UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA
E COMPUTAÇÃO CIENTÍFICA

MARCEL ALVES MORO

META-HEURÍSTICAS GRASP E BRKGA APLICADAS AO PROBLEMA DA DIVERSIDADE MÁXIMA

CAMPINAS - SP
2017

Marcel Alves Moro

Meta-heurísticas GRASP e BRKGA aplicadas ao Problema da Diversidade Máxima

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Matemática Aplicada.

Orientadora: Marcia Aparecida Gomes Ruggiero

Coorientadora: Kelly Cristina Poldi

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Marcel Alves Moro, e orientada pela Profa. Dra. Marcia Aparecida Gomes Ruggiero.

Campinas - SP
2017

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Márcia Pillon D'Aloia - CRB 8/5180

M828m

Moro, Marcel Alves, 1992-
Meta-heurísticas GRASP e BRKGA aplicadas ao problema da diversidade máxima / Marcel Alves Moro. – Campinas, SP : [s.n.], 2017.

Orientador: Marcia Aparecida Gomes Ruggiero.
Coorientador: Kelly Cristina Poldi.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Meta-heurística. 2. GRASP (Meta-heurística). 3. Problema de diversidade máxima. 4. Algoritmos genéticos. I. Ruggiero, Marcia Aparecida Gomes,1956-. II. Poldi, Kelly Cristina,1979-. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: GRASP and BRKGA metaheuristics applied to the maximum diversity problem
Palavras-chave em inglês:
Metaheuristic
GRASP (Metaheuristic)
Maximum diversity problem
Genetic algorithms
Área de concentração: Matemática Aplicada
Titulação: Mestre em Matemática Aplicada
Banca examinadora:
Marcia Aparecida Gomes Ruggiero [Orientador]
Aurelio Ribeiro Leite de Oliveira
Flávio Keidi Miyazawa
Data de defesa: 03-07-2017
Programa de Pós-Graduação: Matemática Aplicada

Dissertação de Mestrado defendida em 03 de julho de 2017 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). MARCIA APARECIDA GOMES RUGGIERO

Prof(a). Dr(a). AURELIO RIBEIRO LEITE DE OLIVEIRA

Prof(a). Dr(a). FLÁVIO KEIDI MIYAZAWA

As respectivas assinaturas dos membros encontram-se na Ata de defesa

Aos meus pais ....
Geraldo e Rosaine.

Agradecimentos

Agradeço a Deus por ter me dado calma, paciência e sabedoria ao longo desta jornada. Por nunca ter me desamparado, nem mesmo nos momentos mais difíceis.

Aos meus pais, Geraldo e Rosaine, pelo apoio e incentivo aos estudos desde cedo, me proporcionando sempre condições favoráveis para isso, tanto financeiras como emocionais.

À minha avó Dominga, por todas as tardes que cuidou de mim quando criança, pelo carinho e amor que sempre teve comigo.

À minha orientadora Marcia Aparecida Gomes Ruggiero e à minha coorientadora Kelly Cristina Poldi, pelos ensinamentos, paciência e atenção durante o mestrado, prestativas e exemplo de professoras para mim.

Aos professores Aurelio Ribeiro Leite de Oliveira e Flávio Keidi Miyazawa, pelas sugestões no exame de qualificação.

Ao meu amigo Mauricio, um irmão que ganhei na faculdade. Ao Carlos, pelas boas conversas na época em que dividíamos kitnet. Ao Lucas Moraes, pela ajuda nas implementações do API. E a todos os meus amigos que conheci na Unicamp, que comemoraram comigo nas alegrias, mas que também me apoiaram e ouviram nos momentos ruins.

À Dona Zefa, pelos bons cafés no IMECC.

À Unicamp e ao IMECC, pela infraestrutura oferecida.

À CAPES, pelo auxílio financeiro.

Resumo

Neste trabalho estudamos o Problema da Diversidade Máxima (PDM), que consiste em selecionar entre um conjunto de elementos, um subconjunto que seja o mais diverso possível. O problema é classificado como NP-difícil. Apresentamos as formulações quadrática e a linear inteira mista, aplicações e exemplo numérico. Resolvemos problemas pequenos de maneira exata e notamos que para problemas maiores é necessário o uso de heurísticas ou meta-heurísticas em sua resolução. Sendo assim, escolhemos as meta-heurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Biased Random-Key Genetic Algorithm* (BRKGA) para serem aplicadas ao PDM. No GRASP, adotamos uma construção de solução selecionando o elemento de acordo com a correspondente contribuição do mesmo ao valor da função objetivo. Após a construção de cada solução, aplicamos uma busca local e em seguida acionamos a técnica *path relinking*. No procedimento de busca local, empregamos uma busca do tipo exaustiva, realizando a melhor troca entre um elemento pertencente a solução e outro não pertencente. Geradas as soluções, aplicamos o *path relinking* na expectativa que entre cada par de soluções, exista uma solução com melhor valor para a função objetivo. Já no BRKGA, implementamos uma função *fitness* e um decodificador de soluções adaptados ao Problema da Diversidade Máxima. A função *fitness* adotada é a soma das diversidades entre os elementos selecionados e a decodificação de soluções é baseada na ordenação das chaves aleatórias. O objetivo deste trabalho é analisar e comparar os resultados obtidos pelas meta-heurísticas GRASP e BRKGA, tendo como referência os melhores resultados da literatura. Os problemas analisados nos testes computacionais foram extraídos da biblioteca MDPLIB. Observamos que as duas meta-heurísticas apresentam bons resultados na resolução do PDM, sendo que para problemas de pequeno e médio porte BRKGA obteve melhor desempenho que GRASP, enquanto que para problemas de grande porte, GRASP supera o desempenho do BRKGA.

