Atividade 4

Giovanni Machado Quintella Gama – RA: 247122
Marcelo De Souza Corumba De Campos – RA: 236730
Everton Romanzini Colombo – RA: 257234

MO824/MC859 - 2º Semestre 2025

1 Introdução

Esta atividade tem como objetivo de explorar *Algoritmos Genéticos* (AG) para a resolução do problema *Maximum Quadratic Binary Function* com restrições de cobertura (MAX-SC-QBF).

Um AG é uma metaheurística populacional que evolui um conjunto de soluções candidatas (indivíduos) ao longo de gerações. Cada indivíduo codifica uma solução para o problema e sua qualidade é medida por uma função de aptidão (*fitness*). Em cada iteração, indivíduos mais aptos são preferencialmente selecionados para reprodução, combinados por recombinação (crossover) e perturbados por mutação, gerando descendentes. Um esquema de substituição (com elitismo) atualiza a população e o processo se repete até um critério de parada (tempo, número de gerações ou estagnação).

Para reforçar a diversidade e reduzir a convergência prematura, adotamos duas estratégias evolutivas alternativas: *Latin Hypercube Sampling* (LHS) e *mutação adaptativa*. O LHS produz uma população inicial mais bem distribuída ao estratificar cada dimensão do espaço de busca (antes de binarizar/ajustar a factibilidade, quando necessário), evitando concentrações artificiais. Já a mutação adaptativa ajusta dinamicamente a taxa de mutação — aumentando-a quando a diversidade cai ou a melhoria estagna e reduzindo-a quando o progresso é consistente — para equilibrar exploração e intensificação.

2 Distribuição das tarefas

Everton Romanzini Colombo: Tradução do framework inicial para python. Implementação do GA padrão para scqbf. Implementação da estratégia alternativa Latin Hypercube. Revisão da estratégia alternativa Adaptive Mutation. Montagem do notebook de experimentos. Revisão do texto. Reorganização das tabelas.

Giovanni Machado Quintella Gama: Elaboração da maior parte do relatório: descrição do problema (MAX-SC-QBF e sua linearização), detalhamento da metodologia, organização das tabelas/figuras e análise crítica dos resultados. Participação das discussões e contribuição para as escolhas da metodologia.

Marcelo De Souza Corumba De Campos: Revisão da implementação do GA padrão e da implementação da estratégia alternativa Latin Hypercube. Implementação da estratégia alternativa Adaptive Mutation e execução dos experimentos e coleta dos dados.

3 Descrição do problema

Seja $N = \{1, \ldots, n\}$ um conjunto de variáveis e $S = \{S_1, \ldots, S_n\}$ uma coleção de subconjuntos $S_i \subseteq N$ que define as variáveis cobertas pelo subconjunto i. Associamos a cada subconjunto i uma variável de decisão $x_i \in \{0, 1\}$ indicando sua seleção. Para cada par (i, j), tem-se um coeficiente $a_{ij} \in \mathbb{R}$ que indica o ganho (positivo ou negativo) obtido quando i e j são selecionados simultaneamente. Juntos, esses

coeficientes formam uma matriz $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ que pondera as interações entre pares de subconjuntos. A função objetivo é uma *Quadratic Binary Function* (QBF), definida a seguir:

$$f(x) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\, x_i\, x_j, \tag{1}$$

podendo-se assumir A simétrica ou, alternativamente, somar apenas sobre $i \le j$ para evitar dupla contagem.

No `MAX-SC-QBF`, busca-se escolher subconjuntos que (i) cubram todo o universo N e (ii) maximizem o ganho quadrático (1). Em notação de conjuntos, a restrição de cobertura pode ser escrita como

$$\bigcup_{\substack{1\le i\le n\\ x_i=1}} S_i \;=\; N, \tag{2}$$

isto é, cada elemento $k \in N$ deve pertencer a pelo menos um subconjunto selecionado. Podemos formular isso como:

$$\text{maximizar} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\, x_i x_j \tag{3}$$

$$\text{sujeito a} \quad \sum_{i:\, k\in S_i} x_i \;\ge\; 1 \qquad \forall\, k \in N \tag{4}$$

