# MO824 - Relatório para atividade 3

Everton Romanzini Colombo e257234@dac.unicamp.br

Paulo Victor de Souza Santos p248438@dac.unicamp.br

Pedro Paulo Gomes do Carmo p299206@dac.unicamp.br

Setembro 2025

## 1 Introdução

O Tabu Search (TS) é uma meta-heurística amplamente empregada na resolução de problemas de otimização combinatória, que aprimora o Local Search (busca local). Seu princípio fundamental consiste em evitar o retorno a soluções já exploradas recentemente, impedindo a ocorrência de ciclos e direcionando a busca para novas áreas do espaço de soluções. Essa abordagem, que se baseia em uma lista "tabu"de movimentos proibidos, permite que o algoritmo escape de ótimos locais e explore de forma mais eficiente o espaço de busca, buscando uma solução globalmente melhor.

Neste trabalho, o Tabu Search foi aplicado especificamente para resolver o problema de otimização MAX-SC-QBF. Código e instâncias encontram-se no seguinte repositório: (Link do Github))

## 2 Descrição do Problema

Seja $N = \{1, ..., n\}$ o conjunto de variáveis da QBF. Seja $\mathcal{S} = \{S_1, ..., S_n\}$ uma coleção de subconjuntos, onde $S_i \subseteq N$ representa o conjunto de variáveis cobertas pelo subconjunto $i$. Cada subconjunto $i$ está associado a uma variável de decisão binária $x_i \in \{0, 1\}$, que indica se o subconjunto é selecionado ($x_i = 1$) ou não ($x_i = 0$).

O objetivo é selecionar um subconjunto de $\mathcal{S}$ de forma a:

1. Maximizar o ganho quadrático total derivado das interações entre os subconjuntos selecionados, onde $a_{ij} \in \mathbb{R}$ é o coeficiente que representa o ganho por selecionar os subconjuntos $i$ e $j$ simultaneamente.

2. Garantir que todas as variáveis do conjunto $N$ sejam cobertas por, pelo menos, um dos subconjuntos selecionados.

O problema MAX-SC-QBF pode ser formalmente expresso como:
**Variáveis de Decisão:** $x_i \in \{0, 1\}, \quad \forall i \in \{1, ..., n\}$

**Função Objetivo:**

$$\text{Maximizar} \quad f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot x_i \cdot x_j$$

**Restrições:** Para garantir que cada variável $k \in N$ seja coberta por pelo menos um subconjunto selecionado, a seguinte restrição deve ser satisfeita:

$$\sum_{i:k \in S_i} x_i \geq 1, \quad \forall k \in N$$

# 3 Metodologia

## 3.1 Heurística construtiva

Durante cada iteração, o algoritmo avalia os elementos que ainda não estão incluídos na solução parcial. O objetivo é identificar o subconjunto de elementos candidatos que, ao serem adicionados, promovem o avanço em direção a uma solução viável, especificamente melhorando a cobertura da solução. Em seguida, um elemento é selecionado aleatoriamente a partir desse conjunto de candidatos. O processo é repetido até que uma solução viável seja construída. A abordagem, neste caso, não se baseia em um critério guloso de maximização de ganho imediato, mas sim no atendimento da restrição de cobertura.

## 3.2 Métodos de Busca

Neste estudo, foram implementados dois métodos de busca local: First Improving e Best Improving. A abordagem inicial de ambos os algoritmos é análoga: a partir dos elementos não incluídos na solução atual, é construído um conjunto de operações de vizinhança candidatas, que incluem a inserção de um elemento na solução, a remoção de um elemento da solução e a troca de um elemento da solução por outro que está fora dela. A ordem de aplicação dessas três operações é aleatoriamente permutada para evitar vieses nos resultados, definindo, assim, o espaço de vizinhanças a ser explorado em cada iteração.

