Submetido para MO824

Atividade 5 - Grupo 8 - MO824

Eduardo Barros Innarelli - RA 170161
Victor Ferreira Ferrari - RA 187890
Jorge Menezes dos Santos - RA 264962

Resumo. Este trabalho apresenta soluções heurísticas para o problema MAX-QBFPT baseadas na metaheurística Busca Tabu. Foram explorados as estratégias tabus de intensificação por reinicio e *strategic oscillation*, além dos métodos de busca local *first-improving* e *best-improving*. Os métodos implementados foram avaliados através de experimentos computacionais. De acordo com os resultados obtidos, as estratégias mais promissoras foram a intensificação por reinicio e a combinação de intensificação com *strategic oscillation*.

Palavras-chave. Busca Tabu, Otimização, QBFPT, QBF, metaheurística.

1. Introdução

Este trabalho consiste na apresentação de soluções heurísticas, baseadas na metaheurística Busca Tabu, para o problema MAX-QBF com triplas proibidas, proposto na Atividade 5 de MO824 2S-2020. Para isso, foram realizados experimentos testando duas das estratégias tabus apresentados por Gendreau e Potvin, em "Tabu Search" [1].

O Problema MAX-QBF com Triplas Proibidas (MAX-QBFPT) é uma variação do Problema MAX-QBF ("*Maximum Quadractic Binary Function*"), em que se objetiva maximizar uma função binária quadrática, de modo que x_i, x_j e x_k não assumam o valor 1 simultaneamente caso (i, j, k) forme uma tripla proibida.

1.1. Modelo Matemático

Uma QBF é uma função binária quadrática $f : \mathbb{B}^n \to \mathbb{R}$ que pode ser expressa como uma soma de termos quadráticos utilizando variáveis binárias. Sejam $a_{ij} \in \mathbb{R}$ $(i, j = 1, \ldots, n)$ os coeficientes de f, no problema em que estamos interessados, uma QBF é dada por:

$$f(x_1, ..., x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot x_i \cdot x_j \tag{1.1}$$

Sejam x_i as variáveis binárias de uma QBF $f(x_1, \ldots, x_n)$, para $i \in \{1 \ldots n\}$ e $\mathcal{T} = \{(i, j, k) \in \mathbb{N} : 1 \leq i < j < k \leq n\}$ o conjunto de todas as triplas ordenadas, sem repetição, dos naturais de 1 a n. Dado um conjunto de triplas proibidas $T \subseteq \mathcal{T}$,

o problema MAX-QBFPT consiste na maximização de $f(x_1, \ldots, x_n)$, de modo que x_i, x_j e x_k não sejam todos iguais a 1, para cada $(i, j, k) \in T$. Com base nisso, o MAX-QBFPT pode ser formulado como segue:

$$\max Z = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot x_i \cdot x_j$$

$$\text{s.a} \quad x_i + x_j + x_k \leq 2, \qquad \forall (i, j, k) \in T \quad (1.2)$$

$$x_i \in \mathbb{B}. \qquad \forall i \in \{1, ..., n\} \quad (1.3)$$

As restrições (1.2) correspondem às restrições das triplas proibidas, cujas quais podem ser qualquer subconjunto de triplas ordenadas sem repetições. As restrições (1.3) impõem o domínio das variáveis de decisão como sendo binárias. A função objetivo maximiza o valor da QBF.

1.2. Conjunto de Triplas Proibidas

Especificamente para esta atividade, as triplas são geradas a partir de duas funções $g, h : [1, n] \rightarrow [1, n]$, que são definidas do seguinte modo:

$$l_1(u) = 1 + ((\beta_1 \cdot (u - 1) + \beta_2) \; mod \; n)$$

$$g(u) = \begin{cases} l_1(u), & \text{se } l_1(u) \neq u \\ 1 + (l_1(u) \; mod \; n), & \text{caso contrário} \end{cases}$$

$$l_2(u) = 1 + ((\pi_1 \cdot (u - 1) + \pi_2) \; mod \; n)$$

$$h(u) = \begin{cases} l_2(u), & \text{se } l_2(u) \neq u \text{ e } l_2(u) \neq g(u) \\ 1 + (l_2(u) \; mod \; n), & \text{se } (1 + (l_2(u) \; mod \; n)) \neq u \text{ e} \\ & \quad (1 + (l_2(u) \; mod \; n)) \neq g(u) \\ 1 + ((l_2(u) + 1) \; mod \; n), & \text{caso contrário} \end{cases}$$

Onde os valores de β_1, β_2, π_1 e π_2 são respectivamente 131, 1031, 193 e 1093.