$$x_i \in \{0,1\} \qquad \forall\, i \in N. \tag{5}$$

Quando se deseja linearizar o produto $x_i x_j$, introduzem-se variáveis auxiliares binárias y_{ij} para representar tais interações e impõem-se as restrições clássicas de linearização:

$$\text{maximizar} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\, y_{ij} \tag{6}$$

$$\text{sujeito a} \quad (4)$$

$$y_{ij} \le x_i, \quad y_{ij} \le x_j \qquad \forall\, i,j \in N \tag{7}$$

$$y_{ij} \ge x_i + x_j - 1 \qquad \forall\, i,j \in N \tag{8}$$

$$x_i \in \{0,1\},\;\; y_{ij} \in \{0,1\} \qquad \forall\, i,j \in N. \tag{9}$$

As restrições (7)–(8) garantem $y_{ij} = 1$ se, e somente se, $x_i = x_j = 1$, tornando a formulação linear inteira equivalente ao modelo quadrático original.

4 Metodologia

Foi utilizado um AG binário sobre o modelo do MAX-SC-QBF descrito na Seção 3. Cada indivíduo codifica uma solução por um cromossomo de alelos binários $x = (x_0, x_1, \ldots, x_n) \in \{0,1\}^n$, onde x_i=1 indica que o subconjunto S_i está selecionado. A *decodificação* constrói a solução como o conjunto de índices ativos, isto é, $\mathcal{S}(x) = \{\, i \in \{1,\ldots,n\} \mid x_i = 1 \,\}$. A aptidão é avaliada pela função objetivo quadrática $f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j$. A factibilidade é checada pela cobertura $\bigcup_{i:x_i=1} S_i = N$; caso a viabilidade seja violada, aplica-se um *reparo* que insere iterativamente índices que aumentam a cobertura até satisfazê-la, deste modo garantindo a restrição de *set cover*.

As instâncias foram geradas com três estratégias de geração distintas, que produziram cinco instâncias cada uma. Todas as instâncias geradas compartilham um conjunto de parâmetros comuns: o valor de n pertence a $\{25, 50, 100, 200, 400\}$, a cardinalidade de cada subconjunto S_i varia entre 0 e n, e os coeficientes da função objetivo são números de ponto flutuante entre -10.00 e 10.00. As especificidades de cada gerador definem a estrutura das instâncias: as instâncias gen1 são totalmente aleatórias, isto é , tanto o tamanho dos subconjuntos quanto os coeficientes são randômicos; as gen2 contêm subconjuntos com tamanho de no mínimo 0.8n, o que eleva a probabilidade de que uma solução sem as restrições

de Set Cover já seja uma cobertura válida; por fim, as instâncias gen3 possuem ao menos 80% de seus coeficientes como não-negativos, numa estratégia para aproximar os problemas do caso trivial onde todos os coeficientes são positivos.

O *solver* base baseia-se no framework disponibilizado, ao qual acoplamos componentes específicos. A configuração *padrão* emprega:

(i) representação binária $x \in \{0,1\}^n$;

(ii) seleção por torneio binário;

(iii) recombinação por *two-point crossover*;

(iv) mutação *bit-flip* com número de loci sorteado de Poisson(λ);

(v) atualização geracional com elitismo (o melhor indivíduo da geração anterior substitui o pior descendente, se pior que ele);

(vi) reparo de factibilidade;

Os critérios de parada utilizados foram:

1. tempo - limite de 30 minutos de execução;
2. paciência (estagnação) - $10n$ iterações sem melhora na solução com $n \in \{25, 50, 100, 200, 400\}$ referente ao tamanho da instância do problema.

Para avaliar o efeito de componentes e hiperparâmetros, executamos seis variações controladas, mantendo constantes os demais fatores do pipeline (decodificação, avaliação, reparo e elitismo):

1. **GA padrão** — população P=100, multiplicador de mutação λ=1, inicialização aleatória.
2. **Padrão (população reduzida)** — P=50, λ=1, inicialização aleatória.
3. **Padrão (mutação mais intensa)** — P=100, λ=3, inicialização aleatória.
4. **Latin Hypercube (LHS)** — P=100, λ=1, inicialização por LHS (estratificação binária por gene).
5. **Mutação Adaptativa** — P=100, λ=1 inicial, ajuste dinâmico de λ guiado por diversidade.
6. **LHS (população reduzida)** - P=50, λ=1, inicialização por LHS (estratificação binária por gene).