Às operações de vizinhança está relacionado um valor de delta função objetivo, que quantifica a melhora ou piora potencial na função objetivo da solução. A decisão de aceitar ou rejeitar uma movimentação é baseada nesse valor. Adicionalmente, o processo de busca incorpora elementos da Busca Tabu. A lista tabu é utilizada para proibir a inclusão/remoção ou qualquer mudança de elementos que foram recentemente modificados, evitando ciclos na busca. Ambas as estratégias, First Improving e Best Improving, verificam se o elemento a ser movimentado está presente na lista tabu. Uma exceção a essa restrição, conhecida como critério de aspiração, é aplicada quando a movimentação resulta em uma solução que supera a melhor solução global encontrada até o momento. Neste caso, a operação é aceita independentemente de o elemento estar na lista tabu.

A principal distinção entre os dois métodos reside na forma como a movimentação a ser executada é selecionada. No First Improving, a primeira operação que resulta em uma melhoria na solução atual é selecionada e executada. Em contraste, o Best Improving avalia todas as movimentações possíveis no espaço de vizinhança e seleciona aquela que oferece o maior ganho (delta).

## 3.3 Busca Tabu

Nos algoritmos de busca, a lista tabu foi implementada com o uso de uma deque (fila de duas pontas) de tamanho fixo, que é definido pelo parâmetro tenure. O mecanismo de atualização da lista tabu é o seguinte: a cada movimento selecionado e executado, os elementos referentes a esse movimento são adicionados ao final da deque. Devido ao seu tamanho fixo, quando a deque atinge sua capacidade máxima, o elemento mais antigo (o do início) é automaticamente removido.

Essa abordagem impede temporariamente que a busca retorne a estados recentemente explorados, pois os movimentos que levariam a esses estados envolveriam elementos que estariam na lista tabu, sendo, portanto, bloqueados. Após um número suficiente de iterações, os movimentos mais antigos são removidos da lista tabu, tornando-os novamente disponíveis para movimentos de vizinhança.

## 3.4 Estratégias Tabu alternativas

Uma das estratégias alternativas implementadas foi o Tabu Search (TS) Probabilístico. Diferentemente da abordagem tradicional, que pode potencialmente avaliar o conjunto inteiro da vizinhança, o TS Probabilístico restringe a busca a uma amostra aleatória dessa vizinhança. A intensidade desse mecanismo é controlada por um parâmetro $p$ que representa a proporção do conjunto de vizinhanças que será amostrada a cada passo da busca. Em nossos experimentos, fixamos $p = 0.6$, pois queríamos que a mudança no tempo de execução das instâncias fosse significante o suficiente a ponto de não poder ser atribuída à variância normal das medidas.

A outra estratégia alternativa implementada foi a Intensificação por Reinício. Após um número fixo de iterações sem melhoria na melhor solução encontrada (parâmetro que chamamos de *Restart Patience*, o qual fixamos em 100), a busca retorna para a melhor solução encontrada até o momento, mas com a restrição adicional de que os elementos das novas soluções deverão obrigatoriamente conter certos elementos fixados. O critério que estabelece quais elementos serão fixados é o número mais recente de vezes consecutivas que um elemento apareceu em uma "melhor solução". A Figura 1 mostra essa estratégia em funcionamento; note como, após um certo número de iterações sem melhora na melhor solução, o valor da solução atual é redefinido para o maior conhecido.

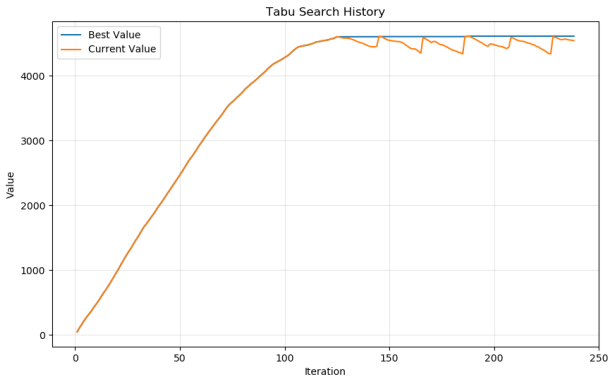


Figura 1: Ilustração da estratégia *Intensification by Restart* em funcionamento.

