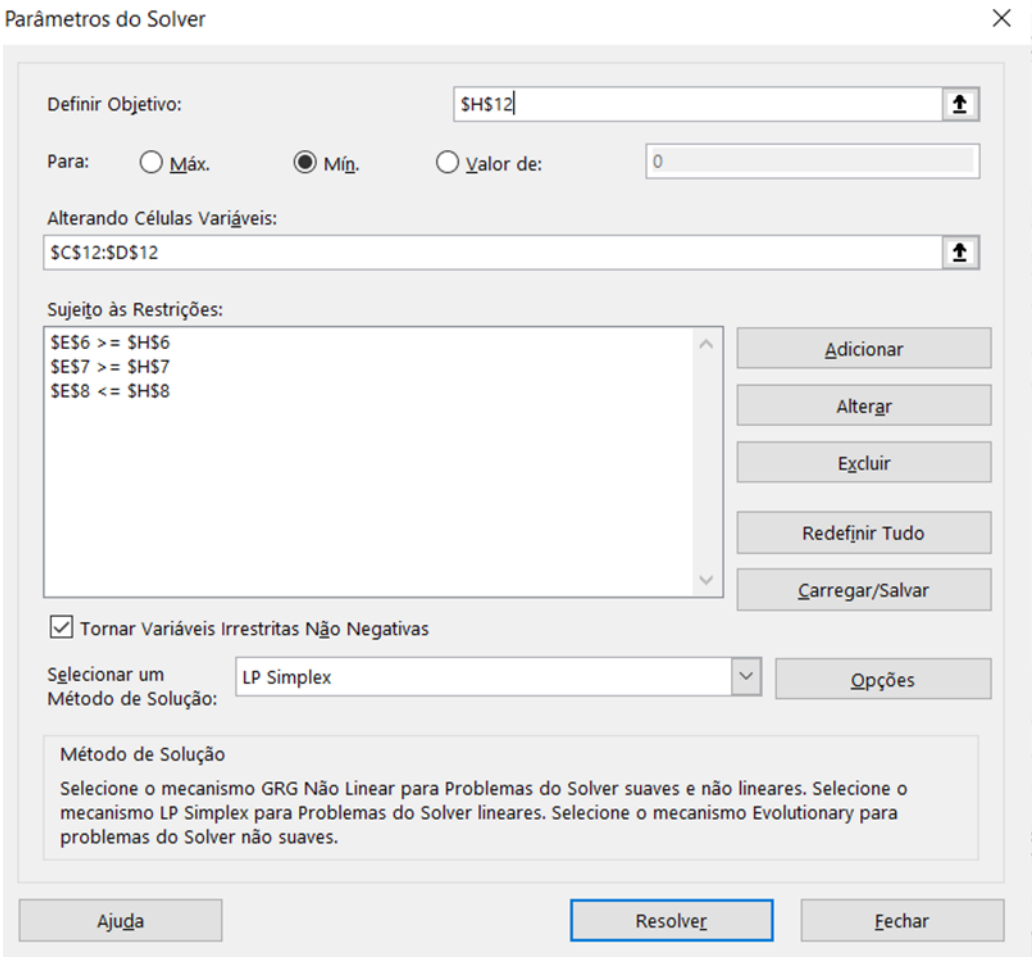
número de refeições diárias	1,2727	2,9091

Custo Total (US$)	
10,909	IV

Minimizar Z

Fórmulas:

I	=C6*C12+D6*D12
II	=C7*C12+D7*D12
III	=C8*C12+D8*D12
IV	=C9*C12+D9*D12



Fonte: a autora.

Exemplo 3 (Figura 59):

Figura 59 – Excel – exemplo 3

PROBLEMA DE TRANSPORTE 1

Custo Unitário (US$)

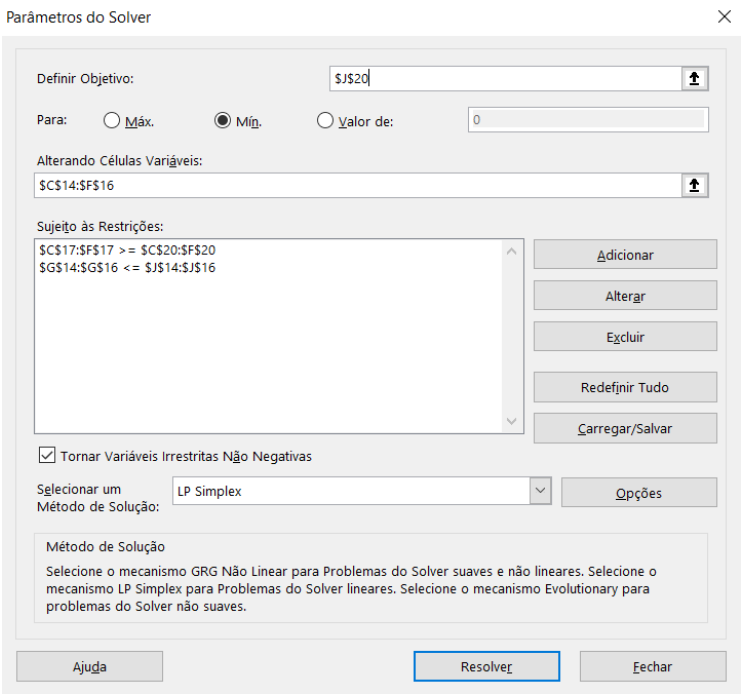
Fábrica (Origem)	Destino (depósito)			
	Depósito 1	Depósito 2	Depósito 3	Depósito 4
1	464	513	654	867
2	352	416	690	791
3	995	682	388	685

Quantidade Remetida (em carretas)

Fábrica (Origem)	Destino (depósito)				Oferta (s_i) Total Remetido			Oferta (s_i) Total Remetido	
	Depósito 1	Depósito 2	Depósito 3	Depósito 4					
1	0	20	0	55	75	I	<=	75	
2	80	45	0	0	125	II	<=	125	
3	0	0	70	30	100	III	<=	100	
Total Recebido	80	65	70	85	300			300	IX
	IV	V	VI	VII	VIII				
	>=	>=	>=	>=				Custo Total (US$)	
Demanda (d_j)	80	65	70	85	balanceado			152.535,00	X
								Minimizar Z	

Fórmulas:

I	=SOMA(C14:F14)
II	=SOMA(C15:F15)
III	=SOMA(C16:F16)
IV	=SOMA(C14:C16)
V	=SOMA(D14:D16)
VI	=SOMA(E14:E16)
VII	=SOMA(F14:F16)
VIII	=SOMA(C17:F17)
IX	=SOMA(J14:J16)
X	=SOMARPRODUTO(C7:F9;C14:F16)



Fonte: a autora.

