LIII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2021

Análise Comparativa do Reactive GRASP usando Path-Relinking e Variable Neighborhood Search Aplicado ao Problema da Diversidade Máxima

Jeanne Pereira
Programa de Pós Graduação em Ciência da Computação (PPGCC) do Instituto de Ciências Exatas e Naturais (ICEN) da Universidade Federal do Pará (UFPA)
Caixa Postal 479 - 66.075-110 - Belém - PA - Brasil
jeanneop22@gmail.com

Filipe Saraiva
Programa de Pós Graduação em Ciência da Computação (PPGCC) do Instituto de Ciências Exatas e Naturais (ICEN) da Universidade Federal do Pará (UFPA)
Caixa Postal 479 - 66.075-110 - Belém - PA - Brasil
saraiva@ufpa.br

RESUMO

Este artigo apresenta a metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) aplicada ao Problema da Diversidade Máxima, o qual é NP-difícil. Uma extensão do GRASP básico denominada *Reactive* GRASP é testada com *Path-Relinking* (PR) e *Variable Neighborhood Search* (VNS). Vários testes são realizados utilizando um banco de dados de *benchmark* com vários datasets. Apesar dos resultados obtidos apresentarem qualidade similar em termos de valor da função objetivo os custos computacionais são significativamente diferentes, informando que o *reactive* GRASP supera os outros avaliados.

PALAVRAS CHAVE. Diversidade Máxima, Metaheurísticas, Reactive GRASP.

MH, OC e PM

ABSTRACT

This paper presents GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic applied to the Maximum Diversity Problem (MDP), that is NP-hard. An extension of basic GRASP called Reactive GRASP is tested separately with Path-Relinking (PR) and Variable Neighborhood Search (VNS). Several tests were performed using a benchmark database with several datasets. Despite the results obtained present similar quality in terms of objective function, the computational costs are significantly different, informing reactive GRASP outperforms the others evaluated.

KEYWORDS. Maximum Diversity, Metaheuristics, Reactive GRASP.

MH, CO and PM

1. Introdução

No problema da diversidade máxima busca-se selecionar um subconjunto de um conjunto de elementos, sendo que os elementos do subconjunto têm a maior variedade em suas caracaterísticas [Kochenberger e Glover, 1999]. A resolução do problema da diversidade máxima pode servir para várias aplicações, como para o gerenciamento de recursos humanos a fim de conter a perda de conhecimento institucional maximizando a diversidade daqueles que são mantidos na empresa; medição da biodiversidade cuja maximização é de grande importância para os sistemas ecológicos [Kochenberger e Glover, 1999]; fármacos, com a finalidade de maximizar a diversidade molecular para a seleção de um subconjunto desses componentes [Agrafiotis, 1997]; entre outras.

Em [Kuo et al., 1993] é apresentada uma prova que caracteriza o problema da diversidade máxima como do tipo NP-difícil. Para problemas NP-difícil há grande dificuldade em encontrar e provar a solução ótima e não há um algoritmo em tempo polinomial para resolvê-los - isso justifica o uso de metaheurísticas para encontrar uma boa solução em tempo computacional aceitável. Dentre as metaheurísticas, uma bastante conhecida é o GRASP, que é uma metaheurística gulosa com característica construtivista, contendo também uma busca local [Feo e Resende, 1995].

No estudo de metaheurísticas, diversos operadores alternativos são propostos de forma a tornar mais robusta a busca por melhores soluções. Por exemplo, no GRASP [Feo e Resende, 1995] foram desenvolvidos operadores como o *Path-Relinking* (PR) [Resende et al., 2010] e o *Variable Neighborhood Search* (VNS) [Hansen e Mladenović, 2001], que modificam a busca padrão da metodologia. Já o mecanismo *reactive* GRASP utilizado para melhorar a fase de construção do algoritmo, torna o GRASP básico menos dependente de ajuste de parâmetros.

Alguns trabalhos relacionados envolvendo o GRASP, o *path-relinking*, o *variable neighborhood search* e também o problema da diversidade máxima serão comentados a seguir. Em [Andrade et al., 2005] é apresentado a hibridização do GRASP com *path-relinking* e um pool de soluções de elite para o problema da diversidade máxima. Em [Marti et al., 2013] são apresentados vários métodos como o GRASP e o VNS que foram aplicados ao problema da diversidade máxima, utilizando como *benchmark* o MDPLIB que está disponível na OR-LIBRARY [Beasley, 1990]. Em [Festa et al., 2002] são testadas diversas combinações do GRASP, VNS e PR para o problema de MAX-CUT. Em [Moro, 2017] o GRASP e um algoritmo genético modificado são aplicados ao problema da diversidade máxima.

Em que pese, no geral, a utilização de operadores alternativos possibilita buscas mais robustas, há cenários onde o uso deles pode não levar a melhorias significativas na qualidade das soluções encontradas ou mesmo prejudicar outras métricas, como o desempenho.

A principal contribuição é que esse artigo descreve uma análise comparativa entre o *reactive* GRASP e os operadores citados para o problema da diversidade máxima. De antemão, percebeu-se que a qualidade das soluções obtidas em simulações realizadas em um grande conjunto de *benchmarks* foram muito próximas, ao contrário do tempo computacional que apresenta uma sólida vantagem de um dos métodos. As demais seções estão assim organizadas. Na Seção 2 é descrita a formulação matemática do problema, enquanto na Seção 3 serão descritos o GRASP básico e as estratégias de VNS e PR. A Seção 4 descreve as instâncias de teste utilizadas e os resultados obtidos com as diferentes variações do GRASP. Por fim, as conclusões do estudo são apresentadas na Seção 5.