## 3.5 Ambiente de execução

| | | | |
|---|---|---|---|
| **SO** | Ubuntu 22.04.5 LTS (64 bits) | **Processador** | Intel Core i5-13450HX |
| **Threads (fís./lóg.)** | 16 / 16 | **Linguagem** | Python 3.12 |
| **Tempo-limite por instância** | 1800 s (30 min) | **Critério adicional** | Paciência = 2500 iterações sem melhoria |

## 3.6 Configuração Experimental

A série de experimentos foi planejada para investigar o impacto de diferentes configurações e parâmetros na performance do algoritmo Tabu Search. Para garantir uma análise controlada, um cenário-base foi estabelecido, com modificações específicas aplicadas em cada teste. As tabelas com os resultados se encontram na seção abaixo.

Os parâmetros para os experimentos são descritos a seguir. O limite de tempo escolhido foi de 30 minutos, e a paciência (definida como o número de iterações sem melhoria na solução) foi estabelecida em 2500.

1. **Cenário Padrão:** Busca Tabu com método de busca *First Improving*, *tabu tenure* igual a $T_1 = 8$, e estratégia tabu padrão.
2. **Cenário Padrão + Best:** Cenário Padrão, mas com método de busca *Best Improving*.
3. **Cenário Padrão + Tenure:** Cenário Padrão, mas com um novo valor de *tabu tenure* igual a $T_2 = 25$.
4. **Cenário Padrão + Método 1:** Cenário Padrão, mas com Tabu Search (TS) Probabilístico (com parâmetro de proporção $p = 0.6$).
5. **Cenário Padrão + Método 2:** Cenário Padrão, mas com Intensification by Restart.

As instâncias foram geradas em e extraídas do seguinte repositório: A1.

# 4 Resultados

## Resumo comparativo das configurações

Tabela 1: Resumo comparativo das configurações (tempos em segundos).

| Configuração | Tempo médio | Mediana | Timeout (%) |
|---|---|---|---|
| 1. Padrão (first, T1=8) | 293.13 | 16.81 | 13.3% |
| 2. Padrão+Best | 319.04 | 33.65 | 13.3% |
| 3. Padrão+Tenure (first, T2=25) | 294.91 | 16.87 | 13.3% |
| 4. Padrão+Method1 (Probabilistic) | 284.29 | 21.03 | 13.3% |
| 5. Padrão+Method2 (Restart) | 306.55 | 21.30 | 13.3% |

A tabela em questão apresenta as médias dos tempos de execução para cada configuração testada. Uma análise dos resultados revela padrões consistentes e, em grande parte, previsíveis.

A configuração 4, que combina o método de busca First Improvement com o TS Probabilístico, demonstrou o menor tempo médio de execução. Esse resultado é esperado, pois o First Improvement é intrinsecamente mais rápido, e o TS Probabilístico, ao restringir o espaço de busca, contribui para uma convergência mais ágil. Em segundo lugar, a configuração 1, que utiliza apenas o First Improvement, corrobora essa observação.

A variação no parâmetro de Tenure parece ter um impacto mínimo no tempo de execução, como evidenciado pela comparação entre as configurações 1 e 3, que mantiveram as demais condições idênticas.

Por fim, é relevante notar que as ocorrências de timeout não foram influenciadas pelas configurações, mas sim por características específicas de certas instâncias. Isso sugere que, para nosso caso, os timeouts estão mais ligados à complexidade das instâncias e a do método de busca.

# 5 Análises

Uma primeira observação relevante é que as instâncias da geração 3 (de 3.1 a 3.5) parecem ser mais simples, uma vez que, em todas as configurações testadas, os valores da função objetivo permaneceram os mesmos. A única variação significativa foi o tempo de execução em cada configuração.

Ao realizar uma comparação geral entre a Configuração 1 e a Configuração 2, nota-se que apenas as instâncias 1.5 e 2.2 apresentaram um valor inferior na função objetivo utilizando a Configuração 2. Esse resultado é inesperado, considerando a diferença de implementação entre o First Improving e o Best Improving. Como o Best Improving avalia toda a vizinhança, espera-se que ele obtenha, na maioria dos casos, um valor superior na função objetivo. A ocorrência de valores inferiores nessas duas instâncias pode ser atribuída ao fato de a busca na Configuração 2 ter convergido para um mínimo local, impossibilitando sua saída; possivelmente, um valor maior de *tenure* melhoraria o desempenho para essas instâncias.