Palavras-chave: Problema da Diversidade Máxima, meta-heurísticas, GRASP, BRKGA.

Abstract

In this work we study the Maximum Diversity Problem (MDP), which consists of selecting among a set of elements, a subset as diverse as possible. The problem is classified as NP-hard. We present the quadratic and mixed integer linear formulations, applications and numerical example. We solve small problems exactly and we note that for larger problems it is necessary to use heuristics or metaheuristics on its resolution. Therefore, we chose *Greedy Randomized Adaptive Search Procedure* (GRASP) and *Biased Random-Key Genetic Algorithm* (BRKGA) metaheuristics to be applied to the MDP. In GRASP, we adopted a solution construction by selecting the element according to its corresponding contribution to the objective function value. After the construction of each solution, we apply a local search and then we activate the *path relinking* technique. In the local search procedure, we used a exhaustive search, making the best exchange between an element which belongs to solution and another element that does not belong. Once the solutions were generated, we apply *path relinking* in the expectation that between each pair of solutions, there is a solution with better objective function value. In BRKGA, we implemented a *fitness* function and a solution decoder adapted to the Maximum Diversity Problem. The *fitness* function adopted is the sum of diversities between selected elements e the decoding of solutions is based on sorting random-keys. The objective of this work is to analyze and compare the results obtained by GRASP and BRKGA metaheuristics, having as reference the best results in the literature. The problems analyzed in the computational tests were extracted from the MDPLIB library. We observed that the two metaheuristics showed good results on MDP's resolution, moreover for small and medium sized problems BRKGA obtained better performance than GRASP, while for large problems, GRASP outperforms BRKGA.

Keywords: Maximum Diversity Problem, metaheuristics, GRASP, BRKGA.

Sumário

Capítulo 1

Introdução

O Problema da Diversidade Máxima (PDM) consiste em selecionar entre um conjunto de n itens, um subconjunto de m itens ($m < n$) de modo que seja o mais diverso possível. A maneira como é quantificada a diversidade varia com relação ao problema e a aplicação. Devido as suas características, o Problema da Diversidade Máxima pode ser classificado como um problema de otimização combinatória.

Problemas de otimização combinatória têm por objetivo maximizar ou minimizar uma função de várias variáveis, sujeito a um conjunto de restrições de igualdade e/ou desigualdade, além de restrições de integralidade sobre as variáveis, que podem ser discretas ou binárias. Nestes problemas o conjunto de todas as soluções pode ser enumerado, associando-se a cada uma delas um valor de função objetivo. No entanto, quando o tamanho do problema é grande, pode ser difícil enumerar todas as soluções.

O primeiro trabalho, com o objetivo de maximizar a diversidade, foi apresentado por Glover (1977), onde estudou-se uma forma de medir a diversidade em um conjunto de itens, e em seguida aplicou-se uma heurística para encontrar uma solução aproximada para o PDM.

O Problema da Diversidade Máxima possui aplicações em diversas áreas, como indústria farmacêutica, biologia, vendas e gerenciamento de recursos humanos. A seguir detalhamos algumas dessas aplicações.

Na indústria farmacêutica, um problema comum que está presente na composição de fármacos, é a seleção de um subconjunto de componentes químicos para síntese e avaliação biológica. A principal finalidade é maximizar a diversidade molecular. Em Agrafiotis (1997), é proposta uma família de algoritmos que combina um mecanismo de busca estocástico e uma função objetivo definida pelo usuário, que codifica qualquer critério de seleção desejável.

Na área de gerenciamento de recursos humanos, podemos destacar entre as aplicações, a seleção de portfólios. Dhir et al. (1993) ilustram um exemplo em análise de seguros, onde busca-se a dispersão e minimização dos riscos através da diversificação dos investimentos. Outro exemplo é a seleção de um grupo de pessoas, sejam elas para compor o corpo estudantil de uma universidade ou formar uma equipe de trabalho. Para isso é desejável que este grupo seja o mais heterogêneo possível. Kuo et al. (1993) descrevem um exemplo no qual deseja-se selecionar um grupo de pessoas para desempenhar os cargos administrativos em uma universidade.

Em preservação ecológica, um tema muito preocupante é a conservação da diversidade biológica, em virtude do aumento das taxas de degradação ambiental causadas pelas atividades humanas. Esforços têm sido feitos para a preservação de espécies em perigo, entretanto os recursos para a preservação são escassos, daí a importância em saber como aplicar os recursos da forma mais eficiente possível. Baseando-se no fato de que a diversidade pode ser quantificada, Glover et al. (1995) propuseram um modelo de programação quadrática 0-1 com objetivo de maximizar a biodiversidade sujeito a restrição de recursos.

Em Kochenberger e Glover (1999), os autores criaram um modelo de diversidade para *Data Mining* (uma importante ferramenta utilizada para desvendar informações escondidas em um conjunto de dados) que aumenta a funcionalidade dos métodos existentes para este tipo de ferramenta.

Tratamos agora da natureza do Problema da Diversidade Máxima. O PDM é classificado como NP-difícil, como demonstrado em Kuo et al. (1993). Os autores comprovam tal fato mostrando que o Problema de Clique (Harary (1969)), o qual é NP-completo, é reduzível ao PDM. Devido a sua natureza, é comum o uso de métodos heurísticos em sua resolução.

Heurísticas são métodos criados a partir de observações empíricas, que apesar de não garantirem uma solução ótima, encontram boas soluções em tempo razoável de processamento (Zäpfel e Braune (2010)). Nos últimos anos um novo conceito surgiu nesta área de pesquisa: as meta-heurísticas.