2. Definição do Problema

O problema da diversidade máxima consiste em selecionar um subconjunto de m elementos de um conjunto de n elementos tal que a soma das distâncias entre os elementos escolhidos é

maximizada. Dado um conjunto de elementos $X = \{x_1, x_2, \cdots, x_n\}$ a formulação matemática do problema presente em [Kuo et al., 1993] e usada neste trabalho é apresentada na Equação (1):

$$\begin{aligned} &Maximizar\ Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \\ &sujeito\ a \sum_{i=1}^{m} x_i = m \\ &x_i \in \{0,1\},\ 1 <= i <= n \end{aligned} \tag{1}$$

Na Equação (1) as variáveis x_i e x_j são variáveis de decisão que podem assumir o valor 1 ou 0 e representam se no conjunto solução os elementos i e j estão presentes, caso o valor de x_i ou x_j seja 1 indica que o elemento i ou j está na solução e 0 caso contrário. d_{ij} é a distância euclidiana. Caso o elemento i tenha p atributos, então cada elemento é na verdade x_{if} com o índice f para as p componentes e para dois elementos x_{if} e x_{jf} a distância d_{ij} é dada pela Equação (2):

$$d_{ij} = \sqrt{\sum_{f=1}^{p} (x_{if} - x_{jf})^2} \tag{2}$$

Na seção seguinte será descrito o GRASP básico e as estratégias para aprimorá-lo comparadas nesse estudo: o *reactive* GRASP, o *path-relinking* e o *variable neighborhood search*.

3. GRASP

A metaheurística GRASP [Feo e Resende, 1995] é composta por três fases que se repetem no decorrer das iterações: primeiro, uma solução é construída de maneira semi-gulosa e, em seguida, é aplicado um método de busca local para verificar se é possível melhorá-la. Realizados esses passos, a solução é então comparada com a melhor solução encontrada até o momento para em caso positivo, substituí-la como a solução a ser apresentada.

Na fase de construção da solução, para a seleção do elemento inicial que irá para a lista restrita (em inglês *Restricted Candidate List* - RCL), foi calculado para cada um a soma das distâncias entre ele e todos os outros elementos, sendo selecionado o que possui a maior soma.

Na seleção dos demais para a RCL, foi utilizada uma medida baseada em qualidade - a RCL contém os elementos com o valor da função gulosa c de um elemento i entre um limite mínimo e máximo, sendo c^{max} a função gulosa de custo máximo e c^{min} a função gulosa de custo mínimo para todos os elementos, conforme a seguinte equação: $c^{min} + \alpha * (c^{max} - c^{min}) <= c(i) <= c^{max}$. O parâmetro α varia entre 0 e 1. Quando α é igual a 0 a construção é puramente aleatória, entretanto quando α é igual a 1 a construção é puramente gulosa. Para determinar o valor da função gulosa a medida que um novo elemento é adicionado na solução, calcula-se a soma das distâncias entre ele e os elementos que já estão na solução parcial. Depois de construída a RCL um elemento é selecionado aleatoriamente para compor a solução.

3.1. Reactive GRASP

No *reactive* GRASP o valor do parâmetro α usado para construir a RCL não é fixo. Ele varia a cada iteração a partir de uma lista contendo os possíveis valores que ele pode assumir. A seleção é guiada pelos valores da solução encontrados ao longo das iterações. O pseudocódigo do *Reactive* GRASP está no Algoritmo 1.

Algoritmo 1: Pseudocódigo do *Reactive* GRASP

```
1  Function Reactive_GRASP (InstanciaEntrada, ψ = {α_1, ··· , α_m}, MaxIter, BlockIter, Seed) :
       Result: MelhorSolEncontrada
2      p_i = 1/m, i = 1, ··· , m;
3      s_i = 0, i = 1, ··· , m;
4      na_i = 0, i = 1, ··· , m;
5      for k = 1, ··· , MaxIter do
6          Selecionar um α_i do conjunto ψ de acordo com a probabilidade p_i;
7          na_i ← na_i + 1;
8          Solucao ← ConstruirSolGulosaRandomica(α_i, Seed);
9          if Solucao não é viavel then
10             Solucao ← Reparar(Solucao);
11         end
12         Solucao ← BuscaLocal(Solucao);
13         AtualizaSolucao(Solucao, MelhorSolEncontrada);
14         s_i ← s_i + z*;
15         if mod (k, BlockIter) = 0 then
16             A_i ← s_i/na_i, i = 1, ··· , m ;
17             q_i = z*/A_i, i = 1, ··· , m;
18             p_i = q_i / Σ_{j=1}^{m} q_j , i = 1, ··· , m;
19         end
20     end
21 End Function
```

O Algoritmo 1 usa a regra proposta em [Prais e Ribeiro, 2000]: na linha 2 dado um conjunto com m valores possíveis para α, inicialmente cada valor tem a mesma probabilidade $p_i = \frac{1}{m}$, para $i = 1, \cdots, m$ de ser escolhido. Nas linhas 14 e 17 z^* é a melhor solução, na linha 16 A_i é a média de todas as soluções quando $\alpha = \alpha_i$, para $i = 1, \cdots, m$ e na linha 18 as probabilidades são atualizadas com $p_i = \frac{q_i}{\sum_{j=1}^{m} q_j}$, com $q_i = z^*/A_i$, para $i = 1, \cdots, m$. Valores maiores de q_i correspondem a valores mais adequados para o parâmetro α.

3.2. Path-Relinking

O *path-relinking* [Resende et al., 2010] funciona como uma estratégia de intensificação do GRASP. Este operador realiza alterações nos componentes de uma solução tentando atingir outra dada solução. No processo, as soluções geradas nesse percurso são avaliadas de forma a verificar se alguma delas melhora o valor da função objetivo original.

Em [Resende et al., 2010] é discutida a hibridização do *path-relinking* com um conjunto de soluções de elite. Nesse caso, é construído um conjunto que armazena as melhores soluções encontradas pelo GRASP. Caso esse conjunto não esteja completo e o candidato a entrar difira dos outros membros, então ele é adicionado; caso o conjunto esteja completo, o candidato é adicionado se for melhor que o pior elemento do conjunto. Quando o conjunto elite chega a um determinado tamanho, é aplicado o *path-relinking* utilizando a solução gerada na iteração corrente do GRASP e a melhor solução desse conjunto.

