Atividade 4
MC859/MO824

Ana Clara Zoppi Serpa, RA 165880, Instituto de Computação (IC), Universidade de Campinas, Campinas, SP.

Eduardo Barros Innarelli, RA 170161, Instituto de Computação (IC), Universidade de Campinas, Campinas, SP.

Resumo. Este documento é um relatório da Atividade 4 da disciplina de Tópicos em Otimização Combinatória ministrada no segundo semestre de 2020. O objetivo da atividade é estudar a solução do problema de maximização de uma função quadrática binária com triplas proibidas, utilizando GRASP com diferentes métodos de construção alternativos para compará-los. Também deseja-se comparar os métodos de busca *best-improving* e *first-improving*.

1. Problema Proposto e Modelo Matemático

1.1. Problema MAX-QBF

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

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

sendo $a_{ij} \in \mathbb{R}$ os coeficientes da função f. O problema de maximização de uma QBF (chamado de MAX-QBF) é NP-difícil, mesmo que nenhuma restrição adicional seja imposta para as variáveis x_i e x_j. No entanto, se todos os coeficientes a_{ij} forem não-negativos, o problema torna-se trivial pois basta que todos os x_i e x_j sejam iguais a 1 para obter a solução ótima.

1.2. Problema MAX-QBFPT

O problema MAX-QBFPT ("Maximum quadractic binary function with prohibited triples") é o seguinte: considere o conjunto $\mathcal{T}$ de todas as triplas ordenadas, sem repetição, dos naturais de 1 a n, ou seja, $\mathcal{T} = \{(i, j, k) \in \mathbb{N}^3 : 1 \leq i < j < k \leq n\}$. Dado um conjunto de triplas proibidas $T \in \mathcal{T}$, deseja-se maximizar uma QBF tal que x_i, x_j e x_k não podem ser todos iguais a 1, para todo $(i, j, k) \in T$. O problema é modelado da seguinte forma:

$$\max \quad Z = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot x_i \cdot x_j \tag{1.1}$$

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

$$x_i \in \mathbb{B}, \forall i = \{1, \ldots, n\} \tag{1.3}$$

Sendo $a_{ij} \in \mathbb{R}(i, j = 1, \ldots, n)$ os parâmetros do problema.

2. Metodologia

2.1. Geração das triplas proibidas

Para cada natural $u \in [1, n]$, foram aplicadas duas funções $g, h : [1, n] \to [1, n]$ para gerar dois novos números que formaram uma tripla proibida. As funções g e h foram definidas tomando como base a seguinte função linear congruente l: $l(u) = 1 + ((\pi_1 \cdot (u-1) + \pi_2) \bmod n)$, sendo π_1 e π_2 números primos.

Para impedir que $g(u) = u$ em algum caso,

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

E, para impedir que $h(u) = u$ ou que $h(u) = g(u)$, h foi definida assim:

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

Os números primos usados para a função g foram $\pi_1 = 131$ e $\pi_2 = 1031$. Para h, os números primos foram $\pi_1 = 193$ e $\pi_2 = 1093$. Assim, o conjunto de triplas proibidas é $T = \{(i,j,k) \in \mathcal{T} : \forall u \in [1,n], (i,j,k) = sort(\{u, g(u), h(u)\})\}$.

2.2. GRASP

Neste trabalho foi implementado o GRASP (*greedy randomized adaptive search procedure*) para solucionar o problema, que consiste de uma metaheurística iterativa onde cada iteração possui uma fase construtiva seguida de uma busca local. Em linhas gerais, uma solução é construída na primeira etapa e sua vizinhança é investigada na segunda visando encontrar uma solução de melhor custo.