A partir dessas funções definimos o conjunto de triplas proibidas como $T = \{(i, j, k) \in \mathcal{T} : \forall u \in [1, n], (i, j, k) = sort(\{u, g(u), h(u)\})\}$.

2. Metodologia

A Busca Tabu é uma metaheurística baseada em busca local proposta por Glover em 1986 que surgiu da ideia de utilizar uma **memória** para superar ótimos locais. A técnica permite a realização de movimentos que não melhoram a solução atual, evitando ciclos por meio de memórias chamadas de **listas tabu**. Uma lista tabu é uma estrutura de dados responsável por armazenar os últimos movimentos aplicados

por um determinado número de iterações, a fim de evitar que eles sejam desfeitos por novos movimentos. Assim, uma descrição básica do método consiste em construir uma solução inicial e realizar iterativamente um movimento de vizinhança que não seja proibido, guardando-o na lista tabu.

Em termos práticos, para adaptar a Busca Tabu a um problema de otimização combinatória, três elementos principais devem ser definidos: a construção da solução inicial, a estrutura de vizinhança e o tamanho da lista tabu (indicando a quantidade de iterações durante as quais a realização de um movimento será proibida). Nesta seção será descrito como os elementos da busca tabu foram definidos para o MAX-QBFPT.

Os critérios de parada utilizados foram o número de iterações e o tempo de execução máximo.

2.1. Heurística Construtiva

A construção da solução inicial segue uma estratégia gulosa no custo de inserção do elemento na solução atual. O elemento de melhor custo que gera uma solução factível (não viola triplas proibidas) é escolhido a cada iteração da heurística, até que um ótimo local seja encontrado, ou seja, a inserção de mais um elemento piora a solução corrente. O custo de um elemento é calculado a partir da contribuição dele para a solução. Se mais de um elemento possuir o mesmo custo de inserção, a escolha é feita aleatoriamente.

2.2. Estratégia Tabu

Para implementar a lista tabu foi utilizada uma *Double Ended Queue* (DEQUE) de inteiros, i.e. uma fila com inserção e remoção de elementos nas duas extremidades. Ao realizar um novo movimento, o elemento que foi inserido ou removido da solução é adicionado na lista tabu e, com isso, todo movimento envolvendo esse elemento se torna proibido durante as próximas *tenure* iterações, salvo os casos onde o critério de aspiração é satisfeito, em que *tenure* é o tamanho da lista.

Embora a lista tabu seja um artifício crucial para evitar que o algoritmo entre em ciclo, algumas vezes a proibição de um movimento pode impedir que uma melhora seja alcançada na solução. Uma forma de amenizar essa característica negativa é utilizar uma função responsável por definir se um movimento pode ser usado mesmo quando for um tabu. Essa função é chamada de critério de aspiração.

O critério de aspiração que foi implementando consiste em avaliar se a execução de um movimento resulta em uma solução com valor superior ao valor da melhor solução encontrada até o momento. Caso o movimento represente uma melhora, seu *status* tabu é ignorado e a alteração é empregada na solução.

Algumas vezes, uma busca tabu padrão pode não ser suficiente para explorar intensivamente regiões promissoras do espaço de busca ou cobrir regiões diversas o bastante para garantir que vários componentes de soluções foram consideradas durante o processo. Tendo isso em vista, estratégias de diversificação e intensificação podem ser empregadas a fim de melhorar a metaheurística.

2.2.1. Intensificação

É comum a busca tabu ficar por muitas iterações presa em regiões pouco promissoras, sem conseguir encontrar soluções melhores. Nesses casos, é desejável recomeçar a busca a partir da melhor solução conhecida, com as componentes mais interessantes fixadas. Esse é o princípio básico de uma estratégia de intensificação por reinício.

Para o QBFPT, consideramos como "componentes mais interessantes" os, no máximo, $\frac{n}{2}$ números que mais apareceram sem interrupção nas últimas iterações. Assim, quando a busca reinicia, esses números não podem ser removidos ou trocados até o fim da intensificação.

Alguns parâmetros, tais como a quantidade de falhas toleráveis até a intensificação ser ativada e o número de iterações em que as componentes permanecem fixas, podem ser ajustados e testados nas instâncias do problema.