Outra variante do *path-relinking* é o *mixed path-relinking* [Resende et al., 2010], onde a cada iteração os papéis da solução inicial e da solução guia são invertidos. O *mixed path-relinking* hibridizado com um pool de soluções de elite foi adotado neste artigo como uma estratégia de intensificação após a busca local a fim de melhorar a solução.

O algoritmo *reactive* GRASP com *mixed path-relinking* que foi implementado está descrito no Algoritmo 2, na linha 13 quando o conjunto Elite está completo, ou seja, atinge o seu tamanho máximo a função `SelecionaSolElite(Elite)` seleciona a melhor solução do conjunto Elite. Na linha 17 a função `AtualizaElite(Elite, Solucao)` adiciona uma nova solução no conjunto Elite ou verifica se ele já está cheio para substituir a pior solução.

O método de otimização local utilizado neste trabalho com o *reactive* GRASP e com o *reactive* GRASP + *Path-Relinking* (PR) foi o *hill-climbing* [Luke, 2013]. No *hill-climbing*, a função de vizinhança seleciona aleatoriamente um elemento que não está na solução e o troca por outro também selecionado aleatoriamente que está na solução. A solução atual só é substituída se a solução gerada for melhor que ela.

Algoritmo 2: Pseudocódigo do *reactive* GRASP com *mixed path-relinking*

```
1  Function Reactive_GRASP_PR (InstanciaEntrada, ψ = {α_1, ··· , α_m}, MaxIter, BlockIter, Seed) :
       Result: MelhorSolEncontrada
2      p_i = 1/m, i = 1, ··· , m;
3      s_i = 0, i = 1, ··· , m;
4      na_i = 0, i = 1, ··· , m;
5      for k = 1, ··· , MaxIter do
6          Selecionar um α_i do conjunto ψ de acordo com a probabilidade p_i;
7          na_i ← na_i + 1;
8          Solucao ← ConstruirSolGulosaRandomica(α_i, Seed);
9          if Solucao não é viavel then
10             Solucao ← Reparar(Solucao);
11         end
12         Solucao ← BuscaLocal(Solucao);
13         if Elite está completo then
14             SolElite ← SelecionaSolElite(Elite);
15             MixedPathRelinking (Solucao, SolElite) ;
16         end
17         AtualizaElite(Elite, Solucao);
18         AtualizaSolucao(Solucao, MelhorSolEncontrada);
19         s_i ← s_i + z*;
20         if mod (k, BlockIter) = 0 then
21             A_i ← s_i/na_i, i = 1, ··· , m ;
22             q_i = z*/A_i, i = 1, ··· , m;
23             p_i = q_i / Σ_{j=1}^{m} q_j, i = 1, ··· , m;
24         end
25     end
26 End Function
```

3.3. Variable Neighborhood Search

O operador *Variable Neighborhood Search* (VNS) [Hansen e Mladenović, 2001] é baseado em mudanças de estrutura de vizinhanças. Nele, operam-se alterações na solução de forma a gerar outras soluções.

Vários esquemas do VNS utilizam uma busca local, como o *Variable Neighborhood Descent* (VND). A busca local pode consumir um tempo computacional significativo para grandes instâncias de um determinado problema. Nesse caso, uma variação conhecida como *Reduced* VNS (RVNS) propõe retirar essa etapa para assim reduzir o tempo computacional sem perda significativa da qualidade da solução [Hansen e Mladenović, 2001].

No problema da diversidade máxima a mudança de vizinhança é baseada na quantidade m de elementos do subconjunto que terá a diversidade maximizada. Seleciona-se aleatoriamente k elementos que estão na solução para permutar com k elementos selecionados aleatoriamente que não estão na solução. O valor de k varia de 1 até m. No *reactive* GRASP + RVNS que está no Algoritmo 3, na linha 12 a função `RVNS` foi utilizada como estratégia de busca local no lugar do *hill-climbing*, na linha 26 a função `Shake` gera um ponto aleatório x' na k-ésima vizinhança de x, enquanto na linha 27 a função `MudancaVizinhanca` é responsável por ir para esse vizinho caso ele seja melhor do que a solução atual.

Na Seção 4 são descritas as instâncias de teste do problema da diversidade máxima e os resultados obtidos pelo *reactive* GRASP, *reactive* GRASP + PR e *reactive* GRASP + RVNS.

4. Resultados

As instâncias de teste utilizadas estão disponíveis na OR-LIBRARY [Beasley, 1990] em um *benchmark* chamado MDPLIB. O MDPLIB também contém os melhores valores conhecidos para os datasets. Foram realizados testes para todas as instâncias do SOM-b, do GKD-c, para as 20 primeiras instâncias do MDG-b e para as 25 primeiras instâncias do MDG-a, totalizando 85 instâncias de testes. As instâncias possuem características diversificadas como por exemplo no dataset SOM-b a distância entre elas é inteira, já no GKD-c é real. Enquanto no MDG-a pode ser tanto real como inteira.