A comparação entre os resultados da Configuração 1 e da Configuração 3 revela um cenário misto. Para as instâncias 1.1, 1.3 e 2.4, a Configuração 1, com o valor de Tenure igual a 8, alcançou um valor maior na função objetivo. As demais instâncias, com exceção da geração 3, obtiveram um desempenho superior na Configuração 3. A princípio, essa variabilidade não parece estar correlacionada ao tamanho "n"das instâncias, visto que os tamanhos são diversos. Da mesma forma, não há indícios de que a forma como as instâncias foram geradas influencie o resultado, pois instâncias tanto da geração 1 quanto da geração 2 apresentaram melhor desempenho em relação à função objetivo em ambas as configurações. Esse resultado sugere que não há um padrão definido para o valor ideal de Tenure, indicando que a escolha depende de cada problema específico e que não existe uma regra clara de que o valor de Tenure deva aumentar/diminuir proporcionalmente ao tamanho de "n". Possivelmente, para otimizar a solução de uma instância específica, seria necessário testar múltiplos valores de Tenure para identificar a configuração mais eficaz.

A análise comparativa entre as configurações 1 e 4 revela padrões semelhantes aos observados na comparação entre as configurações 1 e 2. O desempenho da configuração 1 foi superior em apenas duas instâncias (1.3 e 1.5), apresentando um valor da função objetivo mais alto. Nas demais instâncias, os valores foram iguais ou inferiores à configuração 4. O TS Probabilístico, empregado na configuração 4, restringe a vizinhança de busca e introduz um elemento de aleatoriedade no início de cada iteração. Os resultados sugerem que essa abordagem, em média, resulta em um desempenho superior, provavelmente devido à capacidade da aleatoriedade de

escapar de ótimos locais.

As duas instâncias nas quais a configuração 4 teve um desempenho inferior podem ser atribuídas a fatores aleatórios ou a características específicas dessas instâncias. A natureza da geração dessas instâncias pode ter criado uma topologia de busca na qual a restrição da vizinhança foi prejudicial, levando a um resultado sub-ótimo. No entanto, a pequena discrepância nos valores não permite uma conclusão definitiva.

Por fim, ao comparar a Configuração 1 com a Configuração 5, que utiliza a Intensificação por Reinício (Intensification by Restart), os resultados também foram variados. As instâncias 1.2, 1.3, 1.5, 2.2 e 2.4 alcançaram um valor maior na função objetivo na Configuração 1, que emprega somente o First Improving, em comparação com a Configuração 5. Novamente, essa diferença não parece estar ligada ao tamanho "n"ou à maneira como a instância foi gerada. O padrão observado foi que, em todas as instâncias, o tempo de execução até a condição de parada foi maior na Configuração 5 do que na Configuração 1, o que é esperado devido à implementação da Intensificação por Reinício.

# 6 Tabelas

## Configuração 1 – PADRÃO (first-improving, $T_1 = 8$)

Tabela 2: Resultados da Configuração 1 (PADRÃO)

| instance | $n$ | stop_reason | best_objective | coverage | time_taken (s) |
|---|---|---|---|---|---|
| 1.1 | 200 | `patience exceeded` | 4531.46 | 1.00 | 305.5462 |
| 1.2 | 200 | `patience exceeded` | 4617.21 | 1.00 | 300.9435 |
| 1.3 | 50 | `patience exceeded` | 642.03 | 1.00 | 5.6507 |
| 1.4 | 25 | `patience exceeded` | 243.69 | 1.00 | 0.9056 |
| 1.5 | 100 | `patience exceeded` | 1525.48 | 1.00 | 39.2314 |
| 2.1 | 50 | `patience exceeded` | 562.74 | 1.00 | 5.2707 |
| 2.2 | 400 | `time limit` | 14401.08 | 1.00 | 1800.1219 |
| 2.3 | 25 | `patience exceeded` | 210.16 | 1.00 | 0.8737 |
| 2.4 | 400 | `time limit` | 13440.01 | 1.00 | 1800.8856 |
| 2.5 | 100 | `patience exceeded` | 1506.45 | 1.00 | 41.3563 |
| 3.1 | 400 | `patience exceeded` | 238058.87 | 1.00 | 74.6740 |
| 3.2 | 100 | `patience exceeded` | 14887.43 | 1.00 | 4.1157 |
| 3.3 | 200 | `patience exceeded` | 59765.97 | 1.00 | 16.8105 |
| 3.4 | 25 | `patience exceeded` | 981.05 | 1.00 | 0.2825 |
| 3.5 | 25 | `patience exceeded` | 766.71 | 1.00 | 0.2837 |