2.2.2. Diversificação

Um dos principais problemas de métodos de busca local é a qualidade dos ótimos locais que são encontrados. Como a vizinhança é explorada a partir de um único início, a busca pode acabar restrita a uma pequena porção do espaço de soluções. Para contornar esse problema, métodos de diversificação podem ser empregados. Um dos mecanismos para aplicar a diversificação durante a busca tabu consiste no *strategic oscillation*.

No *strategic oscillation*, uma fronteira de oscilação (do inglês, *oscillation boundary*) é definida para o espaço de busca e o algoritmo realiza movimentos nas soluções a fim de atingir e atravessar essa região crítica de oscilação. Os movimentos são repetidamente realizados em direções opostas, estimulando o algoritmo a cruzar a fronteira continuamente e a considerar diferentes características de soluções.

Ao passo que o *strategic oscillation* é comumente implementado através de movimentos construtivos (que adicionam elementos na solução até que a fronteira seja atravessada) e movimentos destrutivos (que removem elementos para que a fronteira seja atravessada na direção oposta), segundo Glover, Pardalos e Laguna [2] uma forma alternativa de implementar essa estratégia é relaxar algumas das restrições do problema e introduzi-las na função objetivo como penalizações proporcionais às suas violações. Nesse caso, a fronteira de oscilação é definida pela viabilidade das soluções (violação de triplas proibidas) e o ajuste do parâmetro de penalização é realizado para que o algoritmo cruze a fronteira de viabilidade durante sua execução e permita a exploração de novas características para as soluções.

Para implementar o *strategic oscillation*, utilizamos uma penalização dinâmica (semelhante a Gendreau, Hertz e Laporte [3]). O fator de penalidade possui uma configuração inicial p_0, e a cada i iterações sem nenhuma solução viável ter sido encontrada, o fator de penalidade é dobrado. De maneira semelhante, a cada j iterações, caso nenhuma das i últimas soluções tenham sido inviáveis, o fator é diminuído pela metade. A função objetivo é penalizada pela adição do produto entre o fator de penalização e uma medida da quantidade de triplas violadas (no

caso de minimização).

Como o objetivo final ainda é encontrar a melhor solução viável possível para o problema, a solução titular é atualizada apenas quando a solução atual da busca é viável para o problema.

2.3. Movimentos de Vizinhança

A etapa de busca local consiste de um movimento de vizinhança na solução S construída anteriormente. Para o problema MAX-QBFPT, uma vizinhança imediata consiste nas soluções obtidas ao **inserir** ou **remover** um elemento em S, ou **trocar** um elemento de S por outro que não está na solução. A lista de candidatos é atualizada a cada iteração, para conter apenas elementos factíveis, que não violem as triplas proibidas. A exploração dessa vizinhança pode ser feita de dois modos: *first-improving* e *best-improving*.

A estratégia *best-improving* trata de avaliar todos os movimentos possíveis a partir da solução atual, e escolher aquele que fornece a maior melhoria. Em outras palavras, todas as inserções, remoções e trocas são avaliadas, e o melhor movimento é escolhido.

Na estratégia *first-improving*, para cada tipo de movimento foi considerado o primeiro candidato que melhora a solução atual. Em seguida, o melhor desses três movimentos é escolhido para o próximo passo da busca. Essa escolha foi feita de modo a não priorizar um tipo de movimento a outro.

3. Implementação e Avaliação

Assim como indicado na atividade, foram utilizados dois métodos de busca (*first-improving* e *best-improving*), dois valores para o parâmetro *tabu tenure* e duas estratégias tabu alternativas: intensificação por reinício e *strategic oscillation*.

Com base em testes empíricos, a tolerância para a quantidade de iterações sem melhora foi definida em 1000, exceto para a instância de tamanho 200, na qual o parâmetro foi aumentado para 2000, e a quantidade de iterações em que as componentes permanecerão fixas na solução foi definida em 100. A penalidade para a medida de diversificação p_0 foi configurada inicialmente como 100, e a quantidade de iterações i até reavaliação positiva foi fixada como 25. A quantidade de iterações j até reavaliação negativa foi fixada como 200.

