A study of heuristic algorithms for diversity optimization

Tailini Schultz, **Clara dos Santos Becker**, **Marcelo de Souza**

Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina – Udesc Alto Vale

{tailini.schultz,clara.becker}@edu.udesc.br, marcelo.desouza@udesc.br

Abstract. *Designing metaheuristics involves selecting and combining various heuristic components and tuning their parameters to optimize performance. Understanding the contribution of these components and parameters is crucial for algorithm effectiveness. This paper presents an analysis of the GRASP metaheuristic applied to the maximum diversity problem. We propose constructive and local search heuristics to generate different configurations of the algorithm. Additionally, we perform automatic configuration of the GRASP algorithm to improve its performance. These configurations are evaluated experimentally and through ablation analysis, allowing us to identify the most important components and their specific contributions to enhancing the quality of the solutions.*

Resumo. *O projeto de metaheurísticas envolve selecionar e combinar componentes heurísticos, bem como ajustar seus parâmetros para otimizar soluções. Portanto, é importante entender a contribuição dos componentes e parâmetros para o desempenho do algoritmo. Este trabalho analisa a metaheurística GRASP aplicada ao problema da diversidade máxima, propondo heurísticas construtivas e de busca local para criar diversas configurações do algoritmo. Além disso, é feita a configuração automática do GRASP para melhorar seu desempenho. Essas configurações são avaliadas experimentalmente e usando a técnica de ablation analysis, permitindo identificar os componentes mais importantes e suas contribuições para a qualidade das soluções.*

1. Introdução

Os componentes de uma metaheurística, como heurísticas construtivas, vizinhanças ou estratégias de recombinação de soluções, definem a forma como o espaço de soluções é explorado. Avaliar a contribuição desses componentes no desempenho observado do algoritmo é fundamental, não só para entender sua importância, mas para auxiliar na tomada de decisões durante o projeto do algoritmo e, por consequência, otimizar seu desempenho e a qualidade das soluções produzidas.

Este trabalho apresenta uma análise de componentes da metaheurística GRASP (Feo e Resende, 1995) aplicada ao problema da diversidade máxima (Kuby, 1987). São definidas algumas heurísticas construtivas e estratégias de seleção de vizinhos para a busca local. Diferentes combinações desses componentes são avaliados experimentalmente, identificando a contribuição de cada um na qualidade das soluções encontradas. Além disso, são aplicadas técnicas de configuração automática de algoritmos e *ablation analysis*, que permitem encontrar configurações com desempenho otimizado e avaliar a

importância dos seus componentes de maneira automatizada, minimizando o esforço humano empregado nessas tarefas.

A literatura apresenta diferentes abordagens para análise de componentes de metaheurísticas. Villagra et al. (2015) avaliam a contribuição de componentes removendo-os do algoritmo e medindo a variação no desempenho, além de explorar ferramentas de *profiling* para identificar os componentes responsáveis pelo maior volume de processamento durante sua execução. Pessoa et al. (2015) seguem uma abordagem similar à análise manual proposta, onde diferentes combinações de componentes são avaliadas experimentalmente, medindo a influência de cada um deles no desempenho do algoritmo. Finalmente, alguns trabalhos propõem abordagens sistemáticas para a análise de componentes, como as técnicas de *forward selection* (Hutter et al., 2013), fANOVA (Hutter et al., 2014) e *ablation analysis* (Fawcett e Hoos, 2016). Dang e De Causmaecker (2019) aplicam essas técnicas em três estudos de caso, demonstrando suas características e uso. Através do estudo de caso que explora a solução do problema da diversidade máxima usando o algoritmo GRASP, este trabalho compara os resultados da análise de componentes usando uma abordagem manual com aqueles obtidos usando técnicas automatizadas.

O trabalho está organizado da seguinte forma. A Seção 2 formaliza o problema da diversidade máxima e discute algumas aplicações. A Seção 3 apresenta a metaheurística GRASP, as heurísticas construtivas e estratégias de seleção de vizinhos da busca local. A Seção 4 apresenta a análise de componentes, destacando a contribuição das heurísticas construtivas, das buscas locais, os resultados da configuração automática do algoritmo e da técnica de *ablation analysis*. A Seção 5 conclui o trabalho.