Nos testes (1)–(3) utilizamos o GA padrão apenas alterando os parâmetros como descrito acima. A população inicial é construída sorteando um número aleatório de 1s por cromossomo e reparando a cobertura quando necessário.

No teste (4), substituímos apenas a inicialização por *Latin Hypercube* (Algoritmo 1), em que, para cada gene g, geramos uma permutação aleatória dos P indivíduos e atribuímos o alelo como $\pi(p) \bmod 2$. Com isso, cada locus recebe aproximadamente $P/2$ zeros e $P/2$ uns de forma estratificada (requer P par), espalhando a população no espaço binário antes do reparo de cobertura.

Algorithm 1 Inicialização por Latin Hypercube (binária)

Input: n (nº de genes), P (tamanho da população; múltiplo de 2)
Output: População de P cromossomos viáveis
Inicialize *pop* com P vetores zero de tamanho n
for $g \leftarrow 1$ **to** n **do**
  Gere uma permutação π de $\{0, \ldots, P-1\}$
  for $p \leftarrow 1$ **to** P **do**
    $allele \leftarrow \pi[p] \bmod 2$
    $pop[p][g] \leftarrow allele$
  end for
end for
for all cromossomo c em *pop* **do**
  $c \leftarrow$ RepairCoverage(c)
end for
return *pop*

No teste (5), a adaptação da taxa de mutação é guiada pela *diversidade populacional* D, calculada como a média, ao longo dos loci, da heterozigosidade $p(1-p)$ normalizada para $[0, 1]$ (Algoritmo 2). A cada 5 iterações, compara-se D com os limiares D_{low}=0,2 e D_{high}=0,5: quando D está **alto** ($D > D_{\text{high}}$), reduzimos λ em passos de 0,25 (menos mutações, mais intensificação); quando D está **baixo** ($D < D_{\text{low}}$), aumentamos λ em passos de 0,25 (mais mutações, mais exploração). Em todas as atualizações, λ é limitado ao intervalo $[max(\lambda_0 - 2, 0.25), \lambda_0 + 2]$ e, entre ajustes, permanece constante. A Figura 1 ilustra esse mecanismo: a curva azul mostra a evolução de D enquanto a curva vermelha mostra λ; observa-se que λ tende a *diminuir* quando a diversidade se eleva e a *aumentar* quando a diversidade cai, conforme especificado no Algoritmo 3.

Algorithm 2 Diversity (normalizada)

Input: População *Pop* de tamanho P; comprimento do cromossomo n
$D \leftarrow 0$
for $\ell \leftarrow 1$ **to** n **do**
  $p \leftarrow \frac{1}{P} \sum_{c \in Pop} \mathbf{1}[\, c[\ell] = 1 \,]$ {fração de 1s no locus ℓ}
  $D \leftarrow D + p(1-p)$
end for
return $D/(0.25 \cdot n)$ {normalização para o intervalo $[0, 1]$}

Algorithm 3 Mutação Adaptativa (ajuste de λ + mutação padrão)

Input: *offspring*, λ (corrente), λ_0 (original), *it*
Params: D_{low}=0.2, D_{high}=0.5, passo 0.25, clamp $\lambda \in [\max(\lambda_0-2, 0.25), \lambda_0+2]$
if *it* mod $5 = 0$ **then**
  $D \leftarrow$ DIVERSITY(*offspring*)
  if $D < D_{\text{low}}$ **then**
    $\lambda \leftarrow \min(\lambda + 0.25,\ \lambda_0 + 2)$
  else if $D > D_{\text{high}}$ **then**
    $\lambda \leftarrow \max(\lambda - 0.25,\ \max(\lambda_0 - 2, 0.25))$
  end if
end if
for all cromossomo c em *offspring* **do**
  $m \leftarrow$ POISSONSAMPLE(λ)
  $m \leftarrow \min(m, n)$
  $L \leftarrow$ RANDOMSUBSET($\{1, \ldots, n\}, m$)
  for all locus $\ell \in L$ **do**
    $c[\ell] \leftarrow 1 - c[\ell]$ {bit-flip}
  end for
  $c \leftarrow$ REPAIRCOVERAGE(c)
end for
return *offspring*, λ