Considerando n como o tamanho do conjunto e o m como o tamanho do subconjunto a ser selecionado com a maior diversidade, na Tabela 1 tem-se os tamanhos das instâncias para cada

Algoritmo 3: Pseudocódigo do *reactive* GRASP com *Reduced* VNS

```
1  Function Reactive_GRASP_RVNS (InstanciaEntrada, ψ = {α_1, ··· , α_m}, MaxIter, BlockIter, Seed) :
       Result: MelhorSolEncontrada
2      p_i = 1/m, i = 1, ··· , m;
3      s_i = 0, i = 1, ··· , m;
4      na_i = 0, i = 1, ··· , m;
5      for k = 1, ··· , MaxIter do
6          Selecionar um α_i do conjunto ψ de acordo com a probabilidade p_i;
7          na_i ← na_i + 1;
8          Solucao ← ConstruirSolGulosaRandomica(α_i, Seed);
9          if Solucao não é viavel then
10             Solucao ← Reparar(Solucao);
11         end
12         Solucao ← RVNS (Solucao, k_max, t_max) ;
13         AtualizaSolucao(Solucao, MelhorSolEncontrada);
14         s_i ← s_i + z*;
15         if mod (k, BlockIter) = 0 then
16             A_i ← s_i/na_i, i = 1, ··· , m ;
17             q_i = z*/A_i, i = 1, ··· , m;
18             p_i = q_i / Σ_{j=1}^{m} q_j , i = 1, ··· , m;
19         end
20     end
21 End Function
22 Function RVNS (x, k_max, t_max) :
       Result: x
23     repeat
24         k ← 1;
25         repeat
26             x' ← Shake(x,k);
27             x,k ← MudancaVizinhanca(x, x', k);
28         until k = k_max;
29         t ← TempodeCPU()
30     until t > t_max;
31 End Function
```

dataset. Por exemplo, o dataset SOM-b tem uma instância em que o n é igual a 100 e o m é igual a 10.

Tabela 1: Tamanho das instâncias

SOM-b	n	100	100	100	100
	m	10	20	30	40
	n	200	200	200	200
	m	20	40	60	80
	n	300	300	300	300
	m	30	60	90	120
	n	400	400	400	400
	m	40	80	120	160
	n	500	500	500	500
	m	50	100	150	200
GKD-c	n	500			
	m	50			
MDG-a	n	500			
	m	50			
MDG-b	n	500	2000		
	m	50	200		

Após uma série de testes prévios foi definido o número máximo de iterações do GRASP em 100, o número máximo de iterações no *hill-climbing* também em 100 e o tamanho máximo do conjunto elite em 20. Para cada instância o GRASP foi executado 20 vezes. As simulações foram

executadas em um computador Intel Core i5, 2,50 Ghz, 4GB de memória em um Windows 8 64 bits. O GRASP foi implementado em Java.

Os melhores valores encontrados na literatura estão de acordo com a tabela[1] que está no *benchmark* MDPLIB. Para verificar o desempenho dos algoritmos foi utilizado o tempo médio computacional em segundos, para avaliar a qualidade das soluções foi utilizado a porcentagem de aproximação das soluções na Equação (3).

$$Porcentagem = \frac{melhorFuncaoObjetivo}{MDPLIB\_melhorvalor} * 100 \tag{3}$$

Na Tabela 2 a comparação das soluções obtidas para o dataset SOM-b mostra que os resultados obtidos pelos três métodos se aproximaram em mais de 99% dos melhores valores encontrados na literatura, sendo que para duas instâncias chegou-se a obter o melhor valor.

Tabela 2: Comparação dos resultados do SOM-b

Instância	Melhores valores	*Reactive* GRASP (Porcent.)	*Reactive* GRASP + PR (Porcent.)	*Reactive* GRASP + RVNS (Porcent.)
SOM-b_01_n100_m10	333	**333 (100,00%)**	**333 (100,00%)**	**333 (100,00%)**
SOM-b_02_n100_m20	1195	1182 (98,91%)	**1187 (99,33%)**	1181 (98,83%)
SOM-b_03_n100_m30	2457	**2457 (100,00%)**	**2457 (100,00%)**	**2457 (100,00%)**
SOM-b_04_n100_m40	4142	**4138 (99,90%)**	**4138 (99,90%)**	4137 (99,88%)
SOM-b_05_n200_m20	1247	**1241 (99,52%)**	1238 (99,28%)	1238 (99,28%)
SOM-b_06_n200_m40	4450	**4441 (99,80%)**	4430 (99,55%)	**4441 (99,80%)**
SOM-b_07_n200_m60	9437	**9404 (99,65%)**	9399 (99,60%)	9397 (99,58%)
SOM-b_08_n200_m80	16225	16148 (99,53%)	**16169 (99,65%)**	16163 (99,62%)
SOM-b_09_n300_m30	2694	**2669 (99,07%)**	2665 (98,92%)	2666 (98,96%)
SOM-b_10_n300_m60	9689	9620 (99,29%)	9611 (99,19%)	**9622 (99,31%)**
SOM-b_11_n300_m90	20743	**20657 (99,59%)**	20645 (99,53%)	20633 (99,47%)
SOM-b_12_n300_m120	35881	35775 (99,70%)	35763 (99,67%)	**35797 (99,77%)**
SOM-b_13_n400_m40	4658	4620 (99,18%)	**4642 (99,66%)**	4640 (99,61%)
SOM-b_14_n400_m80	16956	16827 (99,24%)	**16862 (99,45%)**	16853 (99,39%)
SOM-b_15_n400_m120	36317	36082 (99,35%)	36090 (99,37%)	**36100 (99,40%)**
SOM-b_16_n400_m160	62487	62235 (99,60%)	62268 (99,65%)	**62274 (99,66%)**
SOM-b_17_n500_m50	7141	**7086 (99,23%)**	7063 (98,91%)	7064 (98,92%)
SOM-b_18_n500_m100	26258	**26083 (99,33%)**	26061 (99,25%)	26070 (99,28%)
SOM-b_19_n500_m150	56572	56212 (99,36%)	**56296 (99,51%)**	56238 (99,41%)
SOM-b_20_n500_m200	97344	97010 (99,66%)	**97064 (99,71%)**	97006 (99,65%)

Na Figura 1 comparam-se os tempos médios computacionais ao se utilizar os três métodos para o SOM-b. O *reactive* GRASP com RVNS apresenta o maior tempo médio computacional para todas as instâncias. Para a instância SOM-b_20 o *reactive* GRASP executa em média 12,1475 s, já o *reactive* GRASP + PR em 19,0807 s e o *reactive* GRASP + RVNS em 45,2275 s.