## Configuração 2 – PADRÃO+BEST (best-improving, $T_1 = 8$)

Tabela 3: Resultados da Configuração 2 (PADRÃO+BEST)

| instance | $n$ | stop_reason | best_objective | coverage | time_taken (s) |
|---|---|---|---|---|---|
| 1.1 | 200 | patience exceeded | 4611.47 | 1.00 | 335.5740 |
| 1.2 | 200 | patience exceeded | 4668.07 | 1.00 | 337.5548 |
| 1.3 | 50 | patience exceeded | 645.62 | 1.00 | 5.6411 |
| 1.4 | 25 | patience exceeded | 261.06 | 1.00 | 0.7579 |
| 1.5 | 100 | patience exceeded | 1522.90 | 1.00 | 43.1742 |
| 2.1 | 50 | patience exceeded | 575.87 | 1.00 | 5.8086 |
| 2.2 | 400 | time limit | 14385.97 | 1.00 | 1801.3169 |
| 2.3 | 25 | patience exceeded | 210.16 | 1.00 | 0.9065 |
| 2.4 | 400 | time limit | 13530.16 | 1.00 | 1800.9073 |
| 2.5 | 100 | patience exceeded | 1550.49 | 1.00 | 45.1624 |
| 3.1 | 400 | patience exceeded | 238058.87 | 1.00 | 369.4454 |
| 3.2 | 100 | patience exceeded | 14887.43 | 1.00 | 5.1266 |
| 3.3 | 200 | patience exceeded | 59765.97 | 1.00 | 33.6477 |
| 3.4 | 25 | patience exceeded | 981.05 | 1.00 | 0.2776 |
| 3.5 | 25 | patience exceeded | 766.71 | 1.00 | 0.2748 |

## Configuração 3 – PADRÃO+TENURE (first-improving, $T_2 = 25$)

Tabela 4: Resultados da Configuração 3 (PADRÃO+TENURE)

| instance | $n$ | stop_reason | best_objective | coverage | time_taken (s) |
|---|---|---|---|---|---|
| 1.1 | 200 | patience exceeded | 4485.01 | 1.00 | 313.2225 |
| 1.2 | 200 | patience exceeded | 4627.74 | 1.00 | 319.6638 |
| 1.3 | 50 | patience exceeded | 600.73 | 1.00 | 5.7711 |
| 1.4 | 25 | patience exceeded | 243.69 | 1.00 | 0.9231 |
| 1.5 | 100 | patience exceeded | 1525.48 | 1.00 | 40.0916 |
| 2.1 | 50 | patience exceeded | 575.87 | 1.00 | 5.7310 |
| 2.2 | 400 | time limit | 14427.58 | 1.00 | 1800.9490 |
| 2.3 | 25 | patience exceeded | 210.16 | 1.00 | 0.8881 |
| 2.4 | 400 | time limit | 13350.38 | 1.00 | 1801.1058 |
| 2.5 | 100 | patience exceeded | 1512.59 | 1.00 | 39.2582 |
| 3.1 | 400 | patience exceeded | 238058.87 | 1.00 | 74.6539 |
| 3.2 | 100 | patience exceeded | 14887.43 | 1.00 | 3.9998 |
| 3.3 | 200 | patience exceeded | 59765.97 | 1.00 | 16.8699 |
| 3.4 | 25 | patience exceeded | 981.05 | 1.00 | 0.2862 |
| 3.5 | 25 | patience exceeded | 766.71 | 1.00 | 0.2877 |