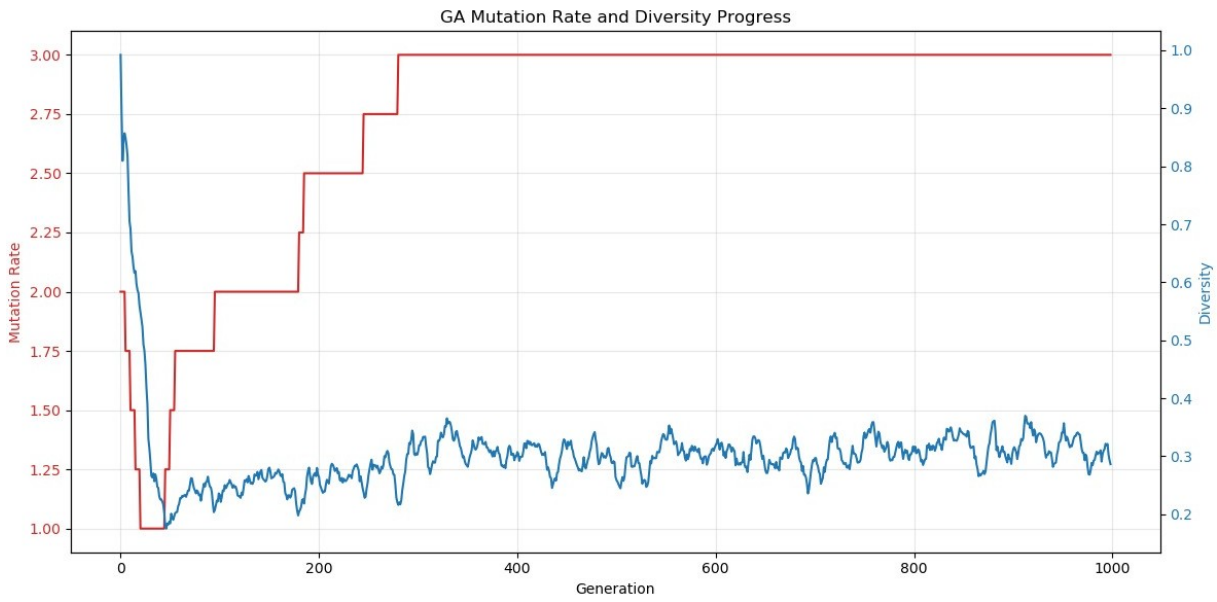

Figura 1: Evolução da taxa de mutação (vermelho, eixo esquerdo) e da diversidade (azul, eixo direito) quando a estratégia *Adaptive Mutation* é utilizada. Observe como a variação da taxa de mutação tende a ser inversamente proporcional à diversidade da população.

5 Resultados

Os dados crus referentes à execução do experimento, assim como todo o código fonte escrito, encontra-se neste repositório do github. Os resultados dos experimentos são mostrados nas tabelas a seguir.

Tabela 1: Valores da função objetivo (maximização), destacando os melhores (azul) e piores (vermelho) resultados.

Instância	n	Gurobi	PADRÃO	PADRÃO +POP	PADRÃO +MUT	PADRÃO +ALT1	PADRÃO +ALT2	Latin + P2
1.1	200	4288.58	4511.01	4614.25	4476.29	4611.47	4349.99	4668.07
1.2	200	4508.92	4668.07	4659.04	4459.87	4614.92	4455.01	4467.85
1.3	50	645.62	645.62	645.62	628.15	645.62	645.62	645.62
1.4	25	261.06	261.06	261.06	261.06	261.06	261.06	261.06
1.5	100	1478.35	1525.48	1522.90	1480.19	1508.25	1489.16	1501.12
2.1	50	575.87	575.87	575.87	575.33	575.87	575.87	575.87
2.2	400	9694.43	14161.95	14334.06	12737.92	14077.94	12471.01	13417.00
2.3	25	210.16	210.16	210.16	206.89	210.16	210.16	210.16
2.4	400	7419.65	13141.63	13267.80	11758.98	13341.82	11644.23	14386.40
2.5	100	1550.49	1515.43	1529.54	1530.02	1550.49	1512.17	1471.07
3.1	400	238058.87	238058.87	238058.87	234543.44	238058.87	236703.50	238058.87
3.2	100	14887.43	14887.43	14887.43	14887.43	14887.43	14887.43	14887.43
3.3	200	59765.97	59765.97	59765.97	57715.50	59765.97	59765.97	59765.97
3.4	25	981.05	981.05	981.05	945.84	981.05	981.05	981.05
3.5	25	766.71	766.71	766.71	766.71	766.71	766.71	766.71
Best count		9	11	10	3	10	8	10
Worst count		4	3	3	9	3	4	4