2. Problema da diversidade máxima

O problema da diversidade máxima consiste em selecionar um subconjunto de m elementos a partir de um conjunto de n elementos, de tal maneira que a distância total entre os elementos selecionados seja máxima. A distância entre dois elementos i e j é representada por d_{ij} e definida pela métrica mais apropriada conforme o domínio de aplicação do problema (Martí e Martínez-Gavara, 2023). Uma abordagem comum é associar um conjunto A de atributos aos elementos, onde os valores de cada atributo $a \in A$ em cada elemento $i \in [n]$ são definidos por a_i, e então usar alguma métrica de distância em função das características dos elementos. Um exemplo é a distância Euclidiana, definida por

$$d_{ij} = \sqrt{\sum_{a \in A} (a_i - a_j)^2}.$$

O problema da diversidade máxima pode ser formulado matematicamente como o programa quadrático binário apresentado pelas Equações (1 – 3). A variável de decisão binária x_i assume o valor 1 quando o elemento é selecionado, e 0 caso contrário. A função objetivo (1) maximiza o somatório das distâncias entre os elementos selecionados. A restrição (2) garante que exatamente m elementos sejam selecionados.

$$\text{Maximiza} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j, \tag{1}$$

$$\text{sujeito a} \quad \sum_{i=1}^{n} x_i = m, \tag{2}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in [n]. \tag{3}$$

O problema da diversidade máxima possui aplicações em diferentes áreas, como na preservação da diversidade biológica de animais (Glover et al., 1995). Os autores propõem uma abordagem para medir a diversidade genética entre diferentes espécies e aplicam modelos de programação matemática para seleção de indivíduos e maximização da diversidade genética. Exemplos de outros domínios de aplicação incluem o gerenciamento de engenharia (Dhir et al., 1993), localização de instalações (Rosenkrantz et al., 2000), projeto de produtos (Glover et al., 1998) e tratamento médico (Kuo et al., 1993).

3. Algoritmos explorados

Este trabalho estuda diferentes componentes da metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*), proposta por Feo e Resende (1995) e apresentada pelo Algoritmo 1. Em cada iteração do GRASP, duas fases são realizadas: a geração de uma solução através de uma heurística construtiva semi-gulosa (linha 4) e, em seguida, o refinamento dessa solução por meio de um procedimento de busca local (linha 5). Assim, a cada iteração uma nova solução é gerada e melhorada, atualizando a solução incumbente quando apropriado (linhas 6 e 7), a qual é retornada quando um número predeterminado de iterações é atingido.

```
 1: procedimento GRASP(M, α, β)
 2:     S_best ← ∅
 3:     para i = 1 até M faça
 4:         S ← CONSTRUÇÃO(α, β)
 5:         S ← BUSCALOCAL(S)
 6:         se S é melhor que S_best então
 7:             S_best ← S
 8:         fim se
 9:     fim para
10:     retorna S_best
11: fim procedimento
```

Algoritmo 1. Pseudocódigo do algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*).

Para implementação do GRASP, é necessário definir as heurísticas para as fases de construção e busca local. São propostas duas heurísticas construtivas semi-gulosas, que consideram os elementos de maior *Distância da Solução Parcial* (DSP) e maior *Distância da Solução Esperada* (DSE). Essas heurísticas construtivas iniciam a partir de uma solução vazia, i.e. onde nenhum elemento foi selecionado. A cada iteração, um

elemento ainda não selecionado é escolhido e inserido na solução parcial. O processo se repete até que a solução esteja completa, i.e. m elementos sejam selecionados. Os detalhes de funcionamento de cada heurística são apresentados a seguir.

Distância da Solução Parcial (DSP). Essa heurística escolhe os elementos em função da sua distância com aqueles já selecionados. Dado um parâmetro real $\alpha \in [0, 1]$, a heurística escolhe aleatoriamente um dos elementos cuja distância com os elementos já selecionados seja maior ou igual ao limiar $g_{\text{max}} - \alpha \cdot (g_{\text{max}} - g_{\text{min}})$. Por exemplo, para $\alpha = 0.5$ e $n = 100$ elementos, dos quais 10 já foram selecionados, os 90 elementos restantes são avaliados conforme sua distância total com os 10 elementos da solução parcial. Se nessa iteração $g_{\text{min}} = 20$ e $g_{\text{max}} = 80$, a heurística escolherá aleatoriamente um entre os elementos com distância maior ou igual a $80 - 0.5 \cdot 60 = 50$. Para a seleção do primeiro elemento, i.e. quando a solução parcial é vazia, a mesma estratégia é adotada, mas considerando a distância de cada elemento com todos os demais. Em alguns casos, é desejado que a estratégia de seleção do primeiro elemento seja distinta das demais iterações (e.g. seleção aleatória). Por isso, na primeira iteração a escolha é feita considerando o parâmetro real $\beta \in [0, 1]$, no lugar no parâmetro α, permitindo estratégias de seleção diferentes para o primeiro e demais elementos.

Distância da Solução Esperada (DSE). O mesmo funcionamento geral da heurística DSP é adotado, incluindo as estratégias de seleção em função dos parâmetros α e β. A diferença da heurística DSE é a distância usada para avaliar os elementos não selecionados. Além da distância total entre o elemento e a solução parcial, é somada a distância total entre o elemento e seus $m-|S|-1$ elementos ainda não selecionados mais distantes, sendo $|S|$ o tamanho da solução parcial S, dado pelo número de elementos já selecionados. Em outras palavras, essa heurística considera tanto a distância dos elementos para a solução parcial, quanto sua diversidade em relação aos elementos ainda não selecionados, limitada ao número de elementos restantes para que a solução seja completa.

