Formulário 3

RELATÓRIO PARCIAL DE INICIAÇÃO CIENTÍFICA/UEMS

Edital: UEMS N° 03/2023

Acadêmico(a): Nicolas Herculano Pires

Orientador(a): Prof. Dr. Cleber Valgas Gomes Mira

Título do projeto: Estudo de heurísticas e modelagem PLI para um problema de seleção e agendamento de portfólio de projetos

Protocolo de aprovação no comitê de ética – **(enviar o comprovante - caso necessário):**

Curso de graduação: Sistemas de Informação

Unidade: Dourados

Área de conhecimento: Ciências Exatas e da Terra

1. RESUMO (máximo 250 palavras)

A pesquisa deste projeto dedica-se ao exame de técnicas heurísticas e à modelagem de Programação Linear Inteira (PLI) com o propósito de aprimorar a seleção e o planejamento temporal de projetos em entidades organizacionais, levando em consideração variáveis como custos, prazos e outros parâmetros relevantes, com o intuito de otimizar critérios específicos sob certas limitações. O estudo avança na proposta de modelar o Problema de Seleção e Agendamento de Projetos (PPSSP) como um problema de PLI, empregando resolvedores de

PLI para efetuar comparações de resultados com a abordagem meta-heurística GRASP. O escopo central é investigar a qualidade das soluções produzidas tanto pelos resolvedores quanto pela heurística em questão.

PALAVRAS-CHAVE

PPSSP, Portfólio de Projetos, Otimização, Meta-heurísticas, Programação Linear Inteira, GRASP, PLI.

2. SOBRE A PESQUISA DE IC, RESPONDA:

2.1 Os objetivos da pesquisa foram atingidos até o presente momento? Justifique em caso de resposta negativa.
(X) SIM ( ) NÃO

2.2 O projeto vem sendo executado a contento e em conformidade com o cronograma proposto? Justifique em caso de resposta negativa.
(X) SIM ( ) NÃO

2.3 Houve alguma mudança? *Justifique em caso de alteração.*
( )Título ( ) Metodologia ( ) Carga Horária ( ) Cronograma (X) Nenhuma

2.4 Descreva as ATIVIDADES realizadas e os RESULTADOS PARCIAIS obtidos no período:

Inicialmente, executaram-se 8 atividades sob supervisão do orientador acadêmico, com o objetivo de elucidar de maneira precisa os conceitos que foram explorados ao decorrer deste trabalho. Salienta-se também que foram realizados exercícios para fixação das atividades a seguir:

1. **Estudo do PPSSP para melhor conhecimento:**
   A princípio, foi realizado um estudo base para compreensão do que é um problema de otimização, o qual busca a melhor solução entre várias possíveis, seguindo certas regras, e também do que são heurísticas, as quais são métodos práticos para encontrar soluções rápidas e razoavelmente boas, úteis quando métodos exatos são muito complexos ou tempo de execução demasiadamente elevado.

   Foram realizadas leituras de TCCs e artigos científicos que foram apresentados na proposta de Iniciação Científica, sendo que o artigo **[1]** apresenta uma solução eficaz para um problema real de seleção e programação de projetos em uma empresa de geração de energia elétrica, oferecendo uma abordagem metodológica baseada na heurística GRASP (Greedy Randomized Adaptive Search Procedure) que supera as soluções manuais em termos de redução de risco. Por outro lado, o TCC **[2]** revisado aborda a utilização da heurística BRKGA (Biased Random-Key Genetic Algorithm) para a resolução de uma versão PPSSPP. De acordo

com os experimentos realizados no trabalho de Tanaka, a heurística GRASP apresenta melhores resultados do que a heurística BRKGA para uma mesma versão do PPSSP.

Observação: O GRASP será abordado no tópico posterior.

2. **Revisão da meta-heurística GRASP para ser utilizada ao final deste trabalho:**

   Foi realizado também o estudo da meta-heurística GRASP, que significa *Greedy Randomized Adaptive Search Procedure* (Procedimentos Adaptativos de Busca Aleatória Gulosa). Esta heurística é uma técnica de otimização para resolver problemas complexos de busca e otimização e combina elementos de construção gulosa de soluções com busca local e aleatoriedade, buscando um equilíbrio entre exploração de novas áreas do espaço de solução e refinamento das soluções existentes para encontrar soluções de boa qualidade em um tempo razoável. Em particular, o estudo do GRASP foi realizado quando este é aplicado ao PPSSP.