De acordo com Gonçalves e Resende (2011), meta-heurísticas são heurísticas para conceber heurísticas. Pode-se dizer que são procedimentos de alto nível, que coordenam heurísticas simples para encontrar soluções de melhor qualidade que aquelas obtidas por heurísticas simples. São exemplos de meta-heurísticas: *Simulated Annealing* (Kirkpatrick et al. (1983)), GRASP (Feo e Resende (1989)), *Tabu Search* (Glover et al. (1993)), Algoritmo genético (Holland (1975) e Reeves (1997)), Algoritmo Colônia de Formigas (Dorigo et al. (2006)), etc.

Nesse trabalho, escolhemos as meta-heurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Biased Random-Key Genetic Algorithm* (BRKGA) para serem aplicadas ao Problema da Diversidade Máxima. Optamos por GRASP, devido aos bons resultados obtidos quando aplicada em diversos problemas de otimização combinatória, inclusive o PDM (Silva et al. (2004) e Duarte e Martí (2007)). Já BRKGA, escolhemos pelo fato de não haver trabalhos na literatura relacionados à aplicação desta meta-heurística ao PDM. O objetivo deste trabalho é aplicar as duas meta-heurísticas ao Problema da Diversidade Máxima e comparar os resultados com os melhores resultados obtidos na literatura. Descrevemos nos próximos parágrafos como está organizado a dissertação.

No Capítulo 2, definimos o Problema da Diversidade Máxima e apresentamos duas formulações apontadas por Kuo et al. (1993). Além disso, para facilitar a compreensão do problema, exibimos um exemplo numérico, onde procura-se selecionar uma equipe de funcionários de forma que a equipe seja a mais diversificada possível.

No Capítulo 3, expomos as duas meta-heurísticas estudadas neste trabalho, GRASP e BRKGA, elucidando o funcionamento de cada uma delas. Apresentamos um exemplo numérico do GRASP proposto por Ghosh (1996) e explicamos um procedimento de *path relinking* do tipo guloso para ser aplicado após a fase de construção da solução e busca local. Em BRKGA propomos uma adaptação da meta-heurística para o Problema da Diversidade Máxima.

No Capítulo 4, apresentamos os resultados numéricos obtidos para um conjunto de problemas da biblioteca MDPLIB. Utilizamos o IBM ILOG CPLEX para resolver problemas pequenos e verificar para quais dimensões ele é resolvido em tempos razoáveis de processamento. Descrevemos alguns detalhes da implementação das meta-heurísticas GRASP e BRKGA. A primeira implementamos em MatLab, já na segunda empregamos o API em C++ desenvolvido por Toso e Resende (2015). Apresentamos e analisamos os resultados obtidos por estas meta-heurísticas e comparamos os resultados com os melhores obtidos na literatura.

Por fim no Capítulo 5, estão as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Problema da Diversidade Máxima (PDM)

Neste capítulo, descrevemos o Problema da Diversidade Máxima e apresentamos duas formulações para o problema. Além disso, exibimos um exemplo numérico para auxiliar na compreensão do mesmo.

2.1 Formulação do problema

O Problema da Diversidade Máxima (PDM) consiste em: dado um conjunto N com n elementos, a cada par de elementos (i, j) com $i, j \in N$ está associado um valor d_{ij}, denominado diversidade entre os elementos i e j. Quanto mais alto este valor, maior é a diversidade entre os elementos em questão. Considera-se que não há diversidade do elemento para com ele mesmo, ou seja, $d_{ii} = 0$ e, além disso, $d_{ij} = d_{ji}$. Temos então uma matriz $n \times n$ simétrica e com diagonal nula para representar a diversidade entre os elementos i e j. O objetivo do problema é selecionar um subconjunto com m elementos $(m < n)$, tal que a soma das diversidades entre os itens selecionados seja máxima. Desta forma, tem-se um total de $\frac{n!}{m!(n-m)!}$ combinações possíveis. Dependendo dos valores de n e m torna-se inviável analisar todas as possibilidades.

Em Kuo et al. (1993), os autores propõem quatro formulações para o Problema da Diversidade Máxima, entre as quais destacamos somente duas delas em nosso trabalho: a formulação quadrática inteira 0-1 e a formulação linear inteira mista.

A formulação quadrática é a seguinte:

$$\max z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \tag{2.1.1}$$

$$s.a. \begin{cases} \sum_{i=1}^{n} x_i = m & (2.1.2) \\ x_i = 0 \text{ ou } 1, \qquad i = 1, \ldots, n, & (2.1.3) \end{cases}$$

onde a variável de decisão x_i é tal que:

$$x_i = \begin{cases} 1, \text{ se o elemento } i \text{ é selecionado;} \\ 0, \text{ caso contrário;} \end{cases}$$

e:

- n: número total de elementos;
- m: número de elementos a selecionar;
- d_{ij}: valor da diversidade entre os elementos i e j.

A formulação quadrática possui apenas uma restrição que corresponde à condição que apenas m itens sejam selecionados. Já na formulação linear inteira mista, os autores consideram os mesmos parâmetros e a variável x_i da formulação quadrática, e incluem um nova variável de decisão y_{ij}:

$$y_{ij} = \begin{cases} 1, \text{ se o par de elementos } (i,j) \text{ é selecionado;} \\ 0, \text{ caso contrário.} \end{cases}$$

E a formulação linear é dada por:

$$\max z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} y_{ij} \tag{2.1.4}$$

$$s.a. \begin{cases} \sum_{i=1}^{n} x_i = m, & & (2.1.5) \\ x_i + x_j - y_{ij} \leq 1, & \forall (i,j) \in Q & (2.1.6) \\ -x_i + y_{ij} \leq 0, & \forall (i,j) \in Q & (2.1.7) \\ -x_j + y_{ij} \leq 0, & \forall (i,j) \in Q & (2.1.8) \\ y_{ij} = 0 \text{ ou } 1, & \forall (i,j) \in Q & (2.1.9) \\ x_i = 0 \text{ ou } 1, & i = 1, \ldots, n, & (2.1.10) \end{cases}$$

em que $N = \{1, 2, \ldots, n\}$ e $Q = \{(i,j) : i, j \in N, i < j\}$.