Exemplo 4 (Figura 60):

Figura 60 – Excel – exemplo 4

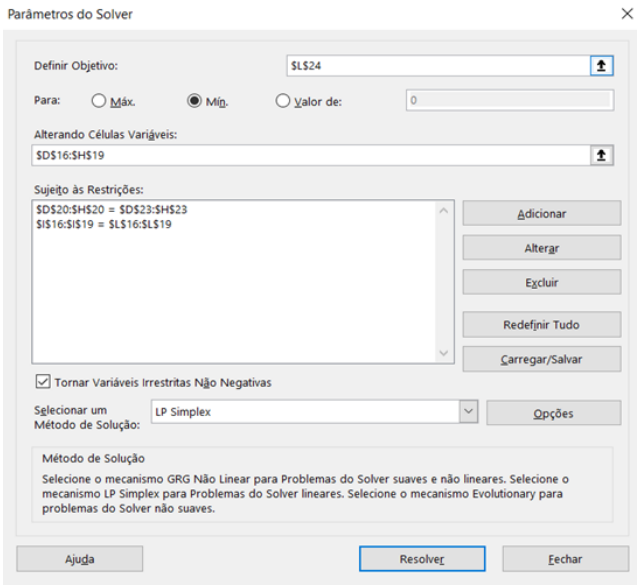
PROBLEMA DE TRANSPORTE 2 (COM TRANSBORDO)

Custos (Reais)	Fontes	Destinos				
		Transbordos		Destinos Finais		
		F.Santana	Itabuna	São Paulo	Rio	Dummy
Origens	Natal	4,0	4,5	9,0	8,0	0
	Aracaju	8,0	8,0	8,0	7,0	0
Transbordos	F.Santana	0,0	1,0	6,0	5,5	0
	Itabuna	999,0	0,0	6,0	5,0	0

Unidades Transportadas	Fontes	Destinos							
		Transbordos		Destinos Finais					
		F.Santana	Itabuna	São Paulo	Rio	Dummy	oferta Total Remetido		oferta Total Remetido
Origens	Natal	0	0	50	0	50	100	I =	100
	Aracaju	0	0	100	100	0	200	II =	200
Transbordos	F.Santana	300	0	0	0	0	300	III =	300
	Itabuna	0	300	0	0	0	300	IV =	300
	Demanda	300	300	150	100	50	900		900
		V =	VI =	VII =	VIII =	IX =	X		
	Demanda	300	300	150	100	50	balanceado		Custo Total
		XI	XII						1.950,00 XIII
									Minimizar z

Fórmulas:

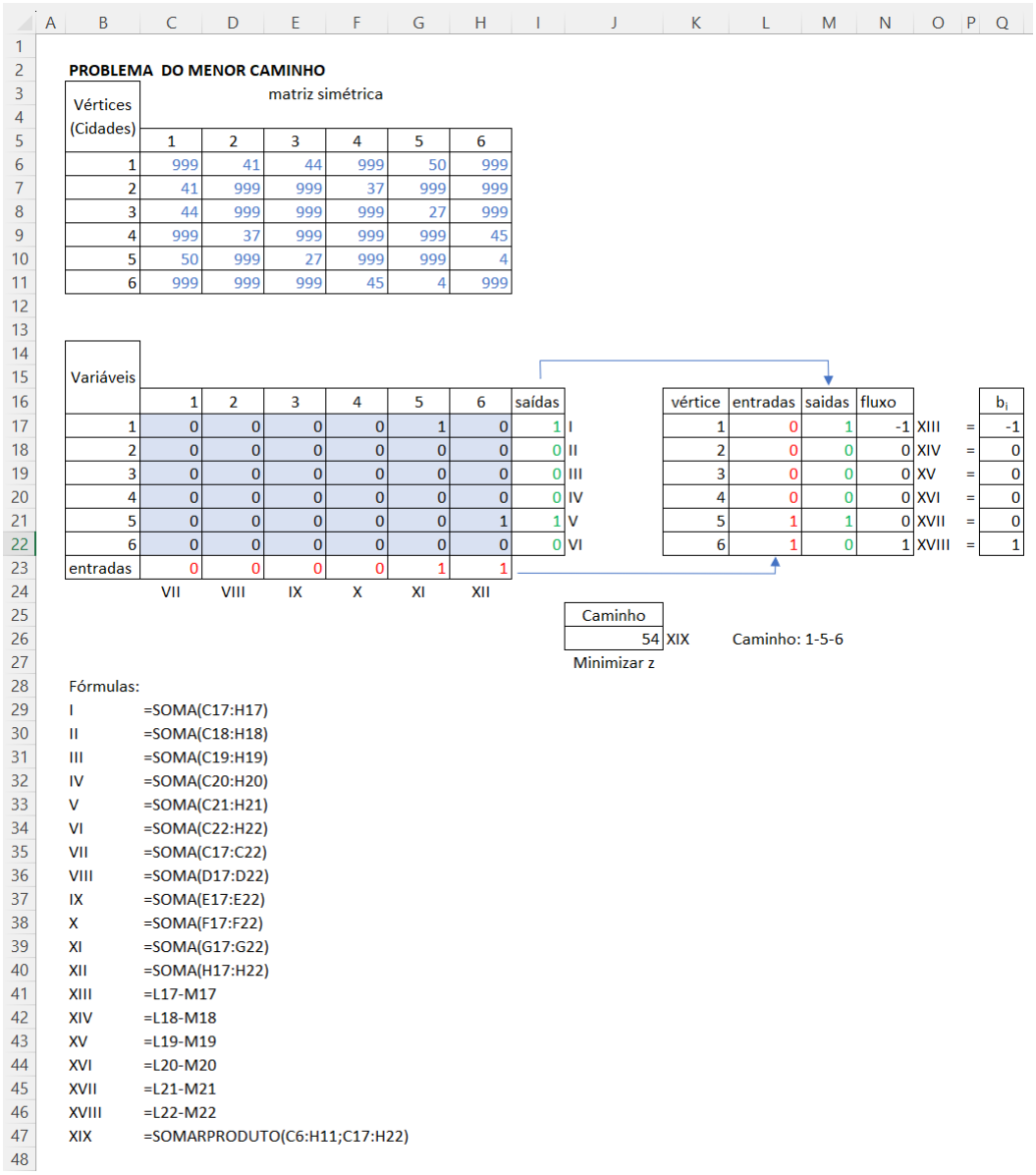
I	=SOMA(D16:H16)
II	=SOMA(D17:H17)
III	=SOMA(D18:H18)
IV	=SOMA(D19:H19)
V	=SOMA(D16:D19)
VI	=SOMA(E16:E19)
VII	=SOMA(F16:F19)
VIII	=SOMA(G16:G19)
IX	=SOMA(H16:H19)
X	=SOMA(D20:H20)
XI	=L18
XII	=L19
XIII	=SOMARPRODUTO(D7:H10;D16:H19)