É importante destacar que as heurísticas construtivas apresentadas podem ser facilmente reduzidas a heurísticas gulosas, quando $\alpha = \beta = 0$, ou a uma construção aleatória, quando $\alpha = \beta = 1$. Para valores de $\alpha, \beta \in (0, 1)$, essas heurísticas realizam uma construção semi-gulosa, onde valores mais próximos de 0 tornam o algoritmo mais guloso, enquanto valores mais próximos de 1 tornam o algoritmo mais aleatório.

Após a construção, a solução é submetida à fase de refinamento, onde é executada uma busca local simples. Essa heurística considera a vizinhança *swap* (Ghosh, 1996), que consiste em substituir um elemento da solução por outro não selecionado. Para $\{i, j\}, i \in S, j \notin S$ o ganho na função objetivo por substituir i por j na solução é definido como $\Delta_{ij} = \sum_{k \in S \setminus \{i\}} d_{jk} - d_{ik}$. Se para todo $\{i, j\}$, $\Delta_{ij} < 0$, a solução S é um ótimo local e a busca local termina, retornando S. Caso contrário, algum par $\{i, j\}$ (chamado vizinho) com melhoria da função objetivo é selecionado, a troca dos elementos é efetivada na solução e a busca continua a partir da solução vizinha gerada. São exploradas três estratégias para seleção de vizinho: *Melhor Melhoria* (MM), *Primeira Melhoria* (PM) e *Melhoria Aleatória* (MA).

Melhor Melhoria (MM). Seleciona o vizinho com maior ganho na função objetivo, i.e. $\arg\max_{i,j} \Delta_{ij}$.

Tabela 1. Componentes e parâmetros implementados pelo algoritmo GRASP.

Componente/parâmetro	Valores
CONSTRUÇÃO	{DSP, DSE}
BUSCALOCAL	{MM, PM, MA}
α	$[0, 1]$
β	$[0, 1]$

Primeira Melhoria (PM). Os elementos são processados em ordem crescente, tanto os elementos candidatos a serem substituídos, quanto os candidatos a substitutos. O primeiro par encontrado cuja substituição melhora o valor da função objetivo (i.e. $\Delta > 0$) é escolhido.

Melhoria Aleatória (MA). Aplica a mesma estratégia da PM, mas processa os elementos em ordem aleatória. Por consequência, é retornado aleatoriamente um entre os vizinhos que melhoram o valor da função objetivo.

A Tabela 1 resume os componentes e parâmetros estudados neste trabalho. São apresentadas as heurísticas construtivas e estratégias de seleção de vizinhos das buscas locais, bem como os parâmetros de gulosidade da fase de construção.

4. Análise de componentes

Esta seção apresenta a análise de componentes do algoritmo GRASP para o problema da diversidade máxima. Conforme apresentado pela Tabela 1, foram implementados componentes para as fases de construção e busca local, além dos parâmetros que definem o nível de aleatoriedade das heurísticas construtivas. A análise proposta se divide em três etapas: (1) a contribuição das heurísticas construtivas; (2) a contribuição das buscas locais, combinadas com as heurísticas construtivas; (3) a configuração automática do algoritmo, incluindo a determinação dos valores para os parâmetros. Nas etapas 1 e 2, serão executados experimentos comparando diferentes versões do algoritmo GRASP, i.e. usando apenas construção aleatória, explorando as heurísticas construtivas e as combinando com as buscas locais. Na etapa 3, o configurador irace (López-Ibáñez et al., 2016) é usado para buscas a melhor configuração do GRASP, i.e. a melhor combinação de componentes e valores para seus parâmetros. O algoritmo resultante é comparado com o melhor algoritmo encontrado nas etapas anteriores. Finalmente, é realizada uma análise da contribuição de cada componente através da técnica de *ablation analysis* (Fawcett e Hoos, 2016).

Foi usado um conjunto de 50 instâncias geradas por Martí et al. (2010) usando o gerador desenvolvido por Silva et al. (2004). Essas instâncias possuem tamanhos $n \in \{25, 50, 100, 125, 150\}$ e $m \in \{2, 5, 7, 10, 12, 15, 30, 37, 45\}$, e valores de diversidade inteiros no intervalo $[0, 9]$. Foram selecionadas aleatoriamente 25 instâncias para avaliação experimental dos algoritmos (as mesmas instâncias foram usadas para comparar os algoritmos em todas as etapas da análise). As outras 25 instâncias foram usadas exclusivamente para o processo de configuração automática do algoritmo (etapa 3). Com isso, se garante que os desempenhos observados sejam comparáveis em todas as etapas dos experimentos, e que a configuração automática do algoritmo segue as boas práticas de

Tabela 2. Desempenho médio da construção aleatória e das heurísticas construtivas.