Para a linearização da formulação quadrática, foram adicionadas as restrições (2.1.6), (2.1.7) e (2.1.8) de forma a relacionar as variáveis de decisão x_i e y_{ij}. A seguir analisamos as possibilidades de modo a justificar que de fato as novas restrições englobam todos os casos possíveis e que as formulações linear e quadrática são equivalentes. Dado qualquer par de elementos $(i,j) \in Q$, temos 4 possibilidades para os valores de x_i e x_j:

(a) $x_i = 0$ e $x_j = 0$;

(b) $x_i = 1$ e $x_j = 0$;

(c) $x_i = 0$ e $x_j = 1$;

(d) $x_i = 1$ e $x_j = 1$.

Para os casos (a), (b) e (c) em que pelo menos uma das variáveis, x_i e/ou x_j, é nula, as restrições (2.1.6), (2.1.7) e (2.1.8) devem implicar em valor zero para o correspondente y_{ij}. Para o caso (d) estas restrições devem implicar que $y_{ij} = 1$.

No caso (a), substituindo os valores nas restrições (2.1.6), (2.1.7) e (2.1.8) tem-se que $y_{ij} \geq -1$, $y_{ij} \leq 0$ e $y_{ij} \leq 0$. Mas como y_{ij} é uma variável binária, implica que $y_{ij} = 0$.

No caso (b), substituindo novamente nas restrições (2.1.6), (2.1.7) e (2.1.8) chega-se em $y_{ij} \geq 0$, $y_{ij} \leq 1$ e $y_{ij} \leq 0$. Por y_{ij} ser binária, $y_{ij} = 0$. De modo análogo, no caso (c) encontra-se $y_{ij} = 0$.

No caso (d) as restrições (2.1.6), (2.1.7) e (2.1.8) fornecem $y_{ij} \geq 1$, $y_{ij} \leq 1$ e $y_{ij} \leq 1$. Logo conclui-se que $y_{ij} = 1$, já que y_{ij} é uma variável binária.

Resumindo, a restrição (2.1.6) garante que $y_{ij} = 1$ quando $x_i = 1$ e $x_j = 1$, e torna-se redundante se x_i e/ou x_j for igual a zero. Já as restrições (2.1.7) e (2.1.8) em conjunto, fazem com que $y_{ij} = 0$ caso x_i e/ou x_j seja igual a zero; e analisando separadamente, (2.1.7) e (2.1.8) são redundantes quando $x_i = 1$ e $x_j = 1$, respectivamente.

As duas formulações são equivalentes, diferindo no número de restrições e variáveis. Enquanto a formulação quadrática envolve n variáveis e uma restrição, a formulação linear possui $\frac{n^2-n}{2}$ variáveis binárias y_{ij}, além das n variáveis x_i e da restrição do número de itens selecionados, totalizando $\frac{3(n^2-n)}{2}$ restrições.

Implementamos o modelo (2.1.4)-(2.1.10) no IBM ILOG CPLEX para resolver problemas pequenos e verificar em quais dimensões o tempo de resolução é viável. Os resultados obtidos estão descritos no Capítulo 4.

2.2 Medida de diversidade

A diversidade entre dois elementos pode ser medida de várias maneiras, mas em geral é utilizada alguma métrica. Primeiramente, escolhe-se um conjunto R de atributos a serem analisados nos elementos do conjunto N. Estes atributos são as características do elemento em questão.

Considerando, por exemplo, um Problema da Diversidade Máxima, onde o conjunto de elementos é um grupo de funcionários, tais atributos podem ser idade, área de formação, experiência profissional, entre outros. A seguir estima-se um valor para cada um dos atributos dos elementos, representando por a_{ik} o valor do atributo k ($k \in R$) para o elemento i ($i \in N$). Se dois funcionários possuem idades próximas um do outro, a diferença entre seus valores no atributo idade é pequena; caso as idades sejam distantes, a diferença entre seus atributos é maior. Adotando a norma p como métrica, a diversidade entre os elementos i e j é dada por:

$$d_{ij} = \sqrt[p]{\sum_{k \in R} |a_{ik} - a_{jk}|^p}. \tag{2.2.1}$$

Vejamos um exemplo, adaptado de Bonotto e Cabral (2014), de um PDM na seleção de funcionários de uma empresa.

Uma empresa deseja escolher dentre 5 funcionários ($n = 5$) uma equipe composta por 3 deles ($m = 3$), de modo que esta equipe seja a mais heterogênea possível. São analisados em cada empregado os seguintes atributos: *Idade*, *Região*, *Área de conhecimento* e *Experiência profissional* (4 atributos, ou seja, $|R| = 4$). A Tabela 2.2.1 nos fornece os valores definidos para cada atributo:

Tabela 2.2.1: Tabela de atributos pessoais.

	Atributos			
Classif.	**Idade**	**Região**	**Área de conhecimento**	**Experiência (anos)**
1	16 - 19	Norte	Ciências Exatas	0 - 1
2	20 - 23	Nordeste	Ciências Agrária	2 - 3
3	24 - 27	Centro Oeste	Ciências Humanas	4 - 5
4	28 - 31	Sudeste	Ciências Sociais Aplicadas	6 - 7
5	32 - 34	Sul	Ciências da Saúde	8 - 10

No atributo *Região*, por exemplo, nota-se que as regiões Norte e Nordeste receberam valores 1 e 2, respectivamente, pois estão próximas uma da outra em relação ao mapa do Brasil. Entretanto a região Sul possui valor 5, já que está mais distante, quando comparada a região Nordeste, que recebeu valor 2. A Tabela 2.2.2 refere-se ao mapeamento dos atributos a_{ik} dos funcionários, em que a linha i é referente ao funcionário e a coluna k, o seu atributo:

Tabela 2.2.2: Mapeamento dos atributos (a_{ik}) referente ao k-ésimo atributo do funcionário i.