2.2.1. Heurística Construtiva

Na heurística construtiva, os números candidatos a entrarem na solução são avaliados de forma gulosa no que diz respeito aos seus custos de inserção: quanto maior esse custo, mais próximo do máximo a solução se encontra. No entanto, para ampliar a variância entre iterações, ao invés de selecionarmos o número cujo custo c melhor afeta a solução, concebemos uma lista de candidatos restrita (RCL, do inglês Restricted Candidate List), que engloba todos elementos com custo no intervalo $[c^{\min}, c^{\min} + \alpha \cdot (c^{\max} - c^{\min})]$. Um número na RCL é, então, selecionado aleatoriamente para ingressar na solução, e o processo se repete enquanto a inserção de um candidato melhorar a solução ou a lista de candidatos esvaziar. Note que o parâmetro α controla o quão guloso ou aleatório é o algoritmo.

Nesse ponto, a diferença entre o MAX-QBF e o MAX-QBFPT está na possibilidade de, no MAX-QBFPT, a introdução de um novo elemento impedir alguns candidatos de entrarem na solução, por formarem triplas proibidas com outros dois elementos dela. Ou seja, é preciso atualizar a lista de candidatos em cada iteração da fase construtiva. Para isso, percorremos todas as triplas e, para cada combinação de dois números de uma tripla, verificamos se o terceiro elemento está na lista de candidatos — se sim, o removemos. Assim, ao fim da etapa construtiva temos uma solução válida para o MAX-QBFPT.

2.2.2. Métodos Alternativos de Construção

Optamos por implementar os métodos alternativos de construção *Reactive GRASP* e *Bias functions*.

No *Reactive GRASP*, é manipulada uma lista de parâmetros $\{\alpha_1, ..., \alpha_m\}$ dentre os quais um é selecionado para atuar na construção da RCL, todos inicializados com a mesma probabilidade de escolha $\frac{1}{m}$. Seja A_i a média das soluções que usaram α_i, e z^* a melhor solução encontrada até então. A probabilidade de um α ser selecionado é atualizada em tempo de execução por $p_i = q_i / \sum_{j=1}^{m} q_j$, onde $q_i = z^*/A_i, \forall i \in \{1, \ldots, m\}$, a cada certa quantidade de iterações (escolhemos $\sqrt{m}$ na nossa implementação). Dessa forma, é dado mais peso aos parâmetros α que retornaram melhores soluções.

Já no segundo método, afetamos diretamente a seleção de um elemento da RCL a partir de uma função *bias*. É atribuído um rank r a cada candidato, correspondente à sua posição na lista ordenada pelos custos de inserção, e selecionado um candidato com uma probabilidade equivalente ao seu *bias*. A seleção passa a favorecer candidatos de ranks melhores, tornando-se enviesada. No nosso caso escolhemos uma função *bias* linear, $\frac{1}{r}$.

2.2.3. Busca Local

Construída uma solução, o algoritmo passa para a fase de busca local. Como uma solução, completa ou parcial, consiste de uma lista de números, utilizamos os operadores básicos de inserção, remoção e troca para explorar sua vizinhança. No método de busca *best-improving*, já implementado no framework de apoio disponibilizado pelos docentes, todas operações possíveis são aplicadas e a busca prossegue com o melhor vizinho encontrado. No *first-improving*, que implementamos, o vizinho escolhido é o primeiro que melhora a solução corrente. Em ambos os métodos, se nenhum vizinho retornar um custo melhor que o corrente, a busca é interrompida.

2.2.4. Critérios de Parada

O máximo de iterações, em todos os experimentos, foi 1000. Limitamos também o tempo de execução para cada instância em 30 minutos.

3. Resultados Experimentais

Os experimentos foram realizados em uma máquina Intel(R) Core(TM) i5-2537M 1.40 GHz com 3 GB de RAM. A modelagem foi implementada em Java, utilizando o framework disponibilizado pelos docentes para apoiar a atividade. A Tabela 2 mostra os resultados obtidos (soluções e tempos de execução em segundos) para os diferentes métodos de construção, busca e valores de α experimentados. Valores de solução destacados em negrito são valores ótimos de solução (ou estão dentro do intervalo fornecido com limitantes para o valor ótimo), células com — na coluna referente ao tempo indicam casos nos quais a execução foi interrompida por ter passado de 30 minutos (nestes casos, é apresentada a solução encontrada antes da interrupção). Para os experimentos com *reactive GRASP*, o valor de α apresentado é o valor de α ao final da execução, já que o algoritmo altera os possíveis α ao longo do tempo, conforme explicado na seção anterior.