Na Tabela 3 são apresentadas as comparações das soluções obtidas para o dataset GKD-c. A exemplo do *benchmark* anterior, os resultados obtidos pelos três métodos também se aproximaram em mais de 99% dos melhores valores encontrados na literatura.

Na Figura 2 comparam-se os tempos médios computacionais ao se utilizar os três métodos para o GKD-c. O *reactive* GRASP com RVNS apresenta o maior tempo médio computacional para todas as instâncias. Para a instância GKD-c_20 o *reactive* GRASP processa em média em 6,7784 s, já o *reactive* GRASP + PR em 12,3796 s e o *reactive* GRASP + RVNS em 48,98 s - ou seja, este último consegue ser 8 vezes mais lento que o primeiro, sendo que a qualidade das soluções encontradas é muito similar.

[1] "mdplib_2010_bestvalues"

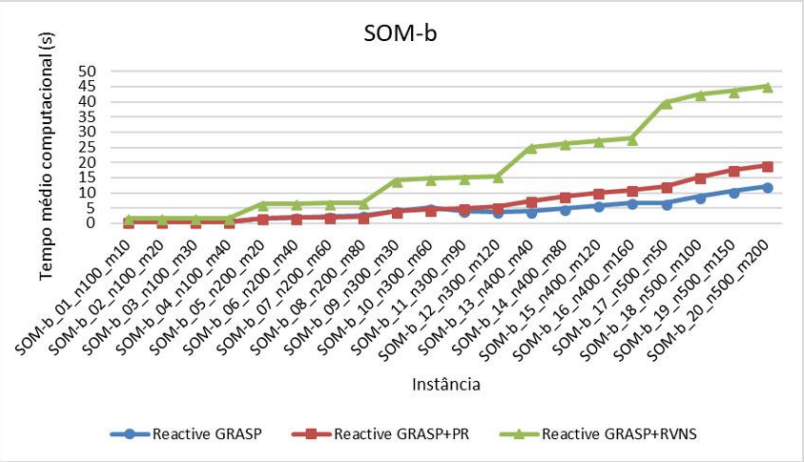

Figura 1: Tempo médio computacional para o SOM-b.

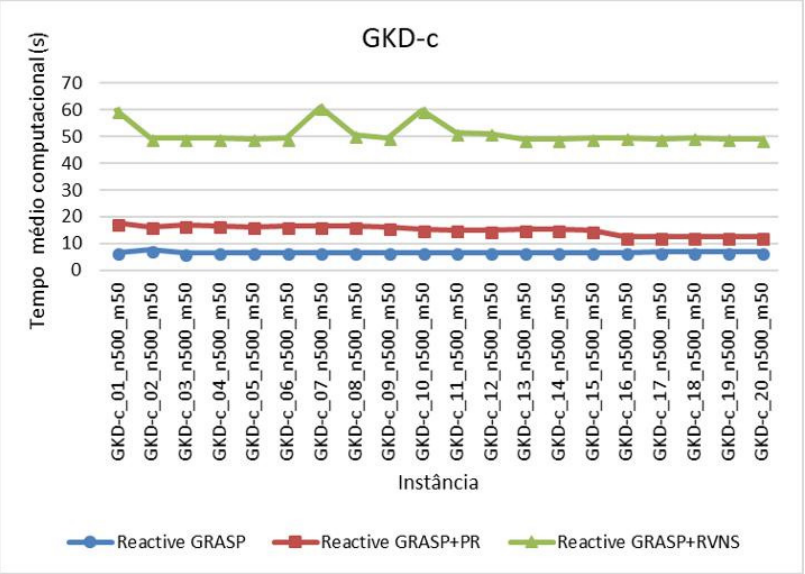

Figura 2: Tempo médio computacional para o GKD-c.

Na Tabela 4 são apresentadas as comparações para as soluções obtidas do dataset MDG-b. Os resultados obtidos pelos três métodos se aproximaram em mais de 98% dos melhores valores encontrados na literatura.

Na Figura 3 comparam-se os tempos médios computacionais dos métodos em análise. Como nos demais, o *reactive* GRASP com RVNS apresenta o maior tempo médio computacional para todas as instâncias. Como exemplo para a instância MDG-b_20 o *reactive* GRASP realiza a operação em média com 6,6527 s, já o *reactive* GRASP + PR em 13,3343 s e o *reactive* GRASP + RVNS em 49,087 s.

A Tabela 5 apresenta a comparação das soluções obtidas para as menores instâncias do dataset MDG-a. Os resultados obtidos pelos três métodos se aproximaram em mais de 98% dos melhores valores encontrados na literatura.