	Atributos			
Funcionário	**Idade**	**Região**	**Área de conhecimento**	**Experiência (anos)**
A	2	2	2	1
B	3	3	2	1
C	1	1	3	2
D	4	5	5	3
E	4	2	4	4

Observando-se as Tabelas 2.2.1 e 2.2.2, podemos concluir que o funcionário A tem de 20 a 23 anos, é natural da região Nordeste, Ciências Agrárias é sua Área de conhecimento e possui de 0 a 1 ano de Experiência Profissional.

Adotando a norma 2 como exemplo, podemos calcular a diversidade entre os funcionários através da equação (2.2.1). Tomando o caso entre os funcionários A e D, a diversidade d_{AD} é calculada da seguinte maneira:

$$\begin{aligned} d_{AD} &= \sqrt{\sum_{k=1}^{4} |a_{Ak} - a_{Dk}|^2} \\ &= \sqrt{|2-4|^2 + |2-5|^2 + |2-5|^2 + |1-3|^2} \\ &= 5{,}0990. \end{aligned}$$

De maneira semelhante, calculamos as outras diversidades, obtendo a seguinte matriz de diversidades ilustrada na Tabela 2.2.3:

Tabela 2.2.3: Diversidade entre os funcionários.

	A	**B**	**C**	**D**	**E**
A	0,0000	1,4142	2,0000	5,0990	4,1231
B	1,4142	0,0000	3,1623	4,2426	3,8730
C	2,0000	3,1623	0,0000	5,4772	3,8730
D	5,0990	4,2426	5,4772	0,0000	3,3166
E	4,1231	3,8730	3,8730	3,3166	0,0000

Uma maneira de encontrar uma solução ótima para este problema é analisar todas as possibilidades de formação de equipes e escolher aquela cuja soma das diversidades entre os funcionários é máxima. Como queremos selecionar 3 funcionários dentre 5, temos $\binom{5}{3} = 10$ possibilidades, conforme ilustradas na Tabela 2.2.4.

Tabela 2.2.4: Possibilidades de equipes.

Equipe	**Diversidades**	**Diversidade Total**
ABC	$d_{AB} + d_{AC} + d_{BC}$	6,5765
ABD	$d_{AB} + d_{AD} + d_{BD}$	10,7559
ABE	$d_{AB} + d_{AE} + d_{BE}$	9,4103
ACD	$d_{AC} + d_{AD} + d_{CD}$	12,5762
ACE	$d_{AC} + d_{AE} + d_{CE}$	9,9961
ADE	$d_{AD} + d_{AE} + d_{DE}$	12,5387
BCD	$\boldsymbol{d_{BC} + d_{BD} + d_{CD}}$	**12,8821**
BCE	$d_{BC} + d_{BE} + d_{CE}$	10,9082
BDE	$d_{BD} + d_{BE} + d_{DE}$	11,4322
CDE	$d_{CD} + d_{CE} + d_{DE}$	12,6668

Analisando a Tabela 2.2.4 observa-se que a equipe mais heterogênea é formada pelos funcionários B, C e D, pois é a que apresenta o maior valor para a diversidade, 12,8821.

No Capítulo 3, apresentamos métodos de solução para o Problema da Diversidade Máxima. Apontamos algumas das meta-heuríticas já aplicadas na resolução do problema e em seguida detalhamos o funcionamento das meta-heurísticas GRASP e BRKGA.

Capítulo 3

Meta-heurísticas GRASP e BRKGA aplicadas ao Problema da Diversidade Máxima.

A resolução de Problemas de Programação Linear Inteira (PPLI) é um tema que vem sendo muito estudado nas últimas décadas. Diversos métodos já foram desenvolvidos desde então, entre eles pode-se destacar métodos do tipo *branch-and-bound* e *branch-and-cut* (Wolsey (1998)). Tais métodos baseiam-se, na maioria das vezes, em ferramentas desenvolvidas para resolução de Problemas de Programação Linear (PPL), como o Método Simplex, por exemplo. A ideia destes métodos é relaxar um PPLI em um PPL e obter um limitante cada vez melhor para o problema. Com isso, é possível deixar de analisar algumas soluções inteiras do problema. O método prossegue até encontrar uma solução ótima ou então descobrir que o problema é infactível ou ilimitado. Apesar de não ter que explorar todas as possíveis soluções do problema, tais métodos se tornam excessivamente lentos em problemas com grande número de variáveis e restrições.

Nesses casos, é conveniente utilizar métodos heurísticos para a resolução do problema, pois apesar de não garantirem uma solução ótima, encontram boas soluções em tempo razoável de processamento.

O Problema da Diversidade Máxima é classificado como NP-difícil, como demonstrado em Kuo et al. (1993). Os autores comprovam tal fato mostrando que o Problema de Clique (Harary (1969)), o qual é NP-completo, é reduzível ao PDM. Devido a sua natureza, é comum o uso de métodos heurísticos em sua resolução. Diversas meta-heurísticas já foram aplicadas ao problema. A seguir descrevemos algumas delas.