3. **Busca por um ambiente de desenvolvimento:**

   Ao decorrer do tempo, foi identificada a necessidade de um ambiente de desenvolvimento para a implementação das. Foi adquirido um computador desktop (*Intel Core* i7 3770, Armazenamento de RAM: 2x 4GB 1333Mhz (totalizando 8GB em dual-channel), Armazenamento de ROM: SSD KingSton 240GB, Placa de vídeo: Radeon RX 550 2GB), e foi adotado o sistema operacional GNU-Linux (distribuição Debian, versão 12) para o melhor desempenho da execução dos programas (resolvedor GLPK versão 5.0). O compilador de linguagem C adotado no trabalho é o GCC versão 13.2.0.

4. **Estudo de modelagem PLI:**

   Foram realizadas leituras sobre a programação linear, baseando-se em livros **[3]**, tutoriais **[4]** e *websites* **[5]**.

   Um problema de Programação Linear (PL) é um tipo de problema de otimização em que o objetivo é maximizar ou minimizar uma função linear, conhecida como função objetivo, sujeita a um conjunto de restrições lineares. Essas restrições definem um espaço de solução viável, e a solução ótima do problema, se existir, está localizada em um ponto deste espaço.

   A relevância da Programação Linear está na sua aplicabilidade em diversas áreas, incluindo economia, engenharia, gestão de operações, ciências da computação, entre outras. Os problemas de PL podem ser utilizados para otimizar recursos, reduzir custos, aumentar lucros, ou encontrar a melhor maneira de alocar recursos limitados. Por exemplo, uma empresa pode usar a PL para determinar a quantidade de produtos a serem produzidos em diferentes linhas de produção, de modo a maximizar o lucro ou minimizar os custos, respeitando as limitações de capacidade de produção e demanda do mercado.

Existem algoritmos eficientes para resolver problemas de Programação Linear, sendo o mais conhecido o Método Simplex, desenvolvido por George Dantzig nos anos 1940. O Método Simplex navega pelas arestas do poliedro definido pelas restrições lineares até encontrar a solução ótima. Embora, na prática, o Método Simplex seja muito eficiente, teoricamente ele pode ter um tempo de execução exponencial nos piores casos. Outro algoritmo importante é o Método dos Pontos Interiores, que, diferentemente do Simplex, move-se pelo interior do poliedro de soluções viáveis e pode oferecer garantias de tempo polinomial para alguns casos. (Trecho retirado de uma conclusão após a leitura do livro "*Introdução a Algoritmos*" **[3]**).

Além da Programação Linear, também foi estudado sobre a Programação Linear Inteira (PLI) e as suas características particulares **[3]**.
Problemas de otimização modelados como problemas PLI possuem a propriedade de que as variáveis de solução são restritas a serem valores inteiros.

5. Estudo da linguagem C para a utilização da biblioteca do Solver GLPK:

Nós estudamos a biblioteca principal do GLPK (com o nome "<glpk.h>") implementada em C para utilizá-la em um programa que tenta encontrar soluções viáveis para alguns problemas de PLI.
Funções Utilizadas:

glp_create_prob(); Utilizada para criar um problema de PLI.

glp_term_out(); Utilizada para ativar os Logs de saída.

glp_set_prob_name(); Utilizada para dar um nome ao problema criado.

glp_set_obj_dir(); Utilizada para definir se o objetivo é maximizar ou minimizar.

glp_add_cols(); Utilizada para inserir colunas ao problema.

glp_add_rows(); Utilizada para inserir linhas ao problema.

glp_set_col_kind(); Utilizada para definir o conjunto que uma coluna pertence (Binário, real, inteiro, entre outros).

glp_set_row_kind(); Utilizada para definir o conjunto que uma linha pertence (Binário, real, inteiro, entre outros).

glp_set_col_bnds(); Utilizada para definir os limites dos itens da coluna passada como parâmetro.

glp_set_row_bnds(); Utilizada para definir os limites dos itens da linha passada como parâmetro.

glp_set_obj_coef(); Utilizada para definir a função objetivo.

glp_set_mat_row(); Utilizada para definir uma restrição em uma determinada linha.

glp_init_smcp(); Utilizada para inicializar as estruturas utilizadas pelo algoritmo simplex.

glp_simplex(); Utilizada para resolver o problema descrevido utilizando o algoritmo simplex

glp_init_iocp(); Utilizada para inicializar parâmetros necessários para resolução do problema utilizando o algoritmo Branch and Bound, além de habilitar o resolvedor para trabalhar com pré-processamento e heurísticas para a resolução de problemas .

glp_intopt(); Utilizada para resolver o problema de acordo com o método passado pelo escopo.

glp_get_status(); Utilizada para informar se o resolvedor encontrou uma solução ótima ou se teve algum problema na modelagem, restrição e afins.

glp_delete_prob(); Utilizada para deletar o problema criado e liberar a memória ocupada pelo resolvedor.

