Busca Tabu aplicada ao problema de maximização de função quadrática com triplas proibidas

B. H. CLAUS[1], IC, Universidade de Campinas

C. F. BAZZANO [2], IC, Universidade de Campinas

P. V. SILVA [3], FEEC, Universidade de Campinas

O presente relatório apresenta a aplicação do algoritmo de Busca Padrão, e outras três modificações, ao problema de maximização de função quadrática com triplas proibidas. **Resumo**. RESUMO.

Palavras-chave. Busca Tabu, Função Binária Quadrática, Aproximação.

1. Introdução

Uma função binária quadrática (QBF) é uma função $f : \mathbb{B}^n \rightarrow \mathbb{R}$ que pode ser expressa como uma soma de termos quadráticos:

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

Onde $a_{ij} \in \mathbb{R} \ \forall \ (i, j = 1, ..., n)$ são os coeficientes da função f. Em notação matricial, uma QBF pode ser expressa como:

$$f(x) = x\prime \cdot A \cdot x \tag{1.2}$$

O problema de maximização de uma função binária quadrática (MAX-QBF) pode ser expresso como:

$$Z = max_x f(x), \tag{1.3}$$

O MAX-QBF é um problema NP-difícil, mesmo que nenhuma restrição adicional seja imposta sobre as variáveis binárias x. No entanto, se os coeficientes a_{ij} forem todos não-negativos, o problema torna-se trivial, uma vez que $x_i = 1 \ \forall \ (i = 1, ..., n)$ é uma solução ótima.

[1]BlenoClaus@gmail.com; Aluno Especial
[2]crisfbazz@gmail.com;
[3]patrickvs@hotmail.com; Aluno de Mestrado

1.1. Problema MAX-QBF com triplas proibidas

Neste trabalho apresentamos a variante do problema MAX-QBF, MAX-QBF com as restrições de triplas proibidas(MAX-QBFPT).

Dado um conjunto de todas as triplas ordenadas T, sem repetição, dos naturais 1 a n, ou seja $T = \{(i, j, k) \in \mathbb{N}^3 : 1 \leq i < j < k \leq n\}$. No problema MAX-QBFPT é dado um conjunto $\tau \subseteq T$ e deseja-se maximizar uma função binária quadrática tal que $x_i x_j e x_k$ não podem ser todos igual a 1. Isso é o equivalente à adicionar a restrição:

$$x_i + x_j + x_k \leq 2 \qquad \forall\ i = \{1, ..., n\} \tag{1.4}$$

ao problema MAX-QBF.

Neste trabalho as triplas proibidas foram definidas da seguinte forma: para cada $u \in [i, n]$ serão aplicadas duas funções $g, h : [1, n] \to [1, n]$ para gerar dois novos números que formarão uma tripla proibida. As funções g e h são definidas usando a função linear congruente l:

$$l(u) = 1 + ((\pi_1 \cdot u + \pi_2) \mod n) \tag{1.5}$$

Onde π_1 e π_2 são normalmente escolhidos como números primos.
Para impedir que $g(u) = u$, define-se a função g da seguinte forma:

$$g(u) = \begin{cases} l(u), & \text{se } l(u) \neq u \\ 1 + (l(u) \mod n), & \text{caso contrário} \end{cases} \tag{1.6}$$

Para impedir que $h(u) = u$ ou $h(u) = g(u)$, define-se a função h da seguinte forma:

$$h(u) = \begin{cases} l(u), & \text{se } l(u) \neq u \wedge l(u) \neq g(u) \\ 1 + (l(u) \mod n), & \text{se } (1 + (l(u) \mod n)) \neq u \wedge (1 + (l(u) \mod n)) \neq g(u) \\ 1 + (l(u) + 1 \mod n), & \text{caso contrário} \end{cases} \tag{1.7}$$

Neste trabalho, os números primos utilizados para a função g são $\pi_1 = 131$ e $\pi_2 = 1031$; para a função h os números primos são $\pi_1 = 193$ e $\pi_2 = 1093$.

2. Busca Tabu

O algoritmo de Busca Tabu (*Tabu Search – TS*) foi inicialmente proposto por Glover como uma estratégia para a solução de problemas de otimização combinatória [1]. Trata-se de uma extensão dos métodos de busca local, que o incentiva a continuar explorando o espaço de busca mesmo ocorrendo movimentos que piorem a solução atual, evitando passar por vizinhanças conhecidas. Este controle é feito por meio de uma lista tabu, que armazena soluções já visitadas visando restringir o espaço de busca à novas soluções e evitando cycling. Porém, a adição de elementos a lista tabu pode "congelar"o algoritmo de busca por restringir todos os possiveis movimentos e, portanto, busca-se artifícios que retirem elementos da lista tabus, os chamados critérios de aspiração [2]. O critério de aspiração mais simples – e utilizado neste trabalho – é permitir que um movimento, mesmo que esteja na lista tabu, seja executado se este melhorar a solução corrente.