Fonte: a autora.

Exemplo 5 (Figura 61):

Figura 61 – Excel – exemplo 5



PROBLEMA DO MENOR CAMINHO

matriz simétrica

Vértices (Cidades)	1	2	3	4	5	6
1	999	41	44	999	50	999
2	41	999	999	37	999	999
3	44	999	999	999	27	999
4	999	37	999	999	999	45
5	50	999	27	999	999	4
6	999	999	999	45	4	999

Variáveis	1	2	3	4	5	6	saídas	
1	0	0	0	0	1	0	1	I
2	0	0	0	0	0	0	0	II
3	0	0	0	0	0	0	0	III
4	0	0	0	0	0	0	0	IV
5	0	0	0	0	0	1	1	V
6	0	0	0	0	0	0	0	VI
entradas	0	0	0	0	1	1		
	VII	VIII	IX	X	XI	XII		

vértice	entradas	saidas	fluxo			b_i
1	0	1	-1	XIII	=	-1
2	0	0	0	XIV	=	0
3	0	0	0	XV	=	0
4	0	0	0	XVI	=	0
5	1	1	0	XVII	=	0
6	1	0	1	XVIII	=	1

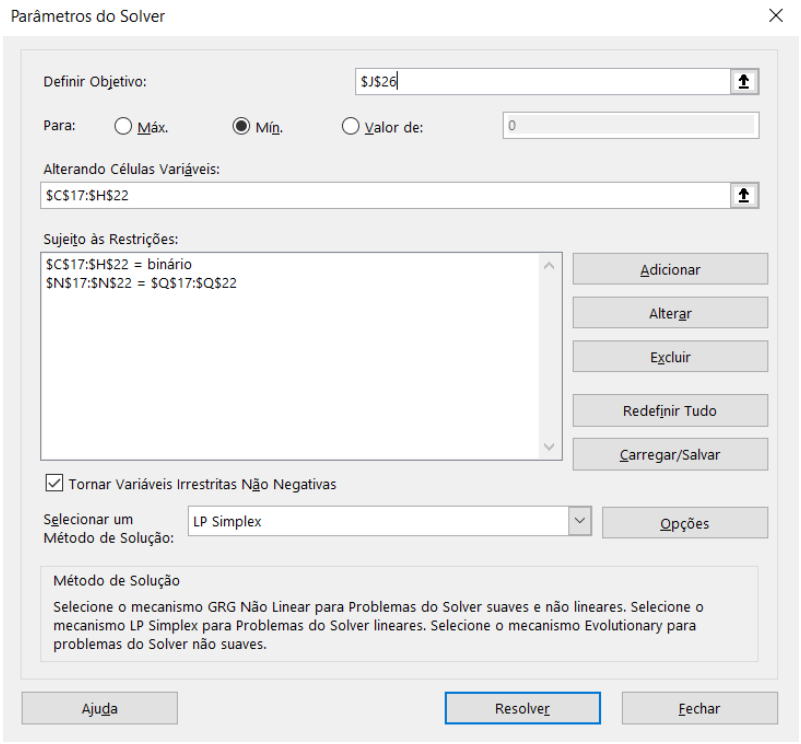
Caminho	
54	XIX

Caminho: 1-5-6

Minimizar z

Fórmulas:

I	=SOMA(C17:H17)
II	=SOMA(C18:H18)
III	=SOMA(C19:H19)
IV	=SOMA(C20:H20)
V	=SOMA(C21:H21)
VI	=SOMA(C22:H22)
VII	=SOMA(C17:C22)
VIII	=SOMA(D17:D22)
IX	=SOMA(E17:E22)
X	=SOMA(F17:F22)
XI	=SOMA(G17:G22)
XII	=SOMA(H17:H22)
XIII	=L17-M17
XIV	=L18-M18
XV	=L19-M19
XVI	=L20-M20
XVII	=L21-M21
XVIII	=L22-M22
XIX	=SOMARPRODUTO(C6:H11;C17:H22)