Instância			Aleatória	DSP [%]		DSE [%]	
#	n	m					
1	25	2	9.0	9.0	[0.0]	9.0	[0.0]
3	25	2	9.0	9.0	[0.0]	9.0	[0.0]
5	25	2	9.0	9.0	[0.0]	9.0	[0.0]
6	25	7	197.0	189.1	[-4.0]	147.5	[-25.1]
9	25	7	196.0	187.4	[-4.4]	143.9	[-26.6]
13	50	5	84.0	101.3	[20.6]	74.8	[-11.0]
15	50	5	84.0	100.7	[19.9]	75.7	[-9.9]
16	50	15	739.0	699.9	[-5.3]	615.5	[-16.7]
18	50	15	727.0	694.6	[-4.5]	598.1	[-17.7]
21	100	10	398.0	335.2	[-15.8]	284.0	[-28.6]
22	100	10	400.0	341.6	[-14.6]	286.9	[-28.3]
25	100	10	384.0	333.7	[-13.1]	283.5	[-26.2]
26	100	30	2647.4	2505.4	[-5.4]	2337.1	[-11.7]
29	100	30	2651.0	2504.3	[-5.5]	2347.1	[-11.5]
32	125	12	546.0	452.4	[-17.1]	401.9	[-26.4]
33	125	12	546.0	458.8	[-16.0]	397.5	[-27.2]
35	125	12	553.0	462.3	[-16.4]	405.9	[-26.6]
36	125	37	3934.5	3716.0	[-5.6]	3508.1	[-10.8]
39	125	37	3881.0	3658.8	[-5.7]	3481.8	[-10.3]
40	125	37	3896.0	3683.5	[-5.5]	3487.6	[-10.5]
41	150	15	832.0	703.2	[-15.5]	622.6	[-25.2]
43	150	15	815.0	695.2	[-14.7]	621.2	[-23.8]
45	150	15	814.0	690.8	[-15.1]	605.9	[-25.6]
47	150	45	5593.4	5314.6	[-5.0]	5084.0	[-9.1]
49	150	45	5592.1	5298.7	[-5.2]	5060.8	[-9.5]
Melhoria média [%]			–		-6.2		-16.7

usar conjuntos de instâncias independentes para a configuração do algoritmo e avaliação das configurações produzidas.

Em todos os experimentos, o GRASP foi executado 10 vezes com 200 iterações (i.e. $M = 200$) em uma máquina com processador AMD Ryzen 3 5300U, com 4 núcleos operando a 3.8 GHz, e executando sistema operacional Linux. Cada experimento foi executado em somente um núcleo de processamento. Os resultados apresentados nesta seção são médias das 10 execuções de cada algoritmo.

4.1. Heurísticas construtivas

Nesta etapa, foram avaliadas três versões do GRASP usando somente a fase de construção, i.e. sem busca local. A primeira versão usa construção aleatória, enquanto as outras usam as heurísticas construtivas DSP e DSE, ambas com valores fixos de $\alpha = 0.5$ e $\beta = 0.3$. A Tabela 2 apresenta os resultados das três versões do GRASP, incluindo o valor médio da melhor solução obtida pelo GRASP nas 10 replicações, e a variação (ou

Tabela 3. Desempenho médio das buscas locais combinadas com a heurística construtiva DSP.