Os valores ótimos conhecidos para o problema foram disponibilizados no enunciado da atividade para fins de comparação.

Tabela 1: Soluções ótimas conhecidas

Instância	**Solução ótima (sem triplas proibidas)**	**Solução ótima (com triplas proibidas)**
qbf20	151	125
qbf40	429	366
qbf60	576	[508, 576]
qbf80	1000	[843, 1000]
qbf100	[1468, 1539]	[1263, 1539]
qbf200	[5385, 5826]	[3813, 5826]
qbf400	[14826, 16625]	[9645, 16625]

4. Análise dos Resultados

Observação: nas análises a seguir, quando dizemos que foram encontradas soluções ótimas, nos referimos a soluções de fato ótimas e iguais às conhecidas disponibilizadas no enunciado, e também a soluções que estão dentro dos intervalos fornecidos. Por exemplo, se uma solução 1263 foi encontrada e o intervalo é [1263, 1539], ela está sendo considerada ótima nos textos abaixo para facilitar a redação — apesar de na verdade apenas estar no intervalo.

Relação entre tempo de execução e tamanho da instância. Em todos os conjuntos de experimentos, o tempo de execução aumentou conforme o tamanho da instância aumentou. Em todos os conjuntos de experimentos, o tempo de execução da instância de tamanho 400 ultrapassou 30 minutos e o experimento foi interrompido. Em dois casos — caso 1: QBF sem triplas proibidas, com *best-improving*, $\alpha = 0.05$; caso 2: QBF com triplas proibidas, com *best-improving*, com *reactive GRASP* —, o tempo de execução da instância de tamanho 200 também ultrapassou 30 minutos e foi interrompido.

Otimalidade de soluções #1. Para o QBF sem triplas proibidas, utilizando o método de construção padrão, $\alpha = 0.05$ e o método de busca *best-improving*, foram encontradas as soluções ótimas para todas as instâncias, exceto a instância qbf400. No entanto, o valor da solução para a instância qbf400 antes de sua execução ser interrompida (14812.0) não está tão distante do início do intervalo de solução ótima fornecido (14826).

Otimalidade de soluções #2. Para o QBF sem triplas proibidas, utilizando o método de construção padrão, $\alpha = 0.05$ e o método de busca *first-improving*, a solução ótima foi encontrada apenas para a instância qbf40. Isso indica que utilizar *first-improving* com $\alpha = 0.05$ para o QBF sem triplas proibidas não foi uma boa estratégia. Resultados melhores com *first-improving* talvez pudessem ser obtidos experimentando valores diferentes de α.

Otimalidade de soluções #3. Para o QBF com triplas proibidas, utilizando o método de construção padrão, $\alpha = 0.05$ e o método de busca *best-improving*, foram encontradas as soluções ótimas para todas as instâncias, exceto qbf20 e qbf100. Para qbf20, a solução encontrada foi 120.0, sendo a solução ótima 125.0. Para qbf100, a solução encontrada foi 1258.0, e sabe-se que a solução ótima está no intervalo [1263, 1539]. Apesar de não serem soluções ótimas, não estão tão distantes dos valores ótimos. No entanto, esperávamos maior dificuldade em encontrar soluções ótimas para instâncias maiores, por exemplo qbf200 e qbf400. Acreditamos que isso possa ter ocorrido devido à natureza aleatória do algoritmo, ou a 0.05 não ser uma boa escolha para o valor de α.

Otimalidade de soluções #4. Para o QBF com triplas proibidas, utilizando o método de construção padrão, $\alpha = 0.05$ e o método de busca *first-improving*, foram encontradas as soluções ótimas apenas para as instâncias qbf200 e qbf400. Isso indica que utilizar *first-improving* com $\alpha = 0.05$ não é uma boa estratégia para o problema. Acreditamos que as soluções ótimas encontradas para qbf200 e qbf400 se devem ao fato de o intervalo ao qual a solução ótima pertence ser bem grande para essas duas instâncias, e não necessariamente à qualidade da estratégia. Valores diferentes de α talvez possam fornecer melhores resultados.