Tabela 3: Comparação dos resultados do GKD-c

Instância	Melhores valores	*Reactive* GRASP (Porcent.)	*Reactive* GRASP + PR ((Porcent.)	*Reactive* GRASP + RVNS (Porcent.)
GKD-c_01_n500_m50	19485,1875	19483,736 (99,99%)	**19485,18697 (100,00%)**	19483,64783 (99,99%)
GKD-c_02_n500_m50	19701,53711	19700,88197 (100,00%)	**19701,53477 (100,00%)**	19700,88197 (100,00%)
GKD-c_03_n500_m50	19547,20703	19544,4127 (99,99%)	**19546,90686 (100,00%)**	19543,91996 (99,98%)
GKD-c_04_n500_m50	19596,46875	**19596,46838 (100,00%)**	**19596,46838 (100,00%)**	**19596,46838 (100,00%)**
GKD-c_05_n500_m50	19602,625	**19602,62199 (100,00%)**	**19602,62199 (100,00%)**	**19602,62199 (100,00%)**
GKD-c_06_n500_m50	19421,55078	**19421,53908 (100,00%)**	**19421,53908 (100,00%)**	**19421,53908 (100,00%)**
GKD-c_07_n500_m50	19534,30664	19530,24813 (99,98%)	**19531,24106 (99,98%)**	19530,91494 (99,98%)
GKD-c_08_n500_m50	19487,32031	19486,41085 (100,00%)	**19486,67755 (100,00%)**	19485,97103 (99,99%)
GKD-c_09_n500_m50	19221,63477	**19221,62878 (100,00%)**	**19221,62878 (100,00%)**	**19221,62878 (100,00%)**
GKD-c_10_n500_m50	19703,35156	19701,13712 (99,99%)	**19703,33991 (100,00%)**	**19703,33991 (100,00%)**
GKD-c_11_n500_m50	19587,12891	19585,73776 (99,99%)	**19587,12018 (100,00%)**	19584,61159 (99,99%)
GKD-c_12_n500_m50	19360,23633	**19360,22393 (100,00%)**	**19360,22393 (100,00%)**	19360,05736 (100,00%)
GKD-c_13_n500_m50	19366,69922	19357,94589 (99,95%)	**19366,14942 (100,00%)**	19357,2807 (99,95%)
GKD-c_14_n500_m50	19458,56641	19458,47134 (100,00%)	**19458,56466 (100,00%)**	19458,47134 (100,00%)
GKD-c_15_n500_m50	19422,15039	19421,07509 (99,99%)	**19422,1464 (100,00%)**	19422,05063 (100,00%)
GKD-c_16_n500_m50	19680,20898	19678,1902 (99,99%)	**19679,45825 (100,00%)**	**19679,45825(100,00%)**
GKD-c_17_n500_m50	19331,38867	**19331,38841 (100,00%)**	**19331,38841 (100,00%)**	19329,42713 (99,99%)
GKD-c_18_n500_m50	19461,39453	**19461,3946 (100,00%)**	**19461,3946 (100,00%)**	**19461,3946 (100,00%)**
GKD-c_19_n500_m50	19477,32813	19473,34172 (99,98%)	**19474,80039 (99,99%)**	19474,33049 (99,98%)
GKD-c_20_n500_m50	19604,84375	**19604,84356 (100,00%)**	**19604,84356 (100,00%)**	**19604,84356 (100,00%)**

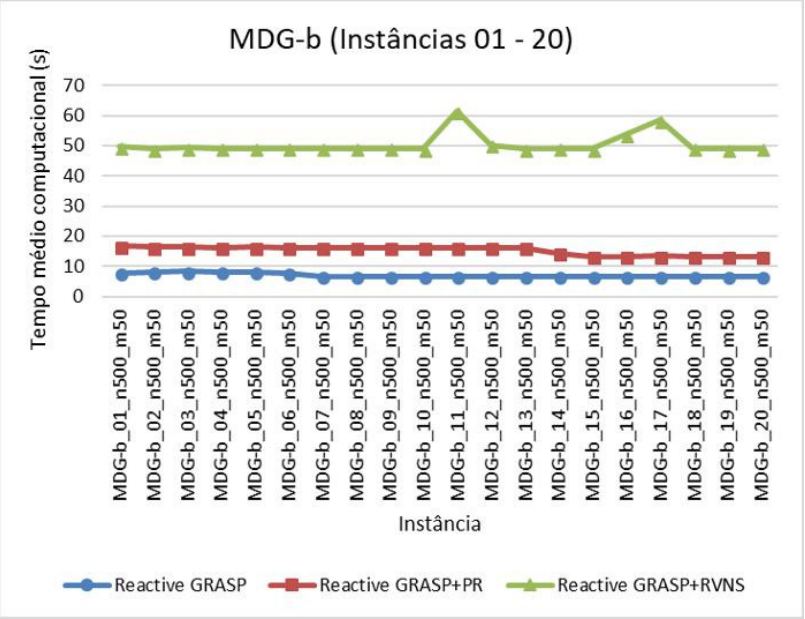

Figura 3: Tempo médio computacional para o MDG-b.

Também na Tabela 5 é apresentada a comparação das soluções obtidas para as maiores instâncias do dataset MDG-a (Instâncias 21 - 25). Como nas instâncias anteriores para esse mesmo dataset, os resultados obtidos pelos três métodos também se aproximaram em mais de 98% dos melhores valores encontrados na literatura.

Na Figura 4 comparam-se os tempos médios computacionais ao se utilizar os três métodos. Como nas demais comparações realizadas até o momento, o *reactive* GRASP com RVNS apresenta maior tempo médio computacional para todas as instâncias. Com o aumento do tamanho das