### Configuração 4 – PADRÃO+METHOD1 (Probabilistic TS)

Tabela 5: Resultados da Configuração 4 (PADRÃO+METHOD1)

| instance | $n$ | stop_reason | best_objective | coverage | time_taken (s) |
|---|---|---|---|---|---|
| 1.1 | 200 | patience exceeded | 4609.88 | 1.00 | 240.5466 |
| 1.2 | 200 | patience exceeded | 4659.04 | 1.00 | 233.1851 |
| 1.3 | 50 | patience exceeded | 591.63 | 1.00 | 4.4512 |
| 1.4 | 25 | patience exceeded | 261.06 | 1.00 | 0.7292 |
| 1.5 | 100 | patience exceeded | 1501.12 | 1.00 | 30.9627 |
| 2.1 | 50 | patience exceeded | 562.74 | 1.00 | 3.9428 |
| 2.2 | 400 | time limit | 14423.10 | 1.00 | 1800.1235 |
| 2.3 | 25 | patience exceeded | 210.16 | 1.00 | 0.7451 |
| 2.4 | 400 | time limit | 13515.62 | 1.00 | 1800.2344 |
| 2.5 | 100 | patience exceeded | 1515.56 | 1.00 | 30.2014 |
| 3.1 | 400 | patience exceeded | 238058.87 | 1.00 | 92.3947 |
| 3.2 | 100 | patience exceeded | 14887.43 | 1.00 | 5.0509 |
| 3.3 | 200 | patience exceeded | 59765.97 | 1.00 | 21.0294 |
| 3.4 | 25 | patience exceeded | 981.05 | 1.00 | 0.3568 |
| 3.5 | 25 | patience exceeded | 766.71 | 1.00 | 0.3558 |

### Configuração 5 – PADRÃO+METHOD2 (Intensification by Restart)

Tabela 6: Resultados da Configuração 5 (PADRÃO+METHOD2)

| instance | $n$ | stop_reason | best_objective | coverage | time_taken (s) |
|---|---|---|---|---|---|
| 1.1 | 200 | patience exceeded | 4531.99 | 1.00 | 374.9800 |
| 1.2 | 200 | patience exceeded | 4579.23 | 1.00 | 385.1392 |
| 1.3 | 50 | patience exceeded | 631.17 | 1.00 | 7.2450 |
| 1.4 | 25 | patience exceeded | 243.69 | 1.00 | 1.1503 |
| 1.5 | 100 | patience exceeded | 1525.35 | 1.00 | 52.2837 |
| 2.1 | 50 | patience exceeded | 575.87 | 1.00 | 7.2638 |
| 2.2 | 400 | time limit | 14341.66 | 1.00 | 1800.5503 |
| 2.3 | 25 | patience exceeded | 210.16 | 1.00 | 1.1225 |
| 2.4 | 400 | time limit | 13430.93 | 1.00 | 1800.5912 |
| 2.5 | 100 | patience exceeded | 1515.56 | 1.00 | 48.6245 |
| 3.1 | 400 | patience exceeded | 238058.87 | 1.00 | 92.1837 |
| 3.2 | 100 | patience exceeded | 14887.43 | 1.00 | 5.1659 |
| 3.3 | 200 | patience exceeded | 59765.97 | 1.00 | 21.2965 |
| 3.4 | 25 | patience exceeded | 981.05 | 1.00 | 0.3516 |
| 3.5 | 25 | patience exceeded | 766.71 | 1.00 | 0.3497 |

# 7 Contribuições individuais dos autores

**Everton:** Tradução inicial do código para python; implementação do TS tradicional para scqbf; implementação completa da estratégia alternativa Probabilistic TS; implementação da estratégia alternativa Intensification by Restart em conjunto com Pedro. Execução dos experimentos. Revisão do texto.

**Pedro:** Implementação da estratégia alternativa Intensification by Restart em conjunto com Everton. Execução de alguns experimentos de teste e backup. Descrição da estratégia alternativa Intensification by Restart no relatório.

**Paulo:** Introdução e descrição do problema no relatório; Descrição das metodologias utilizadas; Análise dos resultados obtidos em cada um dos experimentos; Comparação dos resultados em diferentes cenários.