Otimalidade de soluções #5. Para o QBF com triplas proibidas, utilizando o método de construção padrão, $\alpha = 0.2$ e o método de busca *best-improving*, foram encontradas soluções ótimas para todas as instâncias. Isso indica que $\alpha = 0.2$ é um bom valor para este problema, com essas instâncias.

Comparação de diferentes valores de α #1. Para o QBF com triplas proibidas, utilizando o método de construção padrão e o método de busca *best-improving*, $\alpha = 0.2$ obteve soluções melhores que $\alpha = 0.05$, mas os tempos de execução foram maiores. A diferença entre os tempos de execução é inicialmente bem pequena (casos qbf20, qbf40, qbf60, qbf80), e vai aumentando conforme o tamanho das instâncias aumenta. Neste caso, obtemos soluções de melhor qualidade, a custo de mais tempo de execução.

Comparação de *best-improving* e *first-improving*. Para o QBF sem triplas proibidas, com método de construção padrão e um mesmo $\alpha = 0.05$, o método de busca *first-improving* foi pior quanto à qualidade das soluções — encontrou menos soluções ótimas. Para o QBF com triplas proibidas, com o mesmo método de construção e o mesmo $\alpha = 0.05$, o *first-improving* também foi pior em termos de otimalidade de soluções. Quanto a tempo de execução, tanto para o QBF sem triplas proibidas como para o QBF com triplas proibidas, o *first-improving* foi mais rápido — o que está de acordo com o esperado, afinal, em vez de percorrer todas as soluções para selecionar a melhor delas, escolhe-se a primeira que apresente melhora em relação à solução atual. Talvez a utilização de diferentes valores de α possa melhor aproveitar o *first-improving*, assim combinando otimalidade e velocidade, mas não chegamos a explorar isso nos experimentos deste trabalho.

Otimalidade de soluções #6. Para o QBF com triplas proibidas, utilizando o método de construção *reactive GRASP* e o método de busca *best-improving*, foram encontradas soluções ótimas para todas as instâncias.

Comparação de *reactive GRASP* e método de construção padrão. Os experimentos com *reactive GRASP* e os com método de construção padrão com $\alpha = 0.2$ encontraram soluções ótimas, mas os tempos de execução do método *reactive GRASP* foram maiores, como esperado, afinal fazer com que o valor de α se ajuste ao longo do algoritmo requer mais operações. Isso sugere que, quando não se conhece um bom valor de α, utilizar *reactive GRASP* é uma boa opção, mas, se um valor adequado para α já for conhecido (no caso dos nossos experimentos, acreditamos que $\alpha = 0.2$ seja um bom valor), utilizá-lo com o método de construção padrão pode dar soluções ótimas em menor tempo de execução. O *reactive GRASP* pode inclusive sugerir qual valor de α seria adequado. Por exemplo, pode-se observar na tabela que os valores de α ao final da execução com *reactive GRASP* foram, em três casos, 0.25 — um valor bem próximo do valor adequado 0.2 que encontramos no experimento anterior.

Otimalidade de soluções #7. Para o QBF com triplas proibidas, utilizando o método de construção *bias*, $\alpha = 0.05$ e o método de busca *best-improving*, apenas as instâncias qbf100, qbf200 e qbf400 apresentaram soluções ótimas.

Otimalidade de soluções #8. Para o QBF com triplas proibidas, utilizando o método de construção *bias*, $\alpha = 0.2$ e o método de busca *best-improving*, todas as instâncias apresentaram soluções ótimas.

Comparação de diferentes valores de α #2. Tanto com método de construção padrão e *best-improving* como com método de construção *bias* e *best-improving*, $\alpha = 0.2$ obteve soluções melhores que $\alpha = 0.05$, o que indica que $\alpha = 0.2$ é um bom valor nestes dois cenários, e talvez em geral para este problema. Seria possível verificar isso testando $\alpha = 0.2$ com outros métodos de construção e outras instâncias do QBFPT — seria um bom experimento para um trabalho futuro. Quanto ao tempo de execução, experimentos com $\alpha = 0.2$ foram mais lentos, assim como na comparação anterior (#1) de valores de α. Novamente observa-se um aumento no tempo de execução em troca da obtenção de soluções ótimas.

