MO824 - Análise Comparativa entre as metaheurísticas *GRASP*, *Tabu*, e *GA* para a resolução do problema MAX-QBF com mochila

Lucas Guesser Targino da Silva (203534)

3 de julho de 2022

Esse trabalho tem como objetivo comparar os resultados obtidos pelas implementações de três metaheurísticas:

1. *Greedy Randomized Adaptive Search Procedure* (*GRASP*) [1]
2. *Tabu Search* (*Tabu*) [2]
3. *Genetic Algorithm* (*GA*) [3]

O problema resolvido foi o *MAX-QBF com mochila*, descrito no Apêndice A.

1 Metaheurísticas

Nessa seção são descritas todas as metaheurísticas utilizadas ao longo do trabalho.

1.1 *GRASP*

A metaheurística *GRASP* é descrita no Algoritmo 1, sua estratégia construtiva em Algoritmo 2, e sua estratégia de busca local em Algoritmo 3, todos descritos no Apêndice A.2.

No desenvolvimento do trabalho 4, explorou-se variações do *GRASP*, com diferentes parâmetros, estratégias construtivas, e buscas locais. Duas delas, as que apresentaram melhor desempenho, foram escolhidas para serem utilizadas neste trabalho.

Ambas usam a estratégia construtiva padrão (Algoritmo 2) e o parâmetro α com valor 0.2.

Elas diferem na estratégia de busca local, entretanto. A primeira variação, chamada *GRASP Best*, utiliza *Best Improving*, em que toda a vizinhança é percorrida e a melhor opção selecionada. A segunda variação, chamada *GRASP First*, utiliza *First Improving*, em que a busca na vizinhança retorna a primeira solução encontrada que seja melhor do que a atual.

Em ambas, *Best Improvinge First Improving*, a vizinhança é definida como o conjunto de soluções obtidas a partir da solução atual que tenham:

1. 1 elemento a mais - adição;
2. 1 elemento a menos - remoção;
3. 1 elemento trocado (equivalente a uma adição e uma remoção) - troca;

1.1.1 *GRASP Best*

1. Estratégia construtiva: **padrão** com α igual a **0.2**
2. Estratégia de busca local: **best improving**

1.1.2 *GRASP First*

1. Estratégia construtiva: **padrão** com α igual a **0.2**
2. Estratégia de busca local: **first improving**

1.2 *Tabu*

A metaheurística *Tabu* é descrita no Algoritmo 4.

No desenvolvimento do trabalho 5, explorou-se variações do *Tabu*, com diferentes parâmetros, buscas locais, e variações. Duas delas, as que apresentaram melhor desempenho, foram escolhidas para serem utilizadas neste trabalho.

Ambas usam:

1. solução inicial a solução da estratégia construtiva do *GRASP*, Algoritmo 2;
2. busca local *Best Improving*. Ela é similar a *Best Improving*descrita na Subseção 1.1, entretanto ela não considera a adição ou remoção de elementos que estão na lista tabu T (a menos que eles levem a uma solução melhor do que S^*);
3. *Tenure Ration*, parâmetro que controla o tamanho da lista tabu T, igual a 0.4, o que significa que T pode ter tamanho até 40% do tamanho da entrada.

A primeira variação, chamada *Tabu Vanilla*, implementa *Tabu* com as características acima. A segunda variação, chamada *Tabu com Intensificação e Diversificação*, inclui estratégias de diversificação e intensificação, descritas nas Subsubseção 1.2.1 e Subsubseção 1.2.2.

1.2.1 Estratégia de Intensificação

A estratégia de intensificação aumenta o tamanho da vizinhaça, ao invés de considerar 1 adição, 1 remoção e 1 troca, ela considera:

1. 2 adições e 1 remoção;
2. 2 remoções e 1 adição;
3. 2 adições e 2 remoções;

A intensificação é feita em torno da melhor solução conhecida. Ela é ativada quando passam-se muitas iterações (o critério de parada é definido como um número máximo de iterações, e "muitas iterações" significa 20% número máximo de iterações) sem melhora na solução ótima e sem sua ativação.

1.2.2 Estratégia de Diversificação

A estratégia de diversificação constroi uma nova solução, utilizando a estratégia construtiva do GRASP, e recomeça a busca nesse outro local do espaço de solução. Além disso, antes de começar a busca, faz-se uma busca intensiva (usando a estratégia de intensificação) em torno dessa nova solução.

Seu critério de ativação é o mesmo da Estratégia de Intensificação, excetuando-se que Diversificação é ativada apenas quando Intensificação não é (no programa, há um registro separado para quando cada uma delas foi ativada pela última vez).

Assim, quando se passaram muitas iterações sem melhora na solução ótima, primeiro tenta-se intensificação. Caso essa falhe, executa-se a diversificação.

1.2.3 *Tabu Vanilla*

1. Solução inicial: saída da estratégia construtiva do *GRASP*
2. *Tenure Ration*: **0.4**
3. Estratégia de busca local: ***Best Improving***

1.2.4 *Tabu com Intensificação e Diversificação*

1. Solução inicial: saída da estratégia construtiva do *GRASP*
2. *Tenure Ration*: **0.4**
3. Estratégia de busca local: ***Best Improving***
4. Adição das estratégias: **Intensificação e Diversificação**

1.3 *GA*

Referências

[1] M. G. Resende and C. C. Ribeiro, "Greedy randomized adaptive search procedures: advances and extensions," in *Handbook of metaheuristics*, pp. 169–220, Springer, 2019.