Tabela 4: Comparação dos resultados do MDG-b

Instância	Melhores valores	*Reactive* GRASP (Porcent.)	*Reactive* GRASP + PR ((Porcent.)	*Reactive* GRASP + RVNS (Porcent.)
MDG-b_01_n500_m50	778030,625	769036,76 (98,84%)	770191,89 (98,99%)	**772268,69 (99,26%)**
MDG-b_02_n500_m50	779963,6875	767864,83 (98,45%)	**770296,44 (98,76%)**	768037,73 (98,47%)
MDG-b_03_n500_m50	776768,4375	767627,43 (98,82%)	768272,76 (98,91%)	**769364,73 (99,05%)**
MDG-b_04_n500_m50	775394,625	768511,1 (99,11%)	766318,04 (98,83%)	769311,99 (99,22%)
MDG-b_05_n500_m50	775611,0625	767367,77 (98,94%)	**770372,32 (99,32%)**	768375,7 (99,07%)
MDG-b_06_n500_m50	775153,6875	765347,49 (98,73%)	765696,58 (98,78%)	**767098,4 (98,96%)**
MDG-b_07_n500_m50	777232,875	**771309,79 (99,24%)**	770910,55 (99,19%)	770054,05 (99,08%)
MDG-b_08_n500_m50	779168,75	**769727,43 (98,79%)**	769492,8 (98,76%)	769699,29 (98,78%)
MDG-b_09_n500_m50	774802,1875	767644,54 (99,08%)	768118,56 (99,14%)	**768204,02 (99,15%)**
MDG-b_10_n500_m50	774961,3125	**767340,16 (99,02%)**	766583,72 (98,92%)	765131,49 (98,73%)
MDG-b_11_n500_m50	777468,875	769385,76 (98,96%)	768716,45 (98,87%)	**769778,88 (99,01%)**
MDG-b_12_n500_m50	775492,9375	**768919,32 (99,15%)**	765053,66 (98,65%)	767475,43 (98,97%)
MDG-b_13_n500_m50	780191,875	768552,05 (98,51%)	**769967,13 (98,69%)**	767781,34 (98,41%)
MDG-b_14_n500_m50	782232,75	771191,4 (98,59%)	771396,74 (98,61%)	**772095,57 (98,70%)**
MDG-b_15_n500_m50	780300,375	**774596,21 (99,27%)**	771471,42 (98,87%)	772861,69 (99,05%)
MDG-b_16_n500_m50	775436,25	768405,5 (99,09%)	**772900,4 (99,67%)**	768142,55 (99,06%)
MDG-b_17_n500_m50	776619,125	769839,63 (99,13%)	769171,6 (99,04%)	**770748,58 (99,24%)**
MDG-b_18_n500_m50	775850,75	769385,18 (99,17%)	**770467,55 (99,31%)**	769125,48 (99,13%)
MDG-b_19_n500_m50	778802,9375	**773021,69 (99,26%)**	772144,87 (99,15%)	771176,77 (99,02%)
MDG-b_20_n500_m50	778644,8125	767109,23 (98,52%)	768539,32 (98,70%)	**768800,31 (98,74%)**

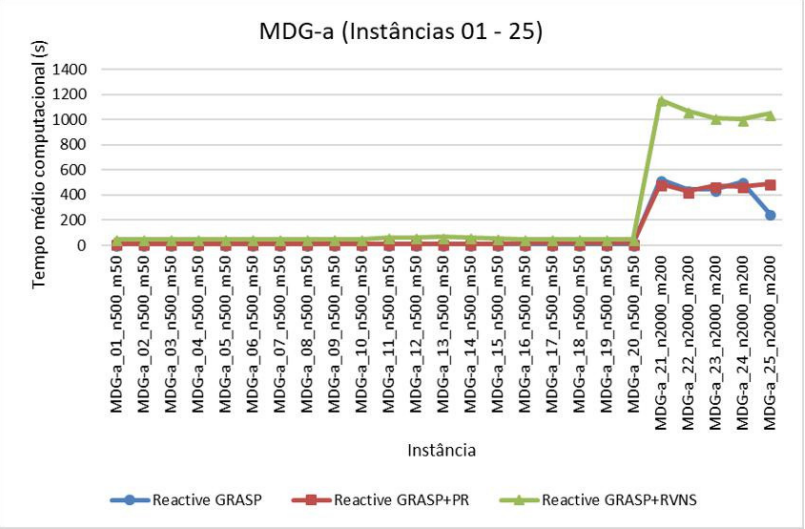

Figura 4: Tempo médio computacional para o MDG-a.

instâncias a diferença no tempo computacional aumenta consideravelmente. Como exemplo para a instância MDG-a_20 o *reactive* GRASP leva em média 8,4556 s, já o *reactive* GRASP + PR leva 16,1782 s e o *reactive* GRASP + RVNS leva 49,0725 s. Para a instância MDG-a_25 o *reactive* GRASP leva em média 255,474 s, já o *reactive* GRASP + PR leva 492,8876 s e o *reactive* GRASP + RVNS leva 1047,7322 s.

Pelos dados obtidos nessa grande quantidade de testes realizados e apresentados nos gráficos, percebe-se que em geral os resultados encontrados são muito próximos para todas as 3 versões do *Reactive* GRASP analisadas. Vale ressaltar que esses resultados estão muito próximos dos melhores resultados encontrados na literatura.

Entretanto, em termos de tempo computacional, o *Reactive* GRASP + RVNS demanda um