As configurações utilizadas nos experimentos computacionais foram:

1. PADRÃO (C1): Busca Tabu com *tenure* $T_1 = 20$, *best-improving* e estratégia tabu padrão.

2. PADRÃO+IR (C2): Busca Tabu PADRÃO mas com intensificação por reinicio.

3. PADRÃO+SO (C3): Busca Tabu PADRÃO mas com *strategic oscillation*.

4. PADRÃO+IR+SO (C4): Busca Tabu PADRÃO mas com intensificação por reinicio e *strategic oscillation*.
5. PADRÃO+IR+SO+FI (C5): Busca Tabu PADRÃO mas com intensificação por reinicio, *strategic oscillation* e *first-improving*.
6. PADRÃO+IR+SO+T2 (C6): Busca Tabu PADRÃO mas com intensificação por reinicio, *strategic oscillation* e *tenure* $T_2 = 30$.

A implementação da solução foi feita em *Java*, com base no *framework* fornecido. Alteramos o código original para incluir o critério de parada por limite de tempo de execução e para implementar as modificações citadas na Seção 2.

As instâncias utilizadas em nossos experimentos foram fornecidas previamente no pacote de atividades. Tais instâncias foram criadas originalmente para o problema MAX-QBF, porém como também foram geradas o conjunto de triplas proibidas, foi possível reaproveitá-las. Algumas instâncias possuem solução ótima conhecida, outras possuem intervalos de referência, que podem ser usados para comparação. A configuração e os limitantes inferiores e superiores do valor ótimo de cada instância são apresentados na Tabela 1

Tabela 1: Instâncias utilizadas nos experimentos

i	Instância	n	MAX-QBF (Z^*)	MAX-QBFPT (Z^*)
1	qbf020	20	151	125
2	qbf040	40	429	366
3	qbf060	60	576	[508, 576]
4	qbf080	80	1000	[843, 1000]
5	qbf100	100	[1468, 1539]	[1263,1539]
6	qbf200	200	[5385, 5826]	[3813, 5826]
7	qbf400	400	[14826, 16625]	[9645, 16625]

O *hardware* usado para testes possui uma CPU "Intel(R) Core(TM) i5-8300H (4C/8T)", com 2.30 GHz de frequência do *clock*. O computador possui 32 GB de RAM, operando com 2667 MHz. O sistema operacional utilizado foi o Ubuntu 20.10 (64 bits). O método foi executado com limite de 1800 segundos (30 minutos), limite de 10000 iterações, e sem limite de memória.

4. Resultados Obtidos e Análise

Nas Tabelas 2 e 3 são apresentados os resultados obtidos. O tempo é medido em segundos e a iteração na qual o valor máximo foi obtido é apresentada na coluna *Iter*. Todos os testes alcançaram o número máximo de iterações antes do tempo limite. A corretude da geração de triplas foi testada separadamente, e os testes foram bem-sucedidos.

De acordo com as configurações da Tabela 2, é possível notar que as três abordagens convergiram para o mesmo custo de solução quando trata-se das instâncias menores ($n \leq 100$), com exceção da instância 20. Isso demonstra uma boa qualidade

da configuração padrão de busca tabu, e dificulta a comparação entre as diferentes configurações.

Ao comparar a estratégia de Intensificação por Reinício com as demais da Tabela 2, percebe-se que todas as soluções obtidas com essa configuração são melhores que as outras, o que atesta a necessidade e o benefício de intensificar a busca em regiões promissoras para o problema em questão. Ademais, com exceção das instâncias 20 e 200, a melhor solução encontrada pela abordagem com intensificação foi obtida com tempo de execução e número de iterações inferiores aos da configuração padrão, reforçando a qualidade do método.

Com relação ao *strategic oscillation*, somente na maior instância foram obtidos resultados melhores do que os da estratégia padrão, sendo que para as demais, com exceção da instância 200, os custos obtidos pelas duas abordagens foram equivalentes. Isso indica que as soluções inviáveis visitadas não levaram necessariamente a uma região melhor. Além disso, para algumas instâncias pequenas ($n = 60$ e $n = 80$), a estratégia C3 gastou um número de iterações bem superior para alcançar os mesmos resultados que as outras duas configurações, o que se deve em grande parte ao elevado número de soluções inviáveis que são visitadas no decorrer do processo.

Tabela 2: Resultados das configurações 1, 2 e 3.