6. Resolução de problemas clássicos de otimização utilizando o Solver GLPK:

O GLPK (GNU Linear Programming Kit), resolvedor de PLI que escolhemos para este trabalho, possui uma biblioteca escrita em C com um vasto número de funções, as quais foram estudadas por meio da própria documentação fornecida pela GNU. Além disso, a documentação também inclui exemplos clássicos de problemas de otimização, dentre eles, o Problema da Mochila (*Knapsack Problem*). Neste problema, dada uma mochila de capacidade limitada e um conjunto com uma certa quantidade de itens, cada item com um peso P e um valor V, busca-se encontrar um subconjunto dos itens que maximize o valor total transportado pela mochila sem ultrapassar sua capacidade. Este exemplo de problema de otimização foi utilizado para aprimorar a compreensão do funcionamento do GLPK. Foram realizados alguns experimentos para instâncias pequenas do Problema da Mochila.

7. Estudo de MathProg:

Durante o desenvolvimento de um programa em C que utiliza bibliotecas do GLPK para resolver uma versão simplificada do problema PPSSP, foi identificada uma dificuldade para a implementação deste código, devido a algumas características do resolvedor em conflito com o objetivo declarado pelo orientador. Nós concluímos que seria mais conveniente utilizar a linguagem de modelagem do MathProg, ao invés da utilização da biblioteca do GLPK em C, isso por conta que a linguagem MathProg se trata de uma linguagem interpretada, logo, a modelagem de como será realizada a solução, fica restrita às instruções passadas entre o resolvedor e a máquina.

MathProg é uma linguagem de modelagem para problemas de otimização, como programação linear e inteira, usada para expressar modelos matemáticos de forma clara **[4]**. É compatível com o resolvedor GLPK e facilita a descrição formal de problemas complexos de otimização.

A seguir, um exemplo de um problema PLI conhecido, em conjunto de sua tradução para MathProg:

Problema: Maximização do Lucro de Produção

Uma fábrica produz dois tipos de produtos, A e B. Cada unidade do produto A vende por $100 e do produto B por $150. A produção de cada unidade de A requer 1 hora de trabalho no Departamento X e 2 horas no Departamento Y. Cada unidade de B requer 3 horas no Departamento X e 2 horas no Departamento Y. O Departamento X tem disponibilidade de 120 horas de trabalho, enquanto o Departamento Y tem disponibilidade de 160 horas. Devido a limitações de mercado, podem ser vendidas no máximo 40 unidades do produto A e 30 unidades do produto B por período. O objetivo é maximizar o lucro total da produção.

Forma Matemática:

- Variáveis de decisão:

  x_A Número de unidades do produto A a serem produzidas.

  x_B Número de unidades do produto A a serem produzidas.

Cada uma dessas variáveis deve ser inteira, pois não podemos produzir uma fração de um produto.

Função Objetivo:

$$Maximizar\ Z\ =\ 100x_A\ +\ 150x_B$$

Sujeito a:

- $x_A + 3x_B \leq 120$ (Restrição do Departamento X)
- $2x_A + 2x_B \leq 160$ (Restrição do Departamento Y)
- $x_A \leq 40$ (Restrição de venda para A)
- $x_B \leq 30$ (Restrição de venda para B)
- $x_A, x_B \geq 0$ e inteiros

Tradução para MathProg:

```
param max_A := 40; # Máximo de unidades do produto A
param max_B := 30; # Máximo de unidades do produto B

var x_A >= 0, integer; # Produção de A
var x_B >= 0, integer; # Produção de B

maximize profit: 100*x_A + 150*x_B; # Função objetivo

# Restrições
s.t. ConstraintX: x_A + 3*x_B <= 120; # Departamento X
s.t. ConstraintY: 2*x_A + 2*x_B <= 160; # Departamento Y
s.t. SaleA: x_A <= max_A; # Vendas de A
s.t. SaleB: x_B <= max_B; # Vendas de B

solve;

printf "Produção de A: %d\n", x_A;
printf "Produção de B: %d\n", x_B;
printf "Lucro máximo: %d\n", 100*x_A + 150*x_B;

data;

end;
```

Descrição:

Neste exemplo de MathProg, as variáveis x_A e x_B representam a quantidade de produtos A e B a serem produzidos, respectivamente. A função objetivo profit busca maximizar o lucro total baseado nos preços de venda e nas quantidades produzidas. As restrições ConstraintX e ConstraintY garantem que a produção não exceda a capacidade dos departamentos X e Y, enquanto SaleA e SaleB asseguram que a produção não ultrapasse as quantidades máximas vendáveis de A e B, respectivamente.