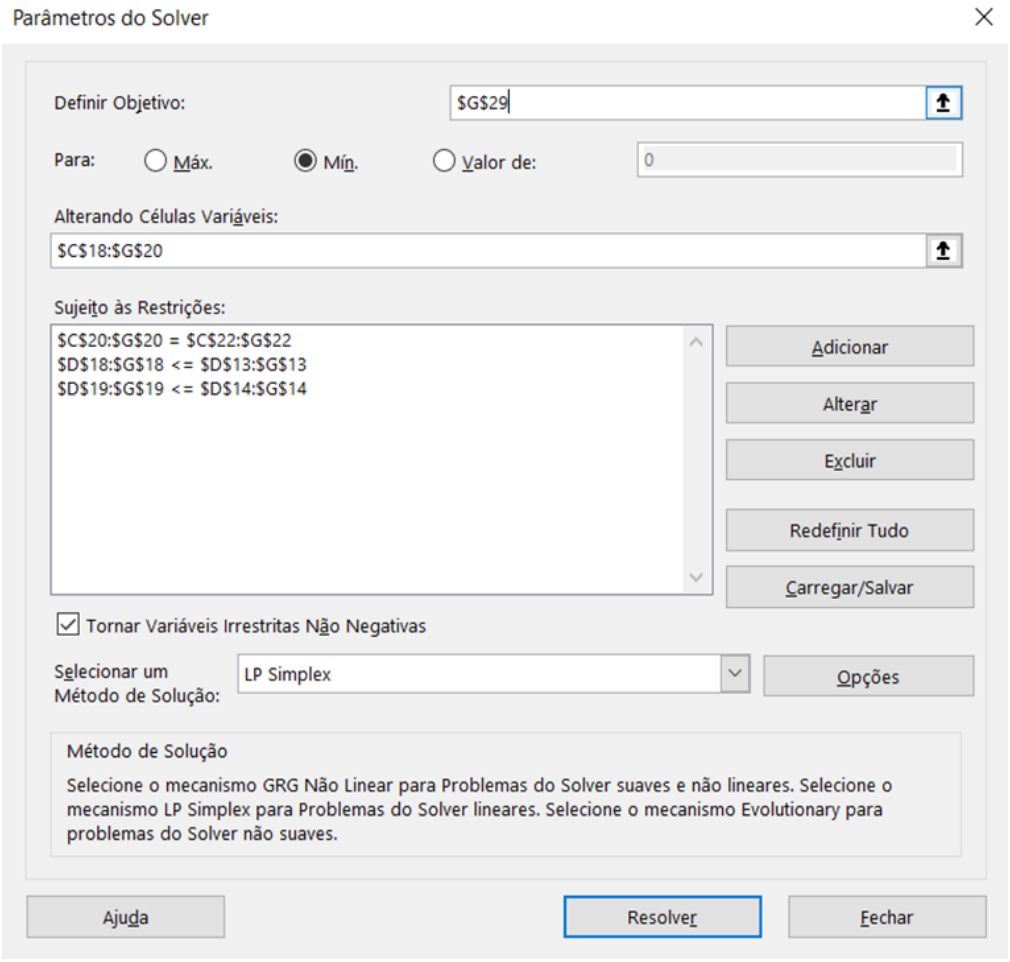
Fonte: a autora

Exemplo 6 (Figura 62):

Figura 62 – Excel – exemplo 6

3	**PROBLEMA DO PLANEJAMENTO AGREGADO**						
4	Custo em Reais						
5	Tipos	Custos Unitários de mão de obra					
6		Anterior	Mês 1	Mês 2	Mês 3	Mês 4	
7	Reg. HN		15	15	15	15	
8	Reg. HE		22	22	22	22	
9	Estoque	3	3	3	3	3	
10	Oferta						
11	Tipos	Oferta (em unidades)					
12		Anterior	Mês 1	Mês 2	Mês 3	Mês 4	
13	Reg. HN		150.000	150.000	150.000	150.000	
14	Reg.HE		50.000	50.000	50.000	50.000	
15	Produção		<=	<=	<=	<=	respectivas ofertas
16	Oferta	Quantidade Produzida no período					
17		Anterior	Mês 1	Mês 2	Mês 3	Mês 4	
18	Reg. HN	0	150.000	150.000	150.000	150.000	
19	Reg. HE	0	0	20.000	0	0	
20	Estoque	0	30.000	0	30.000	0	
21		=	=	=	=	=	Estoques
22	Estoque	0	30.000	0	30.000	0	
23			I	II	III	IV	
24							
25	Demanda	Demanda (em unidades)					
26	Demanda		120.000	200.000	120.000	180.000	
27							
28						Custo Total (US$)	
29						9.620.000,00	
30						Minimizar z	