Instância	PADRÃO (C1)			PADRÃO+IR (C2)			PADRÃO+SO (C3)		
	Valor	Iter.	Tempo (s)	Valor	Iter.	Tempo (s)	Valor	Iter.	Tempo (s)
20	120	4	0.025	125	1303	0.124	120	4	0.03
40	366	8478	1.242	366	2773	0.564	366	623	0.316
60	508	3764	1.614	508	1985	0.888	508	4297	4.507
80	851	6858	7.207	851	3131	2.784	851	8751	27.881
100	1263	4316	9.517	1263	4122	9.423	1263	70	0.432
200	4111	6110	119.262	4122	8738	160.449	4052	1990	126.412
400	10970	8455	1470.686	11207	7293	1177.912	11102	2821	1576.529

A fim de averiguar como a busca tabu se comporta quando estratégias de diversificação e intensificação são aplicadas concomitantemente durante a execução, na Tabela 3 foram reunidos os resultados da execução da metaheurística utilizando as estratégias *strategic oscillation* e Intensificação por Reinicio.

Comparando a configuração C4 da Tabela 3 com as configurações da Tabela 2, percebe-se que grande parte dos valores das soluções obtidas por C4 se repete dentre as configurações C1, C2 e C3. Um comportamento interessante ocorreu para a maior instância: a solução obtida por C4 foi superior, em termos de custo, a todas as soluções da Tabela 1. Além disso, esse resultado foi obtido com um número de iterações inferior ao de C2.

As colunas C4 e C5 da Tabela 3 apresentam os resultados da execução utilizando os métodos de busca *best-improving* e *first-improving*, respectivamente. Como é possível observar, para todas as instâncias, os melhores resultados foram alcançados

ao aplicar o *best-improving*, indicando que o método da primeira-melhora é o menos adequado para o problema de interesse.

Tabela 3: Resultados das configurações 4, 5 e 6.

Instância	PADRÃO+IR+SO (C4)			PADRÃO+IR+SO+FI (C5)			PADRÃO+IR+SO+T2 (C6)		
	Valor	Iter.	Tempo (s)	Valor	Iter.	Tempo (s)	Valor	Iter.	Tempo (s)
20	120	4	0.026	120	650	0.129	120	4	0.032
40	366	623	0.312	366	1279	0.584	366	5077	1.761
60	508	3571	3.396	501	729	0.803	500	4380	4.754
80	843	4416	12.323	821	2719	7.499	851	4286	12.223
100	1263	70	0.409	1263	3222	15.278	1263	2214	12.956
200	4052	1990	119.364	3831	386	15.538	4122	7007	449.224
400	11324	2895	1645.703	11155	330	100.058	11378	2458	1361.001

É visível que não houve uma configuração ideal, no qual foi possível obter as melhores soluções para todas as instâncias. A estratégia que utiliza apenas intensificação foi a única que alcançou a referência em todas as instâncias, então para o caso geral a configuração C2 tem boa qualidade.

Porém, a combinação entre intensificação e diversificação com *tenure* maior (C6) também mostrou sua qualidade. Embora em 2 instâncias não tenha retornado o melhor resultado observado (de tamanhos 20 e 60), as outras instâncias produziram soluções de custo melhor ou equivalente a qualquer outra configuração. Tal comportamento pode ser um indício de que a combinação de estratégias de intensificação e diversificação para o MAX-QBFPT podem ser promissoras na prática, haja vista que o principal interesse na aplicação de heurísticas se dá em instâncias grandes, onde é inviável utilizar métodos exatos.

Em geral, os resultados obtidos foram bons, ultrapassando o custo de referência das duas maiores instâncias em todos os métodos. Se filtrarmos os melhores resultados de cada método, todas as soluções ótimas conhecidas foram alcançadas, e todas as soluções de referência conhecidas foram alcançadas ou ultrapassadas, para as instâncias maiores. Com relação ao tempo de execução, observa-se uma tendência semelhante de crescimento em função do tamanho da instância para todas as configurações.

Referências

[1] M. Gendreau and J.-Y. Potvin, "Tabu search," in *Handbook of Metaheuristics* (M. Gendreau and J.-Y. Potvin, eds.), vol. 146 of *International Series in Operations Research & Management Science*, ch. 2, pp. 41–59, Springer Science+Business Media, 2010.

[2] F. Glover, M. Laguna, P. Pardalos, D.-Z. Du, and R. Graham, "Tabu search: effective strategies for hard problems in analytics and computational science," *Handbook of Combinatorial Optimization, 2nd ed, vol. XXI*, pp. 3261–3362, 01 2013.

[3] M. Gendreau, A. Hertz, and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Mgmt Sci*, vol. 40, pp. 1276–1290, 10 1994.