Tabela 5: Comparação dos resultados do MDG-a

Instância	Melhores valores	*Reactive* GRASP (Porcent.)	*Reactive* GRASP + PR ((Porcent.)	*Reactive* GRASP + RVNS (Porcent.)
MDG-a_01_n500_m50	7833,83252	**7801,45 (99,59%)**	7776,25 (99,26%)	7770,73 (99,19%)
MDG-a_02_n500_m50	7771,66162	7694,3 (99,00%)	7690,06 (98,95%)	**7725,21 (99,40%)**
MDG-a_03_n500_m50	7759,35986	76(99,68 (99,23%)	**7721,54 (99,51%)**	7718,5 (99,47%)
MDG-a_04_n500_m50	7770,2417	7723,63 (99,40%)	**7713 (99,26%)**	7706,28 (99,18%)
MDG-a_05_n500_m50	7755,23096	7665 (98,84%)	**7672,57 (98,93%)**	7688,94 (99,15%)
MDG-a_06_n500_m50	7773,70996	7680,65 (98,80%)	**7723,93 (99,36%)**	7684,22 (98,85%)
MDG-a_07_n500_m50	7771,73096	7669,18 (98,68%)	7669,13 (98,68%)	**7684,33 (98,88%)**
MDG-a_08_n500_m50	7750,88135	7658,95 (98,81%)	**7679,53 (99,08%)**	7679,1 (99,07%)
MDG-a_09_n500_m50	7770,0708	7681,32 (98,86%)	**7697,44 (99,07%)**	7688,61 (98,95%)
MDG-a_10_n500_m50	7780,35059	**7707,65 (99,07%)**	7696,15 (98,92%)	7699,44 (98,96%)
MDG-a_11_n500_m50	7770,95068	7716,26 (99,30%)	7718,73 (99,33%)	**7738,31 (99,58%)**
MDG-a_12_n500_m50	7757,65039	7657,55 (98,71%)	7658,08 (98,72%)	**7682,03 (99,03%)**
MDG-a_13_n500_m50	7798,43164	7689,25 (98,60%)	**7717,02 (98,96%)**	7697,44 (98,70%)
MDG-a_14_n500_m50	7795,63135	7710,73 (98,91%)	7702,01 (98,80%)	**7730,88 (99,17%)**
MDG-a_15_n500_m50	7736,84033	**7668,59 (99,12%)**	7654,32 (98,93%)	7665,14 (99,07%)
MDG-a_16_n500_m50	7792,77197	**7726,37 (99,15%)**	7701,81 (98,83%)	7702,54 (98,84%)
MDG-a_17_n500_m50	7787,20068	**7712,94 (99,05%)**	7693,75 (98,80%)	7689,26 (98,74%)
MDG-a_18_n500_m50	7756,26172	7669,47 (98,88%)	**7670,66 (98,90%)**	7670,27 (98,89%)
MDG-a_19_n500_m50	7755,41064	**7687,64 (99,13%)**	7672,35 (98,93%)	7671,83 (98,92%)
MDG-a_20_n500_m50	7733,86133	7648,35 (98,89%)	7648,29 (98,89%)	**7656,38 (99,00%)**
MDG-a_21_n2000_m200	114259	112930 (98,84%)	112905 (98,81%)	**112939 (98,84%)**
MDG-a_22_n2000_m200	114327	112834 (98,69%)	**112941 (98,79%)**	112851 (98,71%)
MDG-a_23_n2000_m200	114123	112899 (98,93%)	112922 (98,95%)	**112961 (98,98%)**
MDG-a_24_n2000_m200	114040	**112874 (98,98%)**	112845 (98,95%)	112865 (98,97%)
MDG-a_25_n2000_m200	114064	112923 (99,00%)	112974 (99,04%)	**113012 (99,08%)**

tempo considerável quando comparado aos demais métodos. Para certas instâncias esse operador torna o método 8 vezes mais lento quando comparado ao *Reactive* GRASP padrão.

Voltando à análise ao *Reactive* GRASP + PR, é também possível constatar que o operador, em certas instâncias, chega a duplicar o tempo de processamento requerido pelo *Reactive* GRASP padrão, sem no entanto melhorar significativamente as soluções encontradas.

Do exposto, é possível inferir que, ao menos para as instâncias aqui estudadas, que cobrem um grande conjunto de *benchmarks* para o problema da diversidade máxima, a utilização dos operadores RVNS e PR no *Reactive* GRASP não resultou em melhoria significativa das soluções obtidas, tendo, por outro lado, incrementado de forma evidente o tempo computacional de processamento.

5. Conclusão

Neste artigo foram aplicados os algoritmos *reactive* GRASP, *reactive* GRASP com PR e *reactive* GRASP com RVNS para o problema da diversidade máxima considerando o *benchmark* MDPLIB. O estudo realizado objetivou fazer uma comparação entre a qualidade das soluções obtidas e o custo computacional do processamento desses métodos.

Apesar dos resultados obtidos pelos três métodos terem sido bem próximos, a utilização de PR e RVNS aumentou significativamente o tempo de processamento dos métodos. Dessa forma, para o problema e instâncias aqui estudadas, a utilização desses operadores alternativos não trouxe benefícios para o GRASP em si.

A área de estudos sobre metaheurísticas é bastante dependente de muitos fatores, como o problema que se estuda, as instâncias testadas, os métodos utilizados, dentre outros, tornando praticamente impossível que uma conclusão tomada possa ser estendida para qualquer outra situação. Apesar disso, o presente artigo serve como evidência de que é salutar sempre testar operadores, métodos e outros, e verificar se eles agregam de fato melhorias para o problema em que se está trabalhando.

Quanto aos trabalhos futuros, é possível realizar testes para mais instâncias do *benchmark* MDPLIB, combinar o *Reactive* GRASP com PR e RVNS, utilizar outros operadores, ou mesmo realizar estudos de comparações de operadores alternativos com outras metaheurísticas e problemas. Pode-se também fazer uma análise estatística para a escolha dos valores dos parâmetros de forma mais adequada do que realizar apenas alguns testes empíricos.

Referências

Agrafiotis, D. K. (1997). Stochastic algorithms for maximizing molecular diversity. *J. Chem. Inf. Comput. Sci*, 37(5):841–851.

Andrade, M. R. Q., Andrade, P. M. F., Martins, S. L., e Plastino, A. (2005). GRASP with path-relinking for the maximum diversity problem. In *Experimental and Efficient Algorithms*, p. 558–569. Springer Berlin Heidelberg.

Beasley, J. E. (1990). OR-LIBRARY: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Feo, T. e Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Festa, P., Pardalos, P., e Resende, M. (2002). Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, 17(06):1033–1058.

Hansen, P. e Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Kochenberger, G. e Glover, F. (1999). Diversity data mining. Working paper.

Kuo, C.-C., Glover, F., e Dhir, K. S. (1993). Analyzing and modeling the maximum diversity problem by zero-one programming*. *Decision Sciences*, 24(6):1171–1185.

Luke, S. (2013). *Essentials of Metaheuristics*. Lulu, second edition. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

Marti, R., Gallego, M., Duarte, A., e G. Pardo, E. (2013). Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics - HEURISTICS*, 19:591–615.

Moro, M. A. (2017). Meta-heurísticas GRASP e BRKGA aplicadas ao problema da diversidade máxima (dissertação de mestrado). Universidade Estadual de Campinas, Campinas - SP.

Prais, M. e Ribeiro, C. C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176.

Resende, M., Ribeiro, C., Glover, F., e Marti, R. (2010). Scatter search and path-relinking: Fundamentals, advances, and applications. In *Handbook of Metaheuristics*, p. 87–107.