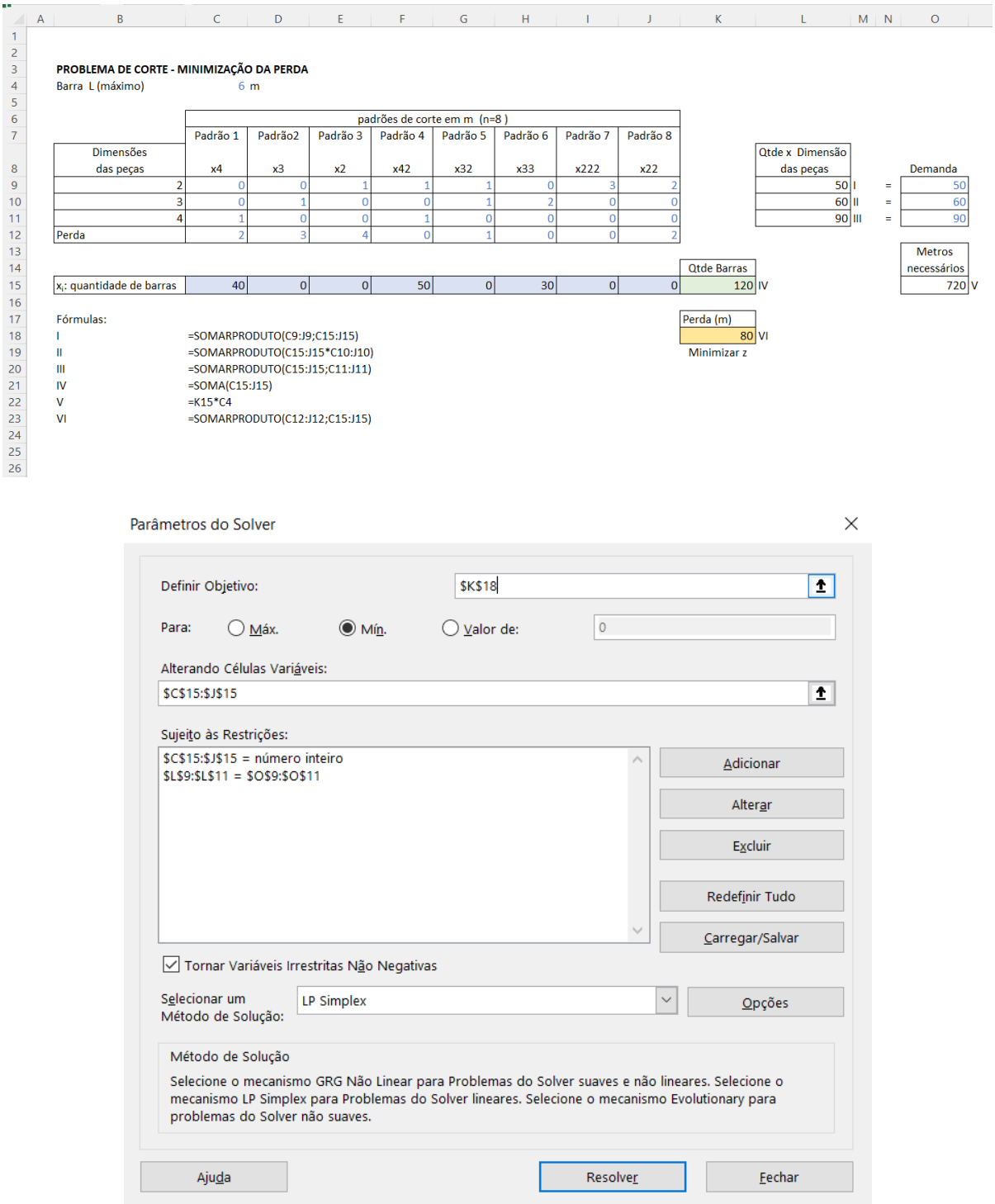
31		
32		
33		
34	Fórmulas:	
35	I	=-D26+C20+D18+D19
36	II	=-E26+D20+E18+E19
37	III	=-F26+E20+F18+F19
38	IV	=-F26+E20+F18+F19
39	V	=SOMARPRODUTO(C7:G9;C18:G20)
40		



Fonte: a autora

Exemplo 7 (Figura 63):

Figura 63 – Excel – exemplo 7



PROBLEMA DE CORTE - MINIMIZAÇÃO DA PERDA

Barra L (máximo) 6 m

	padrões de corte em m (n=8)							
Dimensões das peças	Padrão 1 x4	Padrão2 x3	Padrão 3 x2	Padrão 4 x42	Padrão 5 x32	Padrão 6 x33	Padrão 7 x222	Padrão 8 x22
2	0	0	1	1	1	0	3	2
3	0	1	0	0	1	2	0	0
4	1	0	0	1	0	0	0	0
Perda	2	3	4	0	1	0	0	2

Qtde x Dimensão das peças			Demanda
50	I	=	50
60	II	=	60
90	III	=	90

		Qtde Barras	
x_i: quantidade de barras	40 0 0 50 0 30 0 0	120	IV

Metros necessários: 720 V

Fórmulas:

I	=SOMARPRODUTO(C9:J9;C15:J15)
II	=SOMARPRODUTO(C15:J15*C10:J10)
III	=SOMARPRODUTO(C15:J15;C11:J11)
IV	=SOMA(C15:J15)
V	=K15*C4
VI	=SOMARPRODUTO(C12:J12;C15:J15)

Perda (m): 80 VI

Minimizar z

Fonte: a autora

Exemplo 8 (Figura 64):

Figura 64 – Excel – exemplo 8

PROBLEMA DE CORTE - MINIMIZAÇÃO DO NÚMERO DE BARRAS

Barra L (máximo) 6 m

	padrões de corte em m (n=8)							
Dimensões das peças	Padrão 1 x4	Padrão2 x3	Padrão 3 x2	Padrão 4 x42	Padrão 5 x32	Padrão 6 x33	Padrão 7 x222	Padrão 8 x22
2	0	0	1	1	1	0	3	2
3	0	1	0	0	1	2	0	0
4	1	0	0	1	0	0	0	0
Perda	2	3	4	0	1	0	0	2