Comparação de *bias* e método de construção padrão. Para $\alpha = 0.05$, o método de construção padrão encontrou soluções ótimas para o QBFPT em 5 das 7 instâncias, enquanto o método de construção *bias* encontrou soluções ótimas em 3 das 7 instâncias. Para $\alpha = 0.2$, o método de construção padrão encontrou soluções ótimas para todas as instâncias do QBFPT, e o método *bias* também. Em alguns casos, o tempo de execução com *bias* foi maior que com o método padrão, e, em outros, foi menor. Assim, não podemos afirmar se o método *bias* é melhor em termos de tempo de execução, nem em termos de otimalidade das soluções, mas o fato de bons resultados terem sido obtidos com $\alpha = 0.2$ para os dois métodos indica que o α escolhido tem mais impacto na otimalidade que uma escolha entre utilizar ou não utilizar *bias* — pelo menos para este problema, para essas instâncias. Seria possível averiguar isso com testes em instâncias diferentes, e seria um bom experimento futuro.

Comparação de *bias* e *reactive GRASP*. O *reactive GRASP* encontrou soluções ótimas para todas as instâncias, assim como o *bias*, quando configurado com $\alpha = 0.2$. Quanto ao tempo de execução, os experimentos com *bias* foram mais rápidos. Isso sugere que, se for conhecido um valor adequado de α, utilizar o método *bias* pode ter vantagens em relação ao *reactive GRASP*.

Comparação dos problemas QBF e QBFPT. Utilizando o método de construção padrão, $\alpha = 0.05$ e o método de busca *best-improving*, foi possível obter soluções ótimas para o QBF em 6 das 7 instâncias, e em 5 das 7 instâncias para o QBFPT. Trocando o método de busca para *first-improving*, foi encontrada solução ótima em 1 das 7 instâncias para o QBF, e em 2 das 7 para o QBFPT. A primeira comparação, referente ao *best-improving*, sugere que o QBF é mais fácil de resolver, mas a segunda comparação, referente ao *first-improving*, sugere que o QBFPT seja mais fácil. Assim, essas duas comparações se contradizem e, como nós não exploramos diferentes valores de α para o QBF, ou diferentes métodos de construção, tendo em vista que o problema principal para estudo foi o QBFPT, não é possível afirmar somente a partir dos resultados da tabela qual dos problemas é mais difícil.

5. Principais Conclusões

Observamos que conseguimos obter soluções ótimas (ou dentro dos intervalos no qual a ótima está) para todas as instâncias do QBFPT com as seguintes configurações: construção padrão + busca *best-improving* + $\alpha = 0.2$; construção *reactive GRASP* + busca *best-improving*; construção *bias* + busca *best-improving* + $\alpha = 0.2$, sendo a configuração com *reactive GRASP* a mais lenta. Isso indica que o uso do *reactive GRASP* é vantajoso quando não se conhece um bom α, mas que, sendo conhecido um α adequado, utilizá-lo com o método de construção padrão ou com *bias* pode ser mais vantajoso.

Em todos os experimentos, *best-improving* mostrou-se melhor que *first-improving* para a qualidade das soluções, a custo de mais tempo de execução. Além disso, foi observado que o valor escolhido para α afetou significativamente a otimalidade das soluções encontradas em dois métodos distintos (*bias* e construção padrão), o que indica que estudar diferentes valores de α para encontrar um equilíbrio adequado entre aleatoriedade e estratégia gulosa é importante para obter boas soluções para o problema QBFPT.