Instância			DSP	DSP_{PM} [%]		DSP_{MM} [%]		DSP_{MA} [%]	
#	n	m							
1	25	2	9.0	9.0	[0.0]	9.0	[0.0]	9.0	[0.0]
3	25	2	9.0	9.0	[0.0]	9.0	[0.0]	9.0	[0.0]
5	25	2	9.0	9.0	[0.0]	9.0	[0.0]	9.0	[0.0]
6	25	7	189.1	204.3	[8.0]	204.3	[8.0]	204.0	[7.9]
9	25	7	187.4	198.9	[6.1]	198.3	[5.8]	198.3	[5.8]
13	50	5	101.3	115.2	[13.7]	116.1	[14.6]	115.7	[14.2]
15	50	5	100.7	114.7	[13.9]	114.9	[14.1]	116.3	[15.5]
16	50	15	699.9	751.1	[7.3]	749.5	[7.1]	750.5	[7.2]
18	50	15	694.6	736.4	[6.0]	736.9	[6.1]	739.1	[6.4]
21	100	10	335.2	397.4	[18.6]	393.1	[17.3]	394.5	[17.7]
22	100	10	341.6	395.5	[15.8]	397.5	[16.4]	396.3	[16.0]
25	100	10	333.7	394.3	[18.2]	397.0	[19.0]	398.1	[19.3]
26	100	30	2505.4	2655.4	[6.0]	2653.9	[5.9]	2655.0	[6.0]
29	100	30	2504.3	2673.2	[6.7]	2677.7	[6.9]	2676.4	[6.9]
32	125	12	452.4	546.5	[20.8]	546.7	[20.8]	550.1	[21.6]
33	125	12	458.8	544.4	[18.7]	543.1	[18.4]	540.4	[17.8]
35	125	12	462.3	546.9	[18.3]	545.1	[17.9]	547.7	[18.5]
36	125	37	3716.0	3952.4	[6.4]	3945.2	[6.2]	3949.4	[6.3]
39	125	37	3658.8	3937.5	[7.6]	3935.7	[7.6]	3936.5	[7.6]
40	125	37	3683.5	3923.9	[6.5]	3924.0	[6.5]	3925.9	[6.6]
41	150	15	703.2	818.3	[16.4]	818.5	[16.4]	820.7	[16.7]
43	150	15	695.2	821.3	[18.1]	823.0	[18.4]	826.1	[18.8]
45	150	15	690.8	810.4	[17.3]	802.8	[16.2]	804.3	[16.4]
47	150	45	5314.6	5637.6	[6.1]	5635.3	[6.0]	5642.8	[6.2]
49	150	45	5298.7	5602.8	[5.7]	5600.4	[5.7]	5610.7	[5.9]
Melhoria média [%]			–		10.5		10.5		10.6

melhoria) percentual observada usando as heurísticas construtivas, em comparação com a construção aleatória. São apresentados os resultados para as 25 instâncias usadas para avaliação dos algoritmos, contendo o número da instância (coluna "#") e os valores de n e m.

Pode-se observar que o GRASP usando a construção aleatória possui melhor desempenho, em comparação com as versões que usam as heurísticas construtivas. Analisando o comportamento desses algoritmos, são identificadas duas razões para esse resultado. Por um lado, a construção aleatória produz soluções com grande diversidade, conforme esperado. Para as instâncias consideradas, esse algoritmo é capaz de produzir soluções razoáveis pelo menos alguma vez em 200 iterações. Por outro lado, as heurísticas DSP e DSE não são capazes de produzir soluções diversificadas, especialmente por conta da seleção (muito gulosa) do primeiro elemento, estratégia controlada pelo parâmetro β, cujo valor poderia ser maior para aumentar a aleatoriedade desse passo da construção.

Tabela 4. Desempenho médio das buscas locais combinadas com a heurística construtiva DSE.

Instância			DSE	DSE_{PM} [%]		DSE_{MM} [%]		DSE_{MA} [%]	
#	n	m							
1	25	2	9.0	9.0	[0.0]	9.0	[0.0]	9.0	[0.0]
3	25	2	9.0	9.0	[0.0]	9.0	[0.0]	9.0	[0.0]
5	25	2	9.0	9.0	[0.0]	9.0	[0.0]	9.0	[0.0]
6	25	7	147.5	153.0	[3.7]	153.0	[3.7]	153.0	[3.7]
9	25	7	143.9	147.0	[2.2]	147.0	[2.2]	147.0	[2.2]
13	50	5	74.8	85.0	[13.6]	85.0	[13.6]	85.0	[13.6]
15	50	5	75.7	84.0	[11.0]	84.0	[11.0]	84.0	[11.0]
16	50	15	615.5	646.0	[5.0]	646.0	[5.0]	646.0	[5.0]
18	50	15	598.1	638.0	[6.7]	638.0	[6.7]	638.0	[6.7]
21	100	10	284.0	331.0	[16.5]	331.0	[16.5]	331.0	[16.5]
22	100	10	286.9	333.0	[16.1]	333.0	[16.1]	333.0	[16.1]
25	100	10	283.5	337.0	[18.9]	337.0	[18.9]	337.0	[18.9]
26	100	30	2337.1	2461.0	[5.3]	2461.0	[5.3]	2461.0	[5.3]
29	100	30	2347.1	2486.0	[5.9]	2486.0	[5.9]	2486.0	[5.9]
32	125	12	401.9	474.0	[17.9]	474.0	[17.9]	474.0	[17.9]
33	125	12	397.5	470.0	[18.2]	470.0	[18.2]	470.0	[18.2]
35	125	12	405.9	474.0	[16.8]	474.0	[16.8]	474.0	[16.8]
36	125	37	3508.1	3716.0	[5.9]	3716.0	[5.9]	3716.0	[5.9]
39	125	37	3481.8	3710.0	[6.6]	3710.0	[6.6]	3710.0	[6.6]
40	125	37	3487.6	3695.0	[5.9]	3695.0	[5.9]	3695.0	[5.9]
41	150	15	622.6	719.0	[15.5]	719.0	[15.5]	719.0	[15.5]
43	150	15	621.2	732.0	[17.8]	732.0	[17.8]	732.0	[17.8]
45	150	15	605.9	721.0	[19.0]	721.0	[19.0]	721.0	[19.0]
47	150	45	5084.0	5371.0	[5.6]	5370.5	[5.6]	5371.0	[5.6]
49	150	45	5060.8	5344.5	[5.6]	5344.7	[5.6]	5345.0	[5.6]
Melhoria média [%]			–		9.6		9.6		9.6