Tabela 2: Tempos de execução (em segundos) com indicadores de parada (T para timeout e P para paciência excedida) e estatísticas de resumo. Melhores tempos foram marcados em azul, piores tempos em vermelho.

Instância	n	Gurobi	PADRÃO	PADRÃO +POP	PADRÃO +MUT	PADRÃO +ALT1	PADRÃO +ALT2	Latin + P2
1.1	200	600.03^{T}	1800.43^{T}	997.74^{P}	1800.22^{T}	1800.04^{T}	1800.47^{T}	608.56^{P}
1.2	200	600.03^{T}	1800.24^{T}	626.81^{P}	1800.04^{T}	1244.89^{P}	1800.19^{T}	538.20^{P}
1.3	50	7.41^{P}	25.27^{P}	13.09^{P}	45.32^{P}	28.47^{P}	26.81^{P}	12.45^{P}
1.4	25	0.10^{P}	4.07^{P}	1.87^{P}	7.34^{P}	3.46^{P}	3.48^{P}	2.36^{P}
1.5	100	600.01^{T}	237.63^{P}	90.63^{P}	455.04^{P}	165.39^{P}	370.17^{P}	93.73^{P}
2.1	50	7.13^{P}	30.97^{P}	15.39^{P}	96.88^{P}	28.54^{P}	36.73^{P}	15.67^{P}
2.2	400	600.08^{T}	1800.59^{T}	1800.79^{T}	1800.90^{T}	1801.76^{T}	1800.64^{T}	1800.01^{T}
2.3	25	0.10^{P}	4.60^{P}	2.32^{P}	7.82^{P}	4.21^{P}	4.21^{P}	2.62^{P}
2.4	400	600.08^{T}	1801.96^{T}	1801.01^{T}	1802.24^{T}	1802.56^{T}	1800.89^{T}	1800.65^{T}
2.5	100	600.01^{T}	347.85^{P}	113.93^{P}	1040.95^{P}	237.83^{P}	524.80^{P}	114.22^{P}
3.1	400	18.88^{P}	1800.30^{T}	1801.02^{T}	1801.79^{T}	1800.29^{T}	1803.74^{T}	1800.78^{T}
3.2	100	0.22^{P}	265.63^{P}	121.76^{P}	1211.53^{P}	257.80^{P}	348.15^{P}	127.58^{P}
3.3	200	1.93^{P}	1800.51^{T}	867.13^{P}	1800.82^{T}	1800.29^{T}	1800.57^{T}	868.18^{P}
3.4	25	0.01^{P}	5.05^{P}	2.57^{P}	6.44^{P}	4.94^{P}	5.16^{P}	2.90^{P}
3.5	25	0.01^{P}	4.76^{P}	2.26^{P}	6.95^{P}	4.70^{P}	4.96^{P}	2.42^{P}
Tempo médio (s)		241.60	712.32	537.22	985.62	732.35	748.73	452.69
Mediana (s)		7.41	237.63	90.63	1040.95	165.39	370.17	114.22
% Timeout		40.0%	40.0%	26.7%	46.7%	40.0%	46.7%	26.7%
Best count		9	0	2	0	0	0	5
Worst count		1	3	0	9	2	1	0

(1) Padrão (baseline). Em instâncias pequenas ($n \in \{25, 50\}$), observa-se o caso frequente em que a função objetivo atinge o mesmo valor de referência, com variações no tempo de execução, apenas; o *baseline* é competitivo em qualidade e tempo. Em instâncias maiores, o padrão ainda apresenta bons valores (e.g., melhor valor em 1.5 e 1.2), mas frequentemente atinge o limite de tempo, limitando ganhos adicionais.