8. **Modelagem de uma versão do PPSSP e experimentos** (Em andamento)**:** A versão do PPSSP abordada na proposta do projeto científico está em andamento, logo, **todos os itens da modelagem estão sujeitos a mudanças**. A seguir, uma visão geral de como está a modelagem:

CONJUNTOS:

- I Conjunto dos projetos
- W Conjunto de pontos de atenção
- $MESES$ Conjunto de meses no horizonte de planejamento
- $DURMAX$ Conjunto de durações máximas de projetos

PARÂMETROS:

- R_w para cada $w \in W$
  Risco associado ao ponto de atenção w.
  (Inteiro)

- G_{pw} para cada $p \in I,\ w \in W$
  Matriz a qual indica se o projeto p resolve o ponto de atenção.
  (Binário)

- d_p para cada $p \in I$
  Duração em meses de um projeto p.
  (Inteiro)

- $custo_{pd}$ para cada $p \in I,\ d \in DURMAX$
  Custo de cada projeto p no mês d.
  (Inteiro)

- C_{mes} para cada $m \in MESES$
  Orçamento mensal.
  (Inteiro)

- $deadline_w$ para cada $w \in W$
  Prazo limite para resolver um ponto de atenção w.
  (Inteiro)

- $intoleravel_w$ para cada $w \in W$

Indica se o ponto de atenção w é intolerável.
(Binário)

- HP
  Horizonte de Planejamento.
  (Inteiro)

VARIÁVEIS DE DECISÃO

- x_p, para cada $p \in I$ — Binária, indica se o projeto p é selecionado.
  (Binário)

- m_p, para cada $p \in I$ — Mês de início do projeto p.
  (Inteiro)

- C_p, para cada $p \in I$ — Custo total de um projeto p.
  (Inteiro)

FUNÇÃO OBJETIVO

$$Minimizar \sum_{p \in I} C_p$$

O objetivo é minimizar o custo total dos projetos selecionados.

RESTRIÇÕES

- Prazo Limite para Resolução de Pontos de Atenção Intoleráveis:

  $m_p + d_p \leq deadline_w,\ \forall p \in I,\ w \in W: G_{pw} = 1\ e\ intoleravel_w = 1$

- Agendamento Consistente:

  $m_p \leq HP \times x_p,\ \forall p \in I$

- Restrição de Recursos:

  $\sum_{p \in I} (custo_p \times x_p) \leq C_{mes},\ \forall m \in MESES$

- Ajuste de Custo com Base na Seleção:

  $C_p = (\sum_{d \in DURMAX} custo_{pd}) + 1000 \times (1 - x_p),\ \forall p \in I$

- Garantia de Seleção de pelo Menos um Projeto:

  $\sum_{p \epsilon I} x_p \geq 1$

- Ajuste de Risco com Base na Seleção:

  $\sum_{p \epsilon I} G_{pw} \times x_p \leq R_w, \ \forall w \in W$

- Garantia de Resolução de pelo Menos um Ponto de Atenção:

  $\sum_{p \epsilon I} G_{pw} \geq 1 \ , \forall w \in W$

Observação: A utilização de variáveis de decisão, é temporária, sendo que um dos objetivos que nós temos é extingui-las na versão final da modelagem.

Até o momento, foram realizados experimentos de maneira gradativamente complexas, em outras palavras, as restrições estão sendo modeladas para PLI, implementadas em MathProg uma a uma, após, é realizado os experimentos.

O código da instância atual que estamos utilizando está dividido em 4 partes fundamentais:

1 - Declaração de Conjuntos, Parâmetros e variáveis.

2 - Definição da função objetivo.

3 - Definição das restrições.

4 - Exibição do resultado.

