

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 apresentaram melhor desempenho, e por isso elas foram escolhidas para serem utilizadas nesse 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.

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*

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/M0824A-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 $f : \mathbb{B}^n \rightarrow \mathbb{Z}$ 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 $a_{i,j} \in \mathbb{Z}$, $\forall i, j \in \{1, \dots, n\}$ 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 quadratic 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:

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

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

```

Algorithm 2 Greedy-Randomized-Construction(α)

```

1:  $S \leftarrow \emptyset$ 
2:  $C \leftarrow E$  ▷ Candidates list
3: for  $e \in C$  do
4:    $c(e) \leftarrow \text{Incremental-Cost}(e, S)$  ▷ Increment in the cost by adding  $e$  to  $S$ 
5: while  $C \neq \emptyset$  do
6:    $c_{\min} = \min \{c(e) : e \in C\}$ 
7:    $c_{\max} = \max \{c(e) : e \in C\}$ 
8:    $R \leftarrow \{e \in C : c(e) \leq c_{\min} + \alpha(c_{\max} - c_{\min})\}$  ▷ Restricted candidates list
9:    $s \leftarrow \text{Select-Random-Element}(R)$ 
10:   $S \leftarrow S \cup \{s\}$ 
11:  Update  $C$ 
12:  Update  $c(e)$ 
13: return  $S$ 

```

Algorithm 3 Local-Search(S)

```

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

```

Apêndice B Implementação e execução dos experimentos

O programas 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].