4.2. Buscas locais

Foram testadas outras seis versões do GRASP, combinando as heurísticas construtivas DSP e DSE com as buscas locais usando as estratégias PM, MM e MA. A Tabela 3 mostra os resultados das versões do GRASP usando a construção DSP e sua combinação com das diferentes buscas locais. A Tabela 4 apresenta os mesmos resultados, mas considerando a construção DSE.

Como pode-se observar nas Tabelas 3 e 4, as buscas locais contribuem para um melhor desempenho do algoritmo GRASP. Para todas as instâncias avaliadas, os resultados da versão do GRASP usando busca local foram iguais ou superiores àqueles obtidos com a versão somente construtiva. Desconsiderando as instâncias menores (1, 2 e 5), onde os algoritmos têm o mesmo desempenho, a contribuição das buscas locais combinadas com a construção DSP varia de 5.7% a 21.6%. Quando combinadas com a construção DSE, a contribuição das buscas locais é um pouco menor, variando de 2.2% a 19%. Como pode-se observar nos valores de melhoria médios, não há diferença no desempenho das

Tabela 5. Desempenho médio do melhor algoritmo avaliado e do algoritmo configurado automaticamente.

Instância			Aleatória	DSP_{MA} [%]		AAC [%]	
#	n	m					
1	25	2	9.0	9.0	[0.0]	9.0	[0.0]
3	25	2	9.0	9.0	[0.0]	9.0	[0.0]
5	25	2	9.0	9.0	[0.0]	9.0	[0.0]
6	25	7	197.0	204.0	[3.6]	207.0	[5.1]
9	25	7	196.0	198.3	[1.2]	199.8	[1.9]
13	50	5	84.0	115.7	[37.7]	116.8	[39.0]
15	50	5	84.0	116.3	[38.5]	118.1	[40.6]
16	50	15	739.0	750.5	[1.6]	758.7	[2.7]
18	50	15	727.0	739.1	[1.7]	745.9	[2.6]
21	100	10	398.0	394.5	[-0.9]	407.5	[2.4]
22	100	10	400.0	396.3	[-0.9]	411.1	[2.8]
25	100	10	384.0	398.1	[3.7]	409.3	[6.6]
26	100	30	2647.4	2655.0	[0.3]	2663.2	[0.6]
29	100	30	2651.0	2676.4	[1.0]	2686.8	[1.4]
32	125	12	546.0	550.1	[0.8]	565.6	[3.6]
33	125	12	546.0	540.4	[-1.0]	560.4	[2.6]
35	125	12	553.0	547.7	[-1.0]	566.6	[2.5]
36	125	37	3934.5	3949.4	[0.4]	3963.6	[0.7]
39	125	37	3881.0	3936.5	[1.4]	3958.9	[2.0]
40	125	37	3896.0	3925.9	[0.8]	3938.9	[1.1]
41	150	15	832.0	820.7	[-1.4]	837.5	[0.7]
43	150	15	815.0	826.1	[1.4]	844.2	[3.6]
45	150	15	814.0	804.3	[-1.2]	833.1	[2.3]
47	150	45	5593.4	5642.8	[0.9]	5659.6	[1.2]
49	150	45	5592.1	5610.7	[0.3]	5631.4	[0.7]
Melhoria média [%]			–		3.6		5.1

diferentes estratégias de busca local. Nas versões do GRASP com construção DSP, a contribuição das três buscas locais é de aproximadamente 10.5% de melhoria na qualidade das soluções. Nas versões do GRASP com construção DSE, a contribuição cai para 9.6%.