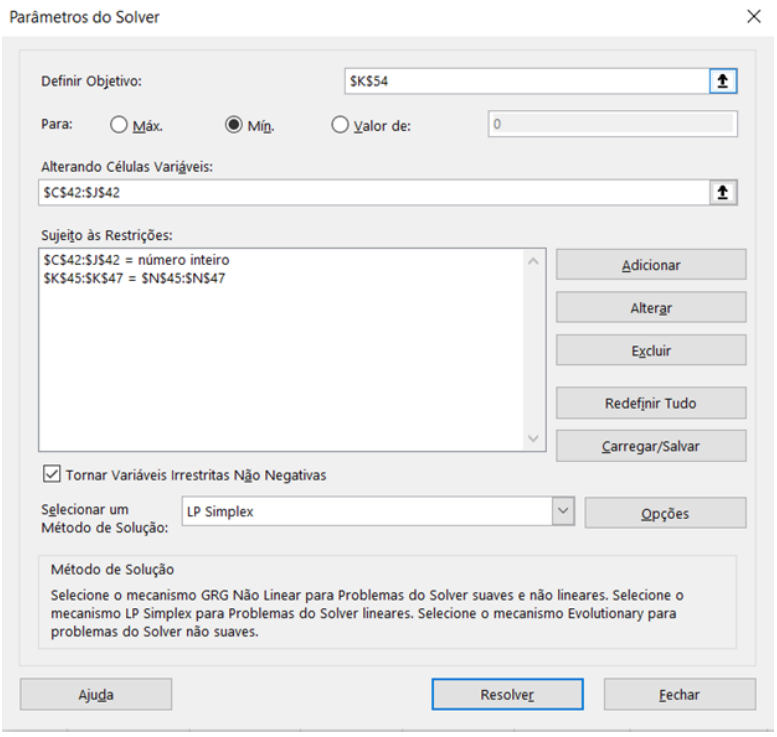
x_i: quantidade de barras	40	0	0	50	0	30	0	0

Dimensões das peças	Padrão 1 * Qtde Barras	Padrão 2 * Qtde Barras	Padrão 3 * Qtde Barras	Padrão 4* Qtde Barras	Padrão 5 * Qtde Barras	Padrão 6 * Qtde Barras	Padrão 7 * Qtde Barras	Padrão 8 * Qtde Barras	Total			Demanda	Demanda * Dimensão peça
2	0	0	0	50	0	0	0	0	50	I	=	50	100
3	0	0	0	0	0	60	0	0	60	II	=	60	180
4	40	0	0	50	0	0	0	0	90	III	=	90	360
													640
perda	80	0	0	0	0	0	0	0	80				720
	IV	V	VI	VII	VIII	IX	X	XI	XII				

Qtde barras		Metros necessários	
120	XIII	720	XIV
Minimizar z			

Fórmulas:

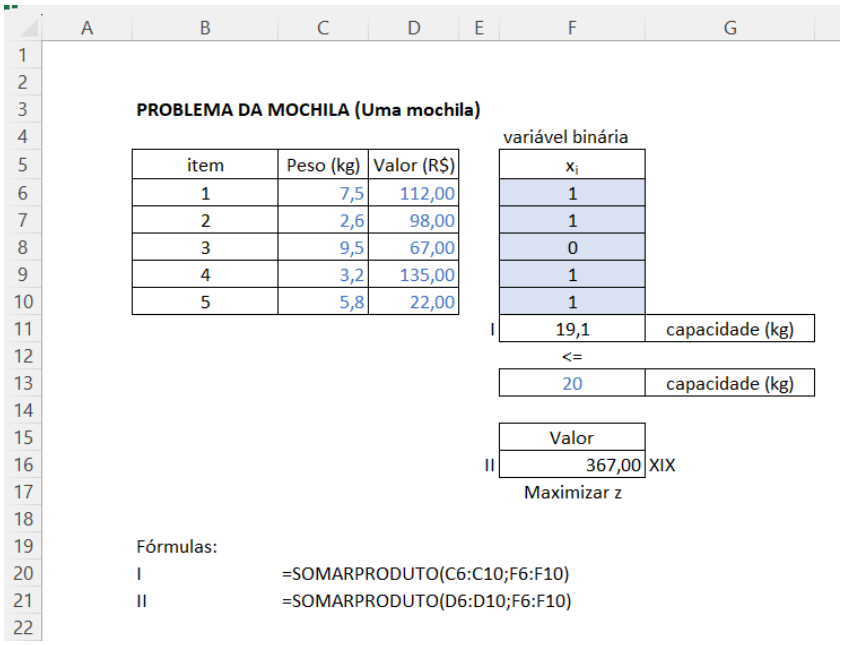
I	=SOMA(C45:J45)
II	=SOMA(C46:J46)
III	=SOMA(C47:J47)
IV	=C42*C39
V	=D42*D39
VI	=E42*E39
VII	=F42*F39
VIII	=G42*G39
IX	=H42*H39
X	=I42*I39
XI	=J42*J39
XII	=SOMA(C50:J50)
XIII	=SOMA(C42:J42)
XIV	=K54*C31



Fonte: a autora

Exemplo 9 (Figura 65):

Figura 65 – Excel – exemplo 9



PROBLEMA DA MOCHILA (Uma mochila)

item	Peso (kg)	Valor (R$)
1	7,5	112,00
2	2,6	98,00
3	9,5	67,00
4	3,2	135,00
5	5,8	22,00

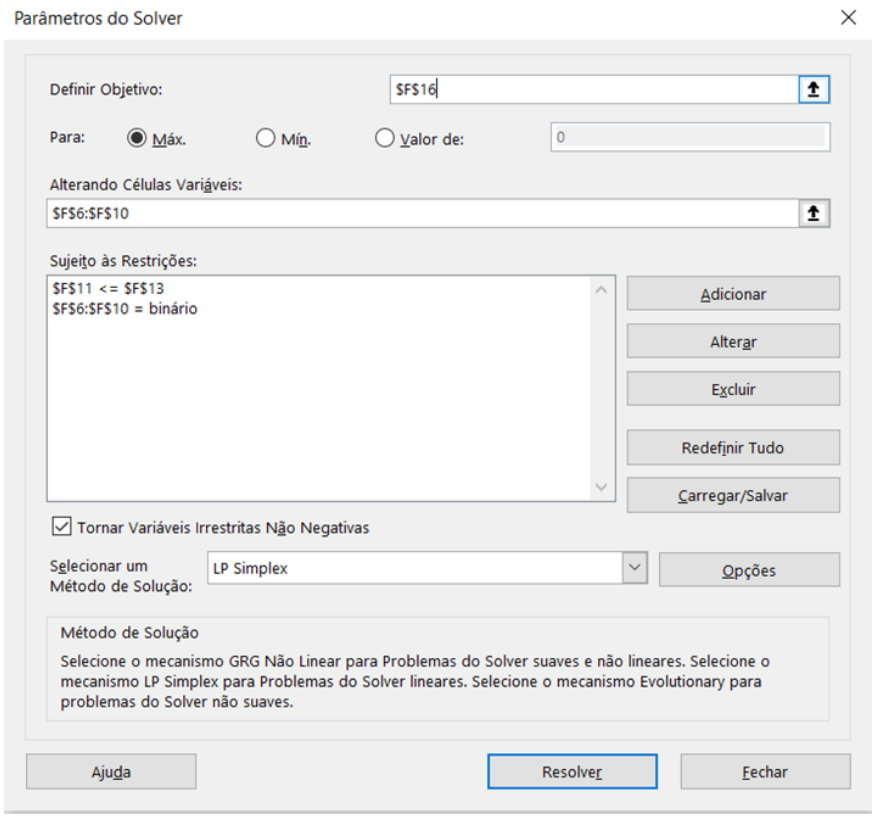
	variável binária x_i	
	1	
	1	
	0	
	1	
	1	
I	19,1	capacidade (kg)
	<=	
	20	capacidade (kg)