Será mostrada várias capturas de tela da situação do código atual utilizado na nossa instância atual.

```
 # Modelo de Programação Linear Inteira para o Problema de Seleção de Projetos
 # com Prazos e Orçamento Anual


 # Declaração dos conjuntos
set I;  # Conjunto de projetos
set W;  # Conjunto de pontos de atenção
set MESES;  # Conjunto de anos no horizonte de planejamento
set DURMAX;  # Conjunto de durações máximas de projetos
 # Parâmetros
param R_w{w in W};  # Risco associado ao ponto de atenção w
param G_w{p in I,w in W} binary;  # Projetos que resolvem o ponto de atenção w (matriz binária)
param d_p{p in I};  # Duração em meses de um projeto p
param custo_p{p in I, d in DURMAX};  # Custo de cada projeto p
param C_mes{m in MESES};  # Orçamento anual
param deadline_w{w in W};  # Prazo limite para resolver um ponto de atenção w
param intoleravel_w{w in W} binary;  # Indica se o ponto de atenção w é intolerável
param HP:= 24;  # Horizonte de Planejamento

 # Variáveis de Decisão
var x_p{p in I} binary;  # 1 se o projeto p é selecionado, 0 caso contrário
var m_p{p in I} >= 0, <= HP integer;  # Mês de início do projeto p
```

```
var c_p{p in I} >= 0;  # Custo do projeto p



 # Função Objetivo
minimize Z: sum{p in I} (c_p[p]);  # Minimizar o custo


 # Restrições

 # Prazo Limite de Resolução de um Ponto de Atenção intolerável
s.t.  DeadlineResolution{  p  in  I,  w  in  W}:  m_p[p]  +  d_p[p]  <=
deadline_w[w]; # para cada p em i e w rm i, dado que g,w seja 1 seja um
ponto intoleravel

 # Agendamento Consistente
s.t. ConsistentScheculing{p in I}: m_p[p] <= HP * x_p[p]; # para cada p
em i, se x_p[p] = 1, m_p[p] + d_p[p] = 1, senão, m_p[p] + d_p[p] = 0

 # Restrição de Recursos
s.t.  ResourceConstraint{m  in  MESES,  d  in  DURMAX}:     sum{p  in
I}(custo_p[p,d] * x_p[p]) <= C_mes[m]; # somatorio dos meses menor que
o custo anual

 # Restrição para ajustar o custo com base na seleção
s.t.  AdjustCost{p  in  I}:  c_p[p]  =  (sum{d  in  DURMAX}  custo_p[p,d])  +
1000 * (1 - x_p[p]);

 # Restrição para garantir que pelo menos um projeto seja selecionado
s.t. AdjustVariable: (sum{p in I} x_p[p]) >= 1;

 # Restrição para ajustar o risco com base na seleção
s.t. AdjustRisk{w in W}: (sum{p in I} G_w[p,w] * x_p[p]) <= R_w[w];

# Restrição para garantir que pelo menos um projeto resolva um ponto de
atenção
s.t. AdjustRisk2{w in W}: sum{p in I} G_w[p,w] >= 1;

# Restrição para ajustar o agendamento
s.t. AdjustScheduling{p in I}: m_p[p] >= x_p[p];

# Restrição para ajustar os custos
s.t.  AdjustCusts:  (sum{m  in  MESES}  C_mes[m])  -  (sum{p  in  I,  d  in
```

```
DURMAX}custo_p[p,d]*x_p[p])  <=  (sum{m  in  MESES}  C_mes[m])/2;  #custo
total durante o HP

 # Resolução do problema
solve;

 # Exibição dos resultados
printf "Projetos selecionados e seus meses de inicio:\n";
for {p in I: x_p[p] = 1} {
    printf "Projeto %s, Mes de Inicio: %d\n", p, m_p[p];
}

end;

 #
```

Observação:

Para utilização do código demonstrado acima, é necessário a instância do problema e de projetos, a instância utilizada foi a seguinte:

1. Conjuntos utilizados:

```
data;

set I := p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16 p17 p18 p19 p20;  #conjunto de projetos

set W := w1 w2 w3;  # Conjunto de pontos de atenção

set MESES := m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 m13 m14 m15 m16 m17 m18 m19 m20 m21 m22 m23 m24;  #qtd mes hp

set DURMAX := d1 d2 d3 d4 d5 d6 d7;
```