4.3. Configuração automática

Nesta etapa, o configurador irace foi usado para buscar a melhor combinação de componentes (heurística construtiva e busca local) e os melhores valores para os parâmetros α e β. O irace foi executado com o máximo de 1000 avaliações (*budget*), usando as 25 instâncias reservadas para essa tarefa. A configuração produzida pelo irace é composta pela construção DSP com busca local PM, $\alpha = 0.08$ e $\beta = 0.46$. Ou seja, trata-se de uma versão do GRASP com construção mais gulosa (pelo menor valor de α), mas com seleção mais aleatória do primeiro elemento (pelo maior valor de β). Essa configuração foi avaliada nas instâncias de teste, permitindo a comparação com os resultados apresentados nas

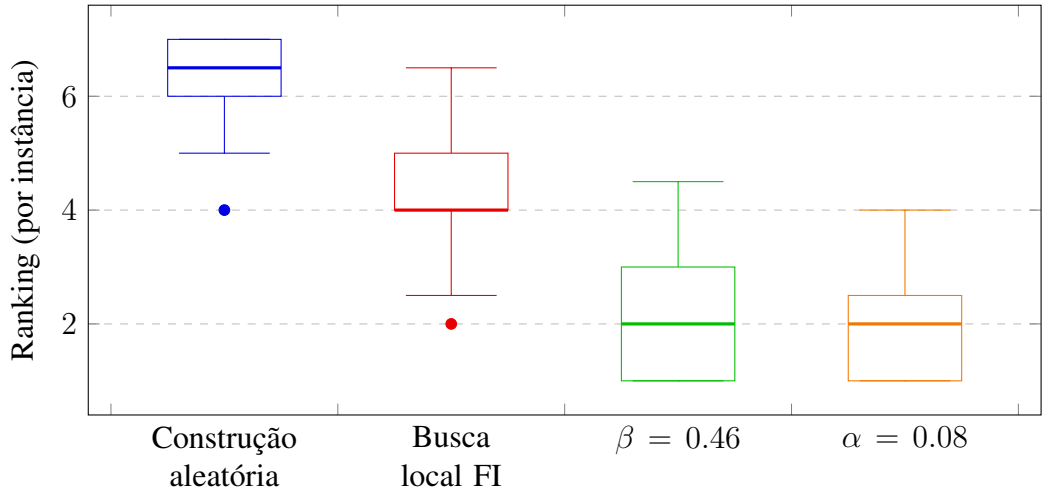

Figura 1. Desempenho da construção aleatória e dos algoritmos gerados pela inclusão de componentes e alteração de valores dos parâmetros; resultados do *ablation analysis*.

seções anteriores.

A Tabela 5 apresenta os resultados da melhor versão do GRASP observada nos experimentos anteriores (DSP_{MA}) e da versão produzida pelo irace (AAC), comparando com o GRASP usando somente construção aleatória. Observa-se que a versão configurada AAC é superior às demais, apresentando uma melhoria média de 5.1% na qualidade das soluções, enquanto o algoritmo DSP_{MA} melhora em média 3.6% a qualidade das soluções. Esses resultados mostram a capacidade da configuração automática de algoritmos em encontrar boas combinações de componentes e valores de parâmetros com pouco esforço humano e computacional (foram realizadas somente 1000 execuções). Além disso, o experimento demonstra a importância de ajustar os parâmetros que controlam a aleatoriedade da fase construtiva para obter o melhor desempenho do algoritmo.

Finalmente, foi realizada uma análise dos componentes do algoritmo GRASP usando a técnica de *ablation analysis* (Fawcett e Hoos, 2016). Essa técnica analisa o trajeto entre uma configuração inicial (e.g. padrão do algoritmo) e uma configuração otimizada, analisando a contribuição de cada alteração na configuração inicial em direção à otimizada. Neste trabalho, consideramos o GRASP com somente construção aleatória como configuração inicial e o GRASP configurado (AAC) como configuração otimizada. O procedimento foi realizado executando as configurações analisadas (i.e. do trajeto explorado) em todo o conjunto de instâncias usadas no processo de configuração automática. Ou seja, não foram usados modelos substitutos (ou *surrogate models*), como explorado em outros trabalhos (Dang e De Causmaecker, 2019).

Os resultados da *ablation analysis* estão resumidos na Figura 1. O gráfico mostra a configuração inicial (GRASP com construção aleatória), seguida dos componentes ou parâmetros mais importantes para a melhoria do desempenho do algoritmo (ordenados pela importância). São apresentados diagramas de caixas dos *rankings* obtidos pelas diferentes configurações desse trajeto nas instâncias usadas na análise (i.e. as 25 instâncias separadas para o processo de configuração).

Pode-se perceber que os resultados observados nos experimentos anteriores são reproduzidos nesta etapa. O componente mais importante é a busca local (neste caso, usando a estratégia PM), que faz com que o algoritmo inicial, ranqueado na 6ª ou 7ª posições para a maioria das instâncias, melhore seu desempenho e seja ranqueado em torno da 4ª ou 5ª posições. Nesta análise, o algoritmo já usa construção DSP, ainda que totalmente aleatória na configuração inicial. Por isso, a próxima melhoria consiste em alterar o valor do parâmetro β, permitindo ajustar a gulosidade da seleção do primeiro elemento. Finalmente, é ajustado o parâmetro α, tornando toda a construção semi-gulosa. Essas duas últimas alterações fazem com que os algoritmos resultantes ocupem as primeiras posições para a maioria das instâncias.