Iniciamos a descrição com o *Simmulated Annealing* (Kirkpatrick et al. (1983)), que busca reproduzir o processo de resfriamento de um metal, onde a temperatura deve ser aumentada lentamente de forma que não comprometa a rigidez do material. Em cada iteração do método, gera-se um movimento aleatório buscando a melhora da solução. Se o movimento aumenta o valor da função objetivo, ele é aceito como novo estado, porém se diminuir, ele ainda pode ser aceito dependendo de uma probabilidade, que é baseada na distribuição de Boltzmann. Dois parâmetros que influenciam diretamente na qualidade da solução é a temperatura inicial e a taxa de redução de temperatura. Em Kincaid (1992) a temperatura inicial foi escolhida como sendo a maior diversidade e a taxa de redução adotada foi 0,89.

Outra meta-heurística aplicada foi o *Greedy Randomized Adaptive Search Procedure* (GRASP) proposta por Feo e Resende (1989), que é composta por duas fases: uma de construção da solução e outra de busca local. Na fase de construção, em cada etapa é selecionado um elemento aleatoriamente em uma lista de candidatos pré selecionados. Na fase de busca local utiliza-se algum método de busca para explorar o espaço de soluções. Em Ghosh (1996), o autor propõe uma fase de construção baseada na contribuição do elemento no valor da função objetivo da solução. Já na busca local, Ghosh apresenta uma busca do tipo exaustiva, realizando a melhor troca entre um elemento pertencente à solução e outro não pertencente. Silva et al. (2004) também aplicam GRASP ao PDM, onde apresentam três fases de construção, duas delas baseadas no critério de um parâmetro previamente estabelecido, k, das maiores distâncias, e outra na inserção do elemento mais distante. Utilizam duas buscas locais, a primeira como aplicada em Ghosh (1996) e a outra baseada em *Variable Neighborhood Search* (VNS). Já Duarte e Martí (2007) propõem duas fases de construção em GRASP. A primeira seleciona um elemento aleatoriamente como sendo o primeiro elemento e em cada etapa adiciona o elemento que possui a maior soma de diversidade com os elementos já selecionados. A fase de busca local é similar a Ghosh (1996), mas ao invés de procurar a melhor troca, seleciona o elemento com menor contribuição para sair da solução e insere o elemento com a maior diversidade em relação ao elemento que saiu.

Outra meta-heurística, também aplicada ao PDM, é a Busca Tabu, ou *Tabu Search* (Glover et al. (1993)), que tem por objetivo explorar o conjunto de soluções de forma a melhorar em cada etapa a solução, de modo que os elementos já inspecionados não voltem a ser analisados. Entre os trabalhos que aplicam esta meta-heurística está Kincaid (1992), que começa a busca com uma solução aleatória inicial. Em cada iteração é gerado um conjunto de trocas aleatórias e seleciona a melhor entre as trocas geradas, respeitando os elementos da lista tabu. Duarte e Martí (2007) também utilizam *Tabu Search*, de maneira similar a Kincaid (1992), porém a solução inicial é gerada por um processo de construção que utiliza memórias estruturais. Além disso, nas primeiras trocas, seleciona um elemento com probabilidade inversamente proporcional a sua contribuição na função objetivo.

Por fim, temos também *Scatter Search* (Laguna e Martí (2012)), um método baseado na evolução de soluções. As soluções de um conjunto de referência são evoluídas através da geração de subconjuntos, atualização e combinação de soluções. Em Gallego et al. (2009) os autores propõem uma distância baseada no número de vezes que os elementos selecionados da solução aparecem no conjunto de referência.

Nas próximas seções detalhamos as meta-heurísticas GRASP e BRKGA bem como suas correspondentes aplicações à resolução do Problema da Diversidade Máxima.

3.1 GRASP

A meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) foi proposta em Feo e Resende (1989). Ela consiste de duas fases: a primeira é a de construção de uma solução e a segunda, uma busca local.

A fase de construção da solução inicia-se com uma avaliação dos candidatos. A cada candidato é atribuído um valor, calculado de acordo com uma função de mérito g, este valor atribuído representa a contribuição do elemento no valor da função objetivo ao incluí-lo na solução. Desta maneira, cada candidato i terá um valor g_i associado, e a

partir destes valores tem-se um g_{min} e um g_{max} que são respectivamente, o mínimo e o máximo da função de mérito g.

Após a avaliação dos candidatos, tem início a elaboração da Lista Restrita de Candidatos (LRC). Constrói-se uma lista de possíveis candidatos a serem adicionados e sorteia-se um aleatoriamente entre eles. Este fato permite a diversificação da solução. A elaboração da lista pode ser realizada de duas formas: por cardinalidade ou baseado no valor da função de mérito.

Por cardinalidade, fixa-se um número k e selecionam-se os k melhores candidatos. Se o problema é de minimização serão selecionados os k candidatos de menor g_i, caso o problema seja maximizar, os k maiores g_i são escolhidos para compor a lista.

A outra maneira é basear-se no valor da função de mérito. Neste caso, selecionam-se os candidatos com g_i dentro de um intervalo pré determinado. Para um problema de minimização, o intervalo é $[g_{min}, \mu]$, com $\mu = g_{min} + \beta(g_{max} - g_{min})$, onde o parâmetro $\beta \in [0, 1]$. Se $\beta = 1$ o processo é totalmente aleatório, já se $\beta = 0$ o processo é guloso. De maneira análoga, o intervalo para um problema de maximização é $[\mu, g_{max}]$ com $\mu = g_{max} - \beta(g_{max} - g_{min})$.

A última etapa é selecionar um elemento aleatoriamente dentre aqueles pertencentes a LRC. Estas três etapas são repetidas até completar a solução.

Na fase de busca local, define-se como solução atual do processo aquela obtida na fase de construção. A seguir os valores da função objetivo são avaliados nas soluções vizinhas da solução atual, sendo que as soluções vizinhas são geradas a partir da solução atual, de acordo com a característica do problema. Caso exista alguma melhor, a solução é atualizada. O procedimento é repetido até que não seja encontrado uma solução vizinha melhor que a solução atual.