	Valor	
II	367,00	XIX
	Maximizar z	

Fórmulas:

I =SOMARPRODUTO(C6:C10;F6:F10)

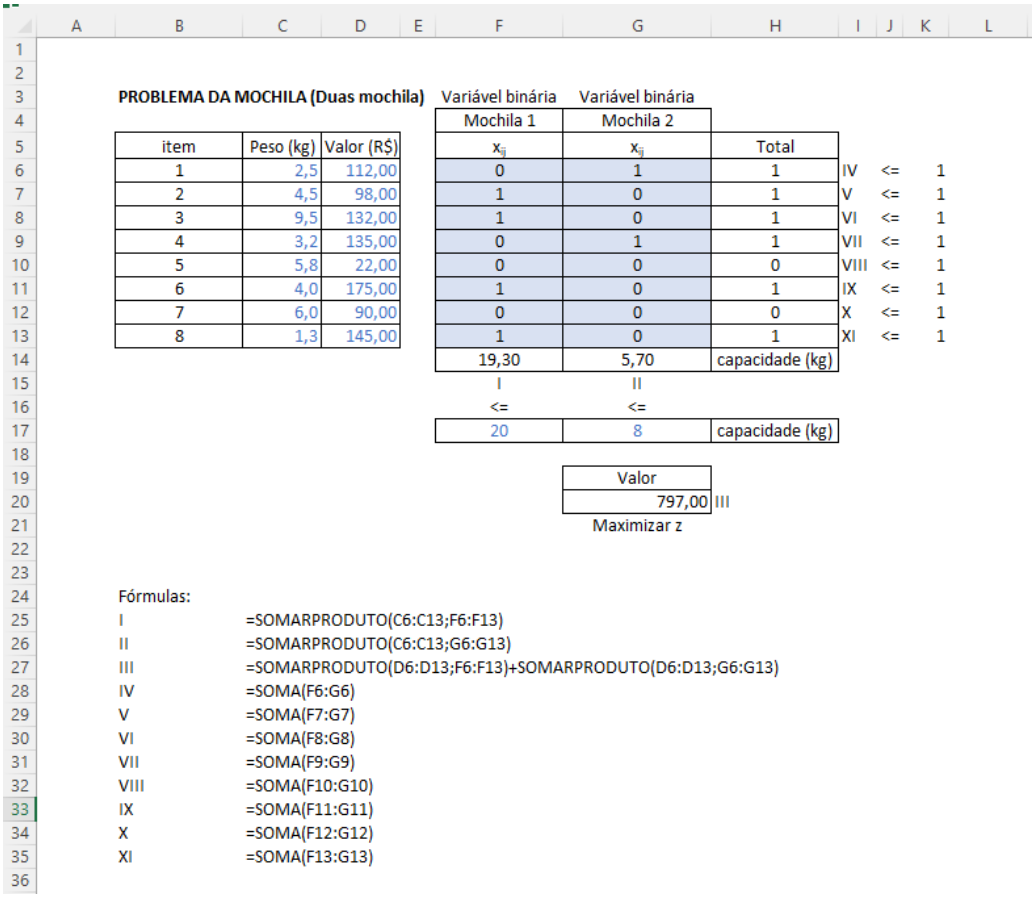
II =SOMARPRODUTO(D6:D10;F6:F10)



Fonte: a autora

Exemplo 10 (Figura 66):

Figura 66 – Excel – exemplo 10



PROBLEMA DA MOCHILA (Duas mochila)

item	Peso (kg)	Valor (R$)
1	2,5	112,00
2	4,5	98,00
3	9,5	132,00
4	3,2	135,00
5	5,8	22,00
6	4,0	175,00
7	6,0	90,00
8	1,3	145,00