O pseudo-código da Busca Tabu está a seguir [2]:

Algorithm 1 Busca Tabu

Escolha (construa) uma solução inicial S_0
Defina $S \leftarrow S_0$, $f^* \leftarrow f(S_0)$, $S^* \leftarrow S_0$, $T \leftarrow \phi$
enquanto Critério de parada não for satisfeito **faça**
    Selecione S em $argmin_{S' \in \overline{N}(S)}[f(S')]$
    se $f(S < f^*$ **então**
        $f^* \leftarrow f(S)$, $S^* \leftarrow S$
        Grave o movimento na lista Tabu T (apague o mais antigo se necessário)
    fim se
fim enquanto

onde:
S é a solução atual,
S^* é a solução incumbente,
f^* é o valor de S^*,
$N(S)$ é a vizinhança de S,
$\overline{N}(S)$, é o subespaço de $N(S)$ que não contem as soluções Tabu e
T é a lista tabu.

2.1. Diversificação por Reinício

A diversificação por reinício é uma metaheurística cuja ideia é tentar diversificar as soluções sendo produzidas para tentar fugir de mínimos ou máximos locais. Nesta trabalho implementamos a diversificação baseada na seguinte ideia: é passado como parâmetro um valor que chamamos de limite de diversificação cujo objetivo é contar quantas interações o algoritmo rodou sem melhorar a solução corrente. Na nossa implementação usamos o valor 1000 para esse limite em todos os testes. É salvo em uma pilha todas as soluções, todas as listas de candidatos a solução e todas as listas tabus. Além disso também salvamos a frequência em que os elementos permanecem na solução. Dessa forma é possível, ao atingir a condição de reinicio (quando o algoritmo atingir um número de interações são maior ou igual ao limite de diversificação sem melhorar a solução atual), descobrir quais elementos tem a menor frequência de permanecia na solução ótima de acordo com a solução escolhida para o reinicio. Note que o elemento de menor frequência depende da solução pois ele não pode estar na solução nem na lista tabu e tem de estar na lista de candidatos a solução (respeitando as triplas proibidas). Na nossa implementação, reiniciamos a diversificação sempre pela ultima solução da pilha e pegamos sempre os dois menores elementos dentre os possíveis para adicionar na solução, garantindo que com 2 não infringimos nenhuma tripla proibidas.

2.2. Busca Tabu Probabilística

A implementação padrão da Busca Tabu pode ter um alto custo computacional, haja visto que a cada iteração é calculado o custo da função objetivo para cada elemento da vizinhança $N(S)$. Objetivando a redução desse custo computacional, a Busca Tabu Probabilística considera apenas uma amostra aleatória de $N(S)$, denominada $N'(S)$. Além de reduzir o custo computacional, a inserção de aleatoriedade reduz a probabilidade de repetição de soluções, permitindo a utilização de uma lista tabu menor.

Em nossa implementação, foi criado um parâmetro $p \in [0, 1]$ que define a probabilidade de o elemento ser avaliado para inclusão ou remoção da solução. além disso, a cada iteração são criados dois vetores de probabilidade, um de dimensão igual à do vetor de elementos que não estão na solução, e outro de dimensão igual à do vetor solução, estes vetores são também usados na busca local de troca.

2.3. Busca Tabu Probabilística com Diversificação por Reinício

A busca tabu probabilística pode gerar amostragens $N'(S)$ que não cobrem igualmente o espaço amostral e que podem não incluir os mínimos locais. Caso medidas não sejam tomadas para explorar melhor o universo de soluções essa metaheurística pode acabar gerando soluções substancialmente piores do que aquelas obtidas com outros métodos. Pensando nisso implementamos um terceiro método hibrido que utiliza a diversificação para cobrir as falhas de cobertura da probabilística.

O método hibrido que implementamos utiliza a busca tabu probabilística para explorar o universo de soluções com um pouco de aleatoriedade. Para contornar o viés da amostragem $N'(S)$ nas soluções geradas nós salvamos a frequência com que cada elemento da lista CL é sorteado para poder participar da solução. Dessa forma conseguimos medir quais elementos foram negligenciados de participar das soluções obtidas até o momento e usamos essa informação para permitir que tais elementos tenham a chance de serem explorados. Utilizamos o método de diversificação por reinício para explorar os elementos negligenciados pela probabilística quando o critério de limite de diversificação é atingido e fazem pelo menos 5% do total de iterações que a solução incumbente não foi melhorada. O critério de limite de diversificação utilizado por nós é atingido quando 33% do total de iterações é executado, dessa forma a diversificação é aplicada 3 vezes durante toda a execução. Quando essas duas condições são satisfeitas reiniciamos a busca sem carregar nenhum elemento da solução atual, uma nova solução vazia é criada e inicializada a partir da frequência de sorteio dos elementos, e as listas TL e CL são reinicializadas. Inicializamos essa nova solução com os elementos que possuem uma frequência de sorteio abaixo de 33% da frequência máxima em relação a frequência miníma registrada, obedecendo as triplas proibidas. A frequência dos elementos escolhidos para compor essa nova solução é incrementada, eles são adicionados a lista tabu e removidos da CL e a busca continua a partir dessa solução com elementos pouco explorados.