2. Parâmetros de pontos de atenção e riscos associados:

```
param R_w :=  # Risco associado ao ponto de atenção w
w1 6
w2 4
w3 5;

param G_w : w1 w2 w3 :=
        p1  0  1  0
        p2  0  0  0
        p3  0  1  0
        p4  1  0  1
        p5  1  0  0
        p6  0  0  0
        p7  0  0  1
        p8  1  0  0
        p9  0  0  1
        p10 0  1  0
        p11  0  1  0
        p12  0  0  0
        p13  0  1  0
        p14  1  0  1
        p15  1  0  0
        p16  0  0  0
        p17  0  0  1
        p18  1  0  0
        p19  0  0  1
        p20  0  1  0;


```

3. Parâmetros de duração e custo de cada projeto:

```
param d_p :=
p1 3
p2 4
p3 2
p4 4
p5 7
p6 4
p7 3
p8 4
p9 2
p10 5
p11 3
p12 4
p13 2
p14 5
p15 7
p16 4
p17 3
p18 4
p19 2
p20 5;

param custo_p :    d1 d2 d3 d4 d5 d6 d7 :=
                p1 10 10 10  0  0  0  0
                p2 12 12 12 12  0  0  0
                p3 15 15  0  0  0  0  0
                p4  2  2  2  1  0  0  0
                p5  5  5  5  5  5  5  5
                p6  6  6  6  6  0  0  0
                p7 12 12 12  0  0  0  0
                p8 10 10 10 10  0  0  0
                p9 20 20  0  0  0  0  0
                p10 8  8  8  8  8  0  0
                p11 11 11 11 0  0  0  0
                p12 10 10 10 10 0  0  0
                p13 25 25  0 0  0  0  0
                p14 10 10 10 10 10 0  0
                p15 5  5  5  5  5  5  5
                p16 10 10 10 10 0  0  0
                p17 20 20 20 0  0  0  0
                p18 10 10 10 10 0  0  0
                p19 25 25 0  0  0  0  0
                p20 10 10 10 10 10 0  0;

```

4. Parâmetros associados ao orçamento mensal, deadline de pontos de atenção e definição de quais pontos de atenção são intoleráveis:

```
param C_mes :=
m1 40
m2 40
m3 40
m4 40
m5 40
m6 40
m7 40
m8 40
m9 40
m10 40
m11 40
m12 40
m13 40
m14 40
m15 40
m16 40
m17 40
m18 40
m19 40
m20 40
m21 40
m22 40
m23 40
m24 40; # Orçamento total


param deadline_w :=
w1 24
w2 24
w3 17;

param intoleravel_w :=
w1 0
w2 0
w3 1;


end;

```

Resultados parciais do objetivo principal:

- Instância do PPSSP resolvida com até três restrições, sendo elas:
    1. Agendamento Consistente
    2. Restrição de recursos
    3. Prazo limite para a resolução de um Ponto de atenção

Referências

1. Cleber Mira, Paulo R Viadanna, Maria Angélica Souza, Arnaldo Moura, João Meidanis, Gabriel A Costa Lima, and Renato P Bossolan. Project scheduling optimization in electrical power utilities. Pesquisa Operacional, 35:285–310, 2015.

2. Bruno Yoichi Tanaka. Aplicação dos métodos GRASP e BRKGA para um problema de otimização de portfólios. Universidade Estadual de Mato Grosso do Sul, 11 2018.

3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition. The MIT Press, 3rd edition, 2009.

4. Andrew Makhorin. Glpk (gnu linear programming kit), 04 2023.

5. Pokutta, Sebastian. "The GNU Linear Programming Kit (GLPK): Resources, Tutorials etc." Sebastian Pokutta's Blog, 24 Jan. 2010.

Por ser a expressão da verdade, firmo a presente declaração ficando responsável pela veracidade das informações contidas neste relatório e ciência do conteúdo da Resolução CEPE-UEMS Nº 2.551 de 16 de dezembro de 2022.

Dourados - MS, 10 de março de 2024.

gov.br Documento assinado digitalmente
NICOLAS HERCULANO PIRES
Data: 19/03/2024 16:07:08-0300
Verifique em https://validar.iti.gov.br

Assinatura do Bolsista

gov.br Documento assinado digitalmente
CLEBER VALGAS GOMES MIRA
Data: 19/03/2024 15:39:40-0300
Verifique em https://validar.iti.gov.br

Assinatura do Orientador

A falta do envio, em prazo estabelecido pela Divisão de Pesquisa, resulta na inadimplência do orientador e orientando.

OBS. Enviar somente em via digital para o e-mail iniciacaocientifica@uems.br