5. Considerações finais

Este trabalho apresenta uma análise de componentes da metaheurística GRASP aplicada ao problema da diversidade máxima. São implementadas diferentes heurísticas construtivas e de busca local, e definidos parâmetros para controlar a aleatoriedade da construção. Diferentes combinações desses componentes são avaliados experimentalmente, identificando a contribuição de cada componente na qualidade das soluções encontradas. Além disso, foi realizada a configuração automática do GRASP, produzindo uma configuração com desempenho otimizado, a qual foi comparada com a melhor configuração observada nos experimentos anteriores. Finalmente, foi aplicada a técnica de *ablation analysis*, medindo a importância de cada componente e seu impacto no desempenho do algoritmo.

A análise mostra que o uso de heurísticas construtivas muito gulosas, especialmente na seleção do primeiro elemento, diminui a diversidade das soluções produzidas, o que piora o desempenho do algoritmo. As buscas locais, por sua vez, contribuem de forma expressiva para a qualidade das soluções encontradas, e se mostraram o componente mais importante. Ao aplicar técnicas de configuração automática de algoritmos e, dessa forma, permitir o ajuste dos parâmetros da fase construtiva, foi possível obter o melhor desempenho entre as versões do GRASP avaliadas. Essas conclusões foram validadas com a aplicação da técnica de *ablation analysis*, através da qual foi possível medir a contribuição de cada componente em termos do *ranking* obtido por cada configuração nas instâncias avaliadas.

Cabe destacar a eficácia e facilidade das abordagens automatizadas exploradas no estudo, i.e. o configurador irace e a técnica de *ablation analysis*. Essas técnicas permitem encontrar boas combinações de componentes e valores de parâmetros com esforço humano mínimo, bem como analisar a importância e contribuição dos componentes sem a necessidade de testar toda possível combinação. Este trabalho demonstra o uso dessas técnicas através do estudo de caso apresentado, bem como as compara com a análise de componentes manual.

Referências

Dang, N. e De Causmaecker, P. (2019). Analysis of algorithm components and parameters: some case studies. Em *Learning and Intelligent Optimization: 12th International Conference, LION 12, Kalamata, Greece, June 10–15, 2018, Revised Selected Papers 12*, páginas 288–303. Springer.

Dhir, K., Glover, F. e Kuo, C.-C. (1993). Optimizing diversity for engineering management. Em *Proceedings of engineering management society conference on managing projects in a borderless world*, páginas 23–26. IEEE.

Fawcett, C. e Hoos, H. H. (2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22:431–458.

Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133.

Ghosh, J. B. (1996). Computational aspects of the maximum diversity problem. *Operations research letters*, 19(4):175–181.

Glover, F., Ching-Chung, K. e Dhir, K. S. (1995). A discrete optimization model for preserving biological diversity. *Applied mathematical modelling*, 19(11):696–701.

Glover, F., Kuo, C.-C. e Dhir, K. S. (1998). Heuristic algorithms for the maximum diversity problem. *Journal of information and Optimization Sciences*, 19(1):109–132.

Hutter, F., Hoos, H. e Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. Em *International conference on machine learning*, páginas 754–762. PMLR.

Hutter, F., Hoos, H. H. e Leyton-Brown, K. (2013). Identifying key algorithm parameters and instance features using forward selection. Em *Learning and Intelligent Optimization: 7th International Conference, LION 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers 7*, páginas 364–381. Springer.

Kuby, M. J. (1987). Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. *Geographical Analysis*, 19(4):315–329.

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.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M. e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Martí, R., Gallego, M. e Duarte, A. (2010). A branch and bound algorithm for the maximum diversity problem. *European journal of operational research*, 200(1):36–44.

Martí, R. e Martínez-Gavara, A. (2023). Discrete diversity and dispersion maximization: A tutorial on metaheuristic optimization.

Pessoa, L. F. D. A., Wagner, C., Hellingrath, B. e Neto, F. B. D. L. (2015). Component analysis based approach to support the design of meta-heuristics for mlclsp providing guidelines. Em *2015 IEEE Symposium Series on Computational Intelligence*, páginas 1029–1038. IEEE.

Rosenkrantz, D. J., Tayi, G. K. e Ravi, S. (2000). Facility dispersion problems under capacity and cost constraints. *Journal of combinatorial optimization*, 4:7–33.

Silva, G. C., Ochi, L. S. e Martins, S. L. (2004). Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. Em *International Workshop on Experimental and Efficient Algorithms*, páginas 498–512. Springer.

Villagra, A., Leguizamón, G. e Alba, E. (2015). Active components of metaheuristics in cellular genetic algorithms. *Soft Computing*, 19:1295–1309.