(2) Padrão com população reduzida (P2). Reduzir a população para P=50 reduz drasticamente o tempo de execução na maioria dos casos (1.4, 2.3, 3.4, 3.5, 1.3, 2.1, 1.5, 2.5, 3.2), com pouca perda de qualidade; em algumas instâncias grandes houve até *melhora* (e.g., 2.2). Em cenários limitados por tempo, o P2 percorre mais gerações, o que explica os bons compromissos entre custo e desempenho (e.g., 1.1, tempo 997s com valor superior ao padrão).

(3) Padrão com mutação mais intensa (M2). Aumentar a taxa para $\lambda = 3$ piorou a qualidade em várias instâncias (2.3, 3.4, 1.2, 1.1, 3.3, 2.2, 2.4, 3.1) e elevou consideravelmente o tempo (chegando a 1041s em 2.5 e a 1800s nas maiores). O excesso de perturbação rompe blocos promissores e aciona o reparo com maior frequência, encarecendo a busca; é a configuração com pior relação qualidade/tempo.

(4) Latin Hypercube (LHS). A inicialização estratificada manteve desempenho semelhante ao padrão nos casos pequenos, e trouxe ganhos pontuais de qualidade em problemas médios (e.g., melhor valor em 2.5), com tempo intermediário. Em instâncias grandes, a qualidade é competitiva, mas a observação do ganho de tempo é limitada quando o processo esbarra no limite.

(5) Mutação Adaptativa (Adaptive). Esta configuração não trouxe ganhos sistemáticos de qualidade e, em várias instâncias grandes, ficou abaixo das alternativas (e.g., 1.1, 2.2, 2.4), além de consumir mais tempo — possivelmente por maior sobrecusto com reparo após mutações mais frequentes nos períodos de baixa diversidade.

(6) Latin Hypercube com população reduzida (Latin + P2). É a combinação com melhor *custo-benefício* geral: costuma ser a mais rápida (ou próxima disso) e atingiu os melhores valores em diversas instâncias grandes (e.g., 1.1 e 2.4), mantendo competitividade nas demais. O LHS distribui melhor a população inicial e o P2 acelera o ciclo reprodutivo, permitindo explorar e intensificar mais gerações dentro do mesmo orçamento de tempo.

6 Conclusão

Este trabalho investigou o uso de *Algoritmos Genéticos* (AG) para resolver o MAX-SC-QBF, explorando dois mecanismos voltados à diversidade: (i) inicialização por *Latin Hypercube Sampling* (LHS) e (ii) mutação adaptativa guiada por heterozigosidade. Mantivemos um *pipeline* comum (avaliação quadrática, reparo de cobertura e elitismo) e variamos apenas componentes/hiperparâmetros relevantes ao equilíbrio exploração–intensificação.

De maneira geral, os resltados indicam que injetar diversidade na inicialização é mais eficaz do que aumentar agressivamente a mutação ao longo da busca. Em um regime limitado por tempo, configurações que aceleram o ciclo evolutivo e começam de um conjunto bem espalhado de soluções (caso do LHS com população reduzida) tendem a oferecer o melhor custo-benefício: percorrem mais gerações, preservam blocos promissores e reduzem o acionamento oneroso do reparo. Por outro lado, intensificar a mutação sem controle rompe estruturas úteis e eleva a frequência de soluções inviáveis, deslocando esforço para o reparo sem ganhos consistentes na qualidade. A mutação adaptativa, tal como formulada aqui, não trouxe melhorias sistemáticas: o esquema de ajuste por limiares de diversidade mostrou-se pouco responsivo e, em momentos de baixa diversidade, induziu sobrecarga adicional de reparo.

Em resumo, a combinação de inicialização LHS e população reduzida mostrou-se uma linha promissora para o MAX-SC-QBF nas condições avaliadas, ao passo que mutação excessiva e adaptação por limiares simples tendem a desperdiçar orçamento de tempo com factibilização.