2.4. Parâmetros Testados

Na tabela 1 temos os parâmetros utilizados para execução dos testes usando a busca tabu padrão e a busca tabu padrão com diversificação por reinicio:

Tabela 1: código para teste referente aos parâmetros testados com o algoritmo padrão e com diversificação por reinicio.

Busca Tabu	**Improving**	**Tenure**	**Código do teste**
Padrão	best	20	SB020
		100	SB100
	first	20	SF020
		100	SF100
Diversificação por Reinicio	best	20	DB020
		100	DB100
	first	20	DF020
		100	DF100

Como o algoritmo de busca tabu probabilística e o algoritmo hibrido possuem um parâmetro extra, os parâmetros testados foram reunidos na tabela 2, onde os prefixos dos códigos são P para o probabilístico e PD para o hibrido:

Tabela 2: código para teste referente aos parâmetros testados com o algoritmo probabilístico e híbrido.

Busca Tabu	**Improving**	**Tenure**	**Probabilidade**	**Código do Teste**
Probabilístico	best	20	25%	[prefixo]B02025
			50%	[prefixo]B02050
			75%	[prefixo]B02075
		100	25%	[prefixo]B10025
			50%	[prefixo]B10050
			75%	[prefixo]B10075
	first	20	25%	[prefixo]F02025
			50%	[prefixo]F02050
			75%	[prefixo]F02075
		100	25%	[prefixo]F10025
			50%	[prefixo]F10050
			75%	[prefixo]F10075

Foram adorados dois critérios de parada: : número máximo de iterações, acordo com a dimensão das instâncias, como podemos ver na tabela 3, e o tempo máximo de execução, que seja igual ao :

Tabela 3: Quantidade de iterações de acordo com a instância.

Instância	**Iteração**
qbf020	10^6
qbf040	10^6
qbf060	10^7
qbf080	10^7
qbf100	10^8
qbf200	10^8
qbf400	10^9

3. Resultados

Os resultados dos teste com a Busca Tabu padrão e com o Método de Diversificação por reinicio estão na tabela 4, onde estão destacados os melhores resultados para cada instância.

Tabela 4: Resultados dos testes executados para a Busca Tabu Padrão e com Diversificação por Reinicio.

Instância	**qbf020**	**qbf040**	**qbf060**	**qbf080**	**qbf100**	**qbf200**	**qbf400**
SB020	99	294	**470**	759	**1122**	3509	9717
SB100	99	262	426	716	999	3605	9682
SF020	**120**	254	414	758	961	3607	**10101**
SF100	99	294	403	716	1085	3566	9956
DB020	99	294	378	656	1092	**3822**	9899
DB100	**120**	294	**470**	759	1106	3801	9372
Df020	99	294	378	656	1092	**3822**	9899
Df100	**120**	**298**	403	**830**	1095	3736	9631

Os resultados dos testes do algoritmo probabilistico e hibrido estão na tabela 5:

A lista tabu de tamanho maior (ternure = 100) permitiu que resultados mais diversos fossem gerados e para varias instancias obteve os melhores resultados. A lista tabu de tamanho menor no método probabilístico permite uma maior exploração da amostragem sendo gerada o que para o método hibrido mostrou-se melhor visto que 3 reinicializações são feitas e temos um menor tempo de exploração em cada nova solução.

Para as quatro primeiras instancias que tivemos tempo para testar, o algoritmo hibrido foi capaz de encontrar uma solução melhor e soluções mais diversas. Este método poderia ser melhorado se os elementos mais sorteados fossem adicionados a lista tabu inicialmente para impedir que solução cicle para o mesmo ponto antes da diversificação.

A metaheuristica de diversificação poderia ser melhorada usando mais informação da frequência dos elementos e menos das soluções passadas.

Tabela 5: Testes dos algoritmos Probabilistico e Hibrido.

Código	**qbf020**	**qbf040**	**qbf060**	**qbf080**
PB02025	99	283	469	627
PB02050	120	283	470	770
PB02075	99	294	442	758
PB10025	120	271	427	758
PB10050	120	294	470	**830**
PB10075	99	294	453	757
PF02025	120	294	469	790
PF02050	120	297	470	788
PF02075	120	262	414	755
PF10025	120	254	370	746
PF10050	120	254	469	**830**
PF10075	120	254	470	750
PDB02025	120	271	405	647
PDB02050	**125**	304	**486**	**830**
PDB02075	**125**	271	378	758
PDB10025	120	246	469	788
PDB10050	**125**	**332**	457	757
PDB10075	120	277	453	769
PDF02025	120	298	374	748
PDF02050	99	298	426	772
PDF02075	120	294	414	828
PDF10025	120	254	409	784
PDF10050	120	294	327	758
PDF10075	120	294	484	762

A metaheuristica probabilística poderia ser melhorada com medidas para melhor explorar o espaço amostral, aplicando buscas locais mais intensas em determinadas iterações.

Referências

[1] F. Glover, "Tabu search—part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[2] M. Gendreau, J.-Y. Potvin, *et al.*, *Handbook of metaheuristics*, vol. 2. Springer, 2010.