Diferentes soluções podem ser obtidas cada vez que o GRASP é aplicado. Tal fato se deve a escolha aleatória na fase de construção da solução. Por isso, para melhorar a qualidade da solução, GRASP é aplicado diversas vezes e ao final delas seleciona-se a melhor solução.

3.2 GRASP aplicado ao Problema da Diversidade Máxima

Descrevemos agora um GRASP aplicado ao PDM que foi proposto por Ghosh (1996). Nesse trabalho, o autor propõe uma fase de construção baseada na contribuição do elemento no valor da função objetivo. Neste processo a solução final é obtida a partir de uma solução parcial M_{k-1} contendo $k-1$ elementos com $1 \leq k \leq m$, onde m é o número de elementos a serem selecionados. Denota-se por N o conjunto de todos os n elementos do problema $N = \{1, 2, \ldots, n\}$.

Na fase de construção, para gerar uma solução M_k a partir de M_{k-1}, inicialmente é feita a avaliação dos candidatos. Para cada candidato $i \in N - M_{k-1}$ é calculado um limitante inferior ($\Delta z_L(i)$) e um limitante superior ($\Delta z_U(i)$), como uma estimativa do valor da função objetivo caso o candidato i seja adicionado à solução.

O cálculo do limitante inferior é dado por:

$$\Delta z_L(i) = \sum_{j \in M_{k-1}} d_{ij} + \sum_{n-m+1 \leq r \leq n-k} d_i^r(Q_{ik}), \tag{3.2.1}$$

onde $d_i^r(Q_{ik})$ denota a r-ésima maior diversidade do elemento i no conjunto Q_{ik}, sendo $Q_{ik} = N - M_{k-1} - \{i\}$. Portanto a primeira parcela representa a soma das diversidades de i com os demais elementos da solução parcial M_{k-1}, e a segunda, a soma das $m - k$ menores diversidades de i com os elementos que não fazem parte de M_{k-1}.

De maneira análoga, o limitante superior $\Delta z_U(i)$ é calculado pela equação:

$$\Delta z_U(i) = \sum_{j \in M_{k-1}} d_{ij} + \sum_{1 \leq r \leq m-k} d_i^r(Q_{ik}), \tag{3.2.2}$$

onde o primeiro somatório representa novamente a soma das diversidades de i com os demais elementos da solução parcial M_{k-1}, e o segundo, a soma das $m - k$ maiores diversidades de i com os elementos não pertencentes a M_{k-1}.

O fator aleatório do GRASP está em escolher em cada iteração um número $\alpha \in (0, 1)$ de maneira arbitrária, para realizar uma ponderação entre o limite inferior e o limite superior. Tal ponderação é calculada por:

$$\Delta z'(i) = (1 - \alpha)\Delta z_L(i) + \alpha \Delta z_U(i). \tag{3.2.3}$$

Calculado os $\Delta z'(i)$ para todos os candidatos $i \in N - M_{k-1}$, escolhe-se aquele de maior valor para incluir na solução, obtendo assim a nova solução parcial M_k com k elementos. A partir de M_k, o procedimento é repetido até obter-se uma solução com m elementos.

Finalizada a fase de construção da solução, dá-se início ao procedimento de busca local. Para este problema, Gosh define como vizinhança de uma solução M, o conjunto de todas as soluções obtidas trocando-se um elemento $i \in M$ por um elemento $j \in N - M$. Sendo assim, rotula-se por $\Delta z(i, j)$, a melhora ou piora no valor da função objetivo, ao efetuar a troca de i por j:

$$\Delta z(i, j) = \sum_{k \in M - \{i\}} (d_{jk} - d_{ik}). \tag{3.2.4}$$

Se $\Delta z(i, j) > 0$, implica que a solução vizinha obtida possui valor de função objetivo maior que a solução atual, caso contrário a solução vizinha tem valor menor.

Uma busca exaustiva é realizada, calculando-se todos os $\Delta z(i, j)$ da solução atual M. A seguir, a atualização é realizada referente a solução vizinha de maior $\Delta z(i, j) > 0$. O procedimento é repetido até a solução atual possuir somente $\Delta z(i, j) < 0$, neste caso o processo de busca local se encerra, retornando como solução da meta-heurística a solução atual.

3.2.1 Exemplo numérico

Para um melhor entendimento da heurística, apresentamos um exemplo do GRASP descrito na Seção 3.1 aplicado a um PDM com $n = 10$ e $m = 5$. A matriz de diversidades está representada na Tabela 3.2.1.

Tabela 3.2.1: Tabela com os valores de diversidades entre os itens.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	24	10	16	6	18	9	20	21	23
2	24	0	8	27	5	25	17	30	3	14
3	10	8	0	23	25	26	3	13	9	24
4	16	27	23	0	5	5	26	18	17	5
5	6	5	25	5	0	3	8	5	6	8
6	18	25	26	5	3	0	11	27	12	4
7	9	17	3	26	8	11	0	29	18	3
8	20	30	13	18	5	27	29	0	20	14
9	21	3	9	17	6	12	18	20	0	3
10	23	14	24	5	8	4	3	14	3	0

Os números em negrito na primeira linha e na primeira coluna representam o índice dos itens, enquanto os demais valores são as diversidades associadas a estes itens. Por exemplo, a diversidade entre os itens 4 e 8 é $d_{48} = 18$. Neste exemplo tem-se que $N = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Vamos iniciar com a solução parcial $M_2 = \{3, 5\}$, onde o subíndice 2 indica o número de elementos da solução parcial, definido na Seção 3.1 como $k-1$. Logo $k-1 = 2$, ou seja, $k = 3$. Calculamos agora a solução parcial M_3 a partir de M_2.