Tabela 2: Resultados Experimentais

Construção	**Busca**	α	**Triplas proibidas**	**Instância**	**Tempo (s)**	**Solução**
padrão	best-improving	0.05	não	qbf20	2.99	**151.0**
padrão	best-improving	0.05	não	qbf40	7.09	**429.0**
padrão	best-improving	0.05	não	qbf60	23.42	**576.0**
padrão	best-improving	0.05	não	qbf80	63.04	**1000.0**
padrão	best-improving	0.05	não	qbf100	161.66	**1468.0**
padrão	best-improving	0.05	não	qbf200	—	**5385.0**
padrão	best-improving	0.05	não	qbf400	—	14812.0
padrão	first-improving	0.05	não	qbf20	0.53	149.0
padrão	first-improving	0.05	não	qbf40	2.63	**429.0**
padrão	first-improving	0.05	não	qbf60	5.97	562.0
padrão	first-improving	0.05	não	qbf80	17.11	970.0
padrão	first-improving	0.05	não	qbf100	28.86	1445.0
padrão	first-improving	0.05	não	qbf200	375.19	5284.0
padrão	first-improving	0.05	não	qbf400	—	14048.0
padrão	best-improving	0.05	sim	qbf20	1.04	120.0
padrão	best-improving	0.05	sim	qbf40	3.50	**366.0**
padrão	best-improving	0.05	sim	qbf60	10.55	**508.0**
padrão	best-improving	0.05	sim	qbf80	22.37	**851.0**
padrão	best-improving	0.05	sim	qbf100	48.71	1248.0
padrão	best-improving	0.05	sim	qbf200	563.18	**4022.0**
padrão	best-improving	0.05	sim	qbf400	—	**11147.0**
padrão	first-improving	0.05	sim	qbf20	0.73	102.0
padrão	first-improving	0.05	sim	qbf40	2.43	298.0
padrão	first-improving	0.05	sim	qbf60	5.68	482.0
padrão	first-improving	0.05	sim	qbf80	12.48	837.0
padrão	first-improving	0.05	sim	qbf100	26.36	1247.0
padrão	first-improving	0.05	sim	qbf200	228.14	**4001.0**
padrão	first-improving	0.05	sim	qbf400	—	**10920.0**
padrão	best-improving	0.2	sim	qbf20	1.05	**125.0**
padrão	best-improving	0.2	sim	qbf40	3.61	**366.0**
padrão	best-improving	0.2	sim	qbf60	13.43	**508.0**
padrão	best-improving	0.2	sim	qbf80	34.26	**843.0**
padrão	best-improving	0.2	sim	qbf100	66.90	**1263.0**
padrão	best-improving	0.2	sim	qbf200	699.41	**3948.0**
padrão	best-improving	0.2	sim	qbf400	—	**10993.0**
reactive	best-improving	0.25	sim	qbf20	1.34	**125.0**
reactive	best-improving	0.25	sim	qbf40	6.55	**366.0**
reactive	best-improving	0.64	sim	qbf60	25.29	**508.0**
reactive	best-improving	0.39	sim	qbf80	77.88	**843.0**
reactive	best-improving	0.64	sim	qbf100	188.97	**1263.0**
reactive	best-improving	0.25	sim	qbf200	—	**4054.0**
reactive	best-improving	0.99	sim	qbf400	—	**10525.0**
bias	best-improving	0.05	sim	qbf20	0.92	99.0
bias	best-improving	0.05	sim	qbf40	3.16	352.0
bias	best-improving	0.05	sim	qbf60	10.12	490.0
bias	best-improving	0.05	sim	qbf80	26.36	826.0
bias	best-improving	0.05	sim	qbf100	48.19	**1263.0**
bias	best-improving	0.05	sim	qbf200	520.15	**4015.0**
bias	best-improving	0.05	sim	qbf400	—	**11122.0**
bias	best-improving	0.2	sim	qbf20	0.97	**125.0**
bias	best-improving	0.2	sim	qbf40	3.87	**366.0**
bias	best-improving	0.2	sim	qbf60	16.27	**508.0**
bias	best-improving	0.2	sim	qbf80	32.40	**843.0**
bias	best-improving	0.2	sim	qbf100	84.60	**1263.0**
bias	best-improving	0.2	sim	qbf200	1076.12	**4019.0**
bias	best-improving	0.2	sim	qbf400	—	**10945.0**