[2] 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. 41–56, Springer Science+Business Media, 2010.

[3] C. R. Reeves, "Genetic algorithms," in *Handbook of metaheuristics*, pp. 109–139, Springer, 2010.

[4] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," *Journal of combinatorial optimization*, vol. 28, no. 1, pp. 58–81, 2014.

[5] L. G. T. da Silva, "Mo824a-combinatorial-optimization," 2022. Disponível em `https://github.com/lucasguesserts/MO824A-combinatorial-optimization`.

Apêndice A *MAX-QBF com mochila* (MAX-KQBF)

Definição 1 (Conjunto Binário)**.** $\mathbb{B} = \{0, 1\}$

Definição 2 (Função Binária Quadrática (QBF))**.** É uma função $f : \mathbb{B}^n \to \mathbb{Z}$ da forma:

$$f(x) = \sum_{j=1}^{n} x_i \cdot a_{i,j} \cdot x_j = x^T \cdot A \cdot x$$

em que $a_{i,j} \in \mathbb{Z},\ \forall i, j \in \{1, \cdots, n\}$ e A é a matriz n por n induzida pelos $a_{i,j}$.

Definição 3 (Problema de Maximização de uma Função Binária Quadrática (MAX-QBF))**.** Dada uma QBF f, um MAX-QBF é um problema da forma:

$$\max_x f(x)$$

Fato 1. MAX-QBF é NP-difícil [4]

Definição 4 (Maximum knapsack quadractic binary function (MAX-KQBF))**.** Dada uma QBF f, um vetor $w \in \mathbb{Z}^n$, e um valor $W \in \mathbb{Z}$, um MAX-KQBF é um problema da forma:

$$\begin{aligned} \max \quad & f(x) \\ \text{subjected to} \quad & w^T x \leq W \\ & x \in \mathbb{B}^n \end{aligned}$$

A.1 Instâncias

Foram utilizadas as instâncias com características conforme a Tabela 1.

Instância	Num. Variáveis	Num. Possibilidades	Intervalo de Otimalidade
kqbf020	20	1.0e+06	[80, 151]
kqbf040	40	1.1e+12	[275, 429]
kqbf060	60	1.2e+18	[446, 576]
kqbf080	80	1.2e+24	[729, 1000]
kqbf100	100	1.3e+30	[851, 1539]
kqbf200	200	1.6e+60	[3597, 5826]
kqbf400	400	2.6e+120	[10846, 16625]

Tabela 1: Instâncias do problema MAX-KQBF. Os dados completos estão disponíveis em [5].

A.2 *GRASP*

Algorithm 1 *GRASP*

1: $S_{\text{best}} \leftarrow \varnothing$
2: **for** $k = 1, \ldots, N_{it}$ **do**
3: $\quad S \leftarrow$ Greedy-Randomized-Construction()
4: $\quad S \leftarrow$ Local-Search(S)
5: $\quad S_{\text{best}} \leftarrow \max\{S, S_{\text{best}}\}$
6: **return** S_{best}

Algorithm 2 Greedy-Randomized-Construction(α)

$S \leftarrow \varnothing$
$C \leftarrow E$ ▷ Candidates list
for $e \in C$ **do**
    $c(e) \leftarrow$ Incremental-Cost(e, S) ▷ Increment in the cost by adding e to S
while $C \neq \varnothing$ **do**
    $c_{min} = \min\{c(e) : e \in C\}$
    $c_{max} = \max\{c(e) : e \in C\}$
    $R \leftarrow \{e \in C : c(e) \leqslant c_{min} + \alpha(c_{max} - c_{min})\}$ ▷ Restricted candidates list
    $s \leftarrow$ Select-Random-Element(R)
    $S \leftarrow S \cup \{s\}$
    Update C
    Update $c(e)$
return S

Algorithm 3 Loca-Search(S)

while S is not local optimal **do**
    $S \leftarrow \arg\max_{S' \in N(S)} \{f(S')\}$ ▷ $N(S)$ is the neighborhood of S ▷ f is the goal function
return S

A.3 *Tabu*

Algorithm 4 *Tabu* (S_0)

$S \leftarrow S_0$ ▷ S_0 is the initial solution ▷ S is the current solution
$S^* \leftarrow S$ ▷ S^* is the current best solution
$f^* \leftarrow f(S^*)$ ▷ f is the goal function
$T \leftarrow \varnothing$ ▷ T is the tabu list
while not Termination-Criteria-Satisfied **do**
    $S \leftarrow \arg\max_{S' \in N(S,T)} \{f(S')\}$ ▷ $N(S,T)$ is the neighborhood of S limited by T
    if $f(S) > f^*$ **then**
        $S^* \leftarrow S$
        $f^* \leftarrow f(S)$
    Update T ▷ record moves and delete old entries
return S^*

Apêndice B Implementação e execução dos experimentos

O programs foram executados num ideapad S145 81S90005BR: Lenovo IdeaPad S145 Notebook Intel Core i5-8265U (6MB Cache, 1.6GHz, 8 cores), 8GB DDR4-SDRAM, 460 GB SSD, Intel UHD Graphics 620 no ambiente:

1. sistema operacional: Fedora 35
2. Java versão 17

3. Gradle versão 7.4

O desenvolvimento da solução do problema foi feito em Java, baseado nos frameworks disponibilizados pelos professores. O código pode ser encontrado em [5].