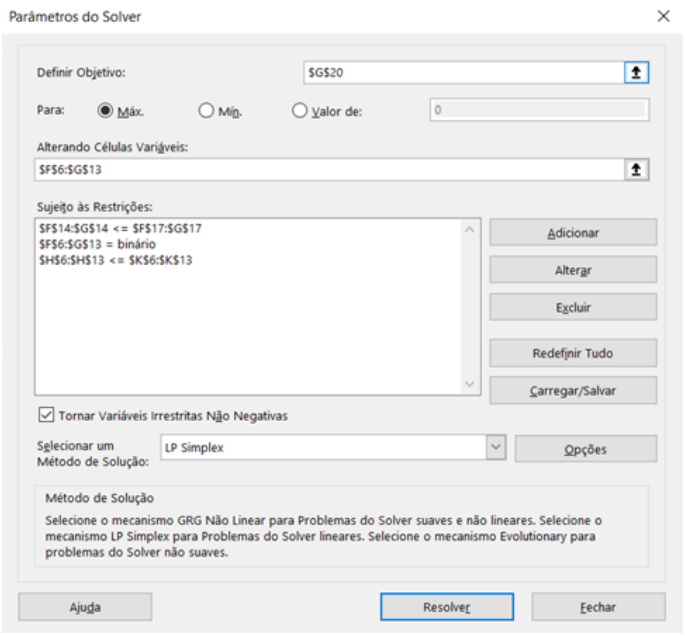
Variável binária	Variável binária				
Mochila 1	Mochila 2				
x_{ij}	x_{ij}	Total			
0	1	1	IV	<=	1
1	0	1	V	<=	1
1	0	1	VI	<=	1
0	1	1	VII	<=	1
0	0	0	VIII	<=	1
1	0	1	IX	<=	1
0	0	0	X	<=	1
1	0	1	XI	<=	1
19,30	5,70	capacidade (kg)			
I	II				
<=	<=				
20	8	capacidade (kg)			

Valor	
797,00	III

Maximizar z

Fórmulas:

I	=SOMARPRODUTO(C6:C13;F6:F13)
II	=SOMARPRODUTO(C6:C13;G6:G13)
III	=SOMARPRODUTO(D6:D13;F6:F13)+SOMARPRODUTO(D6:D13;G6:G13)
IV	=SOMA(F6:G6)
V	=SOMA(F7:G7)
VI	=SOMA(F8:G8)
VII	=SOMA(F9:G9)
VIII	=SOMA(F10:G10)
IX	=SOMA(F11:G11)
X	=SOMA(F12:G12)
XI	=SOMA(F13:G13)



Fonte: a autora

ANEXOS

ANEXO A - Grade curricular

A Figura 67 apresenta a grade curricular do curso de Análise e Desenvolvimento de Sistemas, da FATEC-SP, turma 2021.

Figura 67 – Grade das disciplinas de ADS FATEC-SP

TECNOLOGIA EM ANÁLISE E DESENVOVIMENTO DE SISTEMAS

Análise e Desenvolvimento de Sistemas - noturno em oito semestres

1º Semestre	2º Semestre	3º Semestre	4º Semestre	5º Semestre	6º Semestre	7º Semestre	8º Semestre
Programação em Micro-informática (4)	Linguagem de Programação (4)	Estruturas de Dados (4)	Engenharia de Software I (4)	Engenharia de Software II (4)	Engenharia de Software III (4)	Laboratório de Engenharia de Software (4)	Seminários em Informática (2)
Algoritmos e Lógica de Programação (4)	Comunicação e Expressão (4)	Sistemas Operacionais I (4)	Programação Orientada a Objetos (4)	Sistemas Operacionais II (4)	ESCOLHA I - Laboratório de Banco de Dados - Sistemas distribuídos (4)	ESCOLHA II - Auditoria de Sistemas - Laboratório de Redes (4)	ESCOLHA III - Inteligência Artificial - Projeto de Redes Computadores (4)
Matemática Discreta (4)	Matemática (4)	Estatística Aplicada (4)	Programação Linear e Aplicações (4)	Banco de Dados (4)	Redes de Computadores (4)	Gestão de Projetos (4)	ESCOLHA IV - Tópicos Especiais em Informática -Sist. Oper. de Redes de Computadores (4)
Administração Geral (4)	Sistemas de Informação (4)	Eletiva I (4)	Eletiva II (4)	Eletiva III (4)	Segurança da Informação (2) / Sociedade e Tecnologia (2)	Gestão de Equipes (2) / Empreende-dorismo (2)	Gestão e Governança de Tecnologia da Informação (4)
Arquitetura e Organização de Computadores (4)	Laboratório de Arquitetura e Organização de Computadores (4)	Laboratório de Hardware (2)	Contabilidade (2)	Interação Humano Computador (2)	Economia e Finanças (2)	Metodologia da Pesquisa Científico-Tecnológica (2)	Ética e responsabilida de profissional (2)
Aulas: Semanais 20 Semestrais 400	Aulas: Semanais 20 Semestrais 400	Aulas: Semanais 18 Semestrais 360	Aulas: Semanais 18 Semestrais 360	Aulas: Semanais 18 Semestrais 360	Aulas: Semanais 18 Semestrais 360	Aulas: Semanais 18 Semestrais 360	Aulas: Semanais 16 Semestrais 320

Estágio Curricular (a partir do 3º semestre) - 240 horas Trabalho de Graduação (a partir do quinto semestre) - 160 horas

DISTRIBUIÇÃO DAS AULAS POR EIXO FORMATIVO

Disciplinas BÁSICAS	Aula	%	Disciplinas PROFISSIONAIS	Aula	%
Cálculo e Estatística	320	10,96%		760	26,03%
Economia	80	2,74%		1200	41,09%
Comunicação	80	2,74%		240	8,22%
Administração	80	2,74%	Transversal (multidisciplinar)	160	5,48%
TOTAL	**560**	19,18%	**TOTAL**	**2360**	80,82%

Eixo tecnológico no CNCST: Informação e comunicação

RESUMO DE CARGA HORÁRIA

2920 aulas → 2433 horas (atende CNCST) + 240 horas de ESTÁGIO CURRICULAR + 160 horas do Trabalho de Graduação = **2833 HORAS**

ROGRAMAÇÃO LINEAR E APLICAÇÕES – 80 aulas

Ementa: Matrizes. Sistemas Lineares. Programação Linear: Método Gráfico e Método Simplex. Aplicações: Método do Transporte.

Bibliografia básica:

ANDRADE, E. L. Introdução à pesquisa operacional. 4.ed. LTC, 2009.

KOLMAN, B. Introdução à álgebra linear com aplicações. 8.ed. LTC, 2006.

Fonte: http://www.fatecsp.br/paginas/ProjetoPedagogico/ProjPed-ADS.pdf