Primeiramente, para cada candidato $i \in N - M_2 = \{1, 2, 4, 6, 7, 8, 9, 10\}$ calculamos o limitante inferior ($\Delta z_L(i)$) e o limitante superior ($\Delta z_U(i)$). Considerando os cálculos para $i = 1$, o limitante inferior é dado por:

$$\begin{aligned} \Delta z_L(i) &= \sum_{j \in M_{k-1}} d_{ij} + \sum_{n-m+1 \leq r \leq n-k} d_i^r(Q_{ik}) \\ \Delta z_L(1) &= \sum_{j \in M_2} d_{1j} + \sum_{6 \leq r \leq 7} d_1^r(Q_{13}). \end{aligned}$$

A primeira parcela é calculada por:

$$\sum_{j \in M_2} d_{1j} = d_{13} + d_{15} = 10 + 6 = 16.$$

Sabendo que $Q_{13} = N - M_2 - \{1\} = \{2, 4, 6, 7, 8, 9, 10\}$, ordenamos as diversidades de $\{1\}$ com relação ao conjunto Q_{13} de forma decrescente, onde definimos que $d_1^r(Q_{13})$ representa a r-ésima maior diversidade. Tais diversidades estão destacadas em azul na Tabela 3.2.2, enquanto que sua ordenação pode ser visualizada na Tabela 3.2.3.

Tabela 3.2.2: Diversidades do item 1 com relação aos itens do conjunto Q_{13} destacadas em azul.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	24	10	16	6	18	9	20	21	23
2	24	0	8	27	5	25	17	30	3	14
3	10	8	0	23	25	26	3	13	9	24
4	16	27	23	0	5	5	26	18	17	5
5	6	5	25	5	0	3	8	5	6	8
6	18	25	26	5	3	0	11	27	12	4
7	9	17	3	26	8	11	0	29	18	3
8	20	30	13	18	5	27	29	0	20	14
9	21	3	9	17	6	12	18	20	0	3
10	23	14	24	5	8	4	3	14	3	0

Tabela 3.2.3: Valores de $d_1^r(Q_{13})$ ordenados.

$\boldsymbol{d_1^r(Q_{13})}$	**Valor**
$d_1^1(Q_{13})$	24
$d_1^2(Q_{13})$	23
$d_1^3(Q_{13})$	21
$d_1^4(Q_{13})$	20
$d_1^5(Q_{13})$	18
$d_1^6(Q_{13})$	16
$d_1^7(Q_{13})$	9

Podemos então calcular a segunda parcela:

$$\sum_{6\leq r\leq 7} d_1^r(Q_{13}) = d_1^6(Q_{13}) + d_1^7(Q_{13}) = 16 + 9 = 25. \tag{3.2.5}$$

Logo, o limitante inferior é:

$$\Delta z_L(1) = \sum_{j\in M_2} d_{1j} + \sum_{6\leq r\leq 7} d_1^r(Q_{13}) = 16 + 25 = 41. \tag{3.2.6}$$

Já o limitante superior pode ser obtido da seguinte maneira:

$$\begin{aligned} \Delta z_U(i) &= \sum_{j\in M_{k-1}} d_{ij} + \sum_{1\leq r\leq m-k} d_i^r(Q_{ik}) \\ \Delta z_U(1) &= \sum_{j\in M_2} d_{1j} + \sum_{1\leq r\leq 2} d_1^r(Q_{13}). \end{aligned}$$

A primeira parcela já foi calculada anteriormente no limitante inferior, enquanto que a segunda parcela é:

$$\sum_{1\leq r\leq 2} d_1^r(Q_{13}) = d_1^1(Q_{13}) + d_1^2(Q_{13}) = 24 + 23 = 47. \tag{3.2.7}$$

Portanto o limitante superior é dado por:

$$\Delta z_U(1) = \sum_{j\in M_2} d_{1j} + \sum_{1\leq r\leq 2} d_1^r(Q_{13}) = 16 + 47 = 63. \tag{3.2.8}$$

De maneira análoga, obtemos os limitantes para os demais itens $i \in N - M_2$, conforme mostra a Tabela 3.2.4.

Tabela 3.2.4: Valores dos limitantes dos itens do conjunto $N - M_2$.

	$\boldsymbol{\Delta z_L(i)}$	$\boldsymbol{\Delta z_U(i)}$
1	41	63
2	30	70
4	38	81
6	38	81
7	23	66
8	50	77
9	21	56
10	38	69

A próxima etapa da heurística consiste em realizar uma ponderação entre o limitante inferior e superior obtendo assim um valor $\Delta z'$:

$$\Delta z'(i) = (1-\alpha)\Delta z_L(i) + \alpha\Delta z_U(i), \tag{3.2.9}$$

onde o fator aleatório da heurística α usado neste exemplo foi fixado em $\alpha = 0{,}3$ por simplicidade. Assim, para $i = 1$ temos:

$$\begin{aligned} \Delta z'(1) &= (1-\alpha)\Delta z_L(1) + \alpha\Delta z_U(1) \\ &= 0{,}7 \times 41 + 0{,}3 \times 63 \\ &= 47{,}6. \end{aligned}$$

De maneira similar obtemos os valores de $\Delta z'$ para os demais itens de $N - M_2$, que podem ser visualizados na Tabela 3.2.5.

Tabela 3.2.5: Ponderação dos limitantes $\Delta z'$.

	$\Delta z'$
1	47,6
2	42,0
4	50,9
6	50,9
7	35,9
8	58,1
9	31,5
10	47,3

O item que possui o maior $\Delta z'$ é incluído na solução, que nesta situação é o item 8. A solução parcial agora é $M_3 = \{3, 5, 8\}$.

Prosseguindo da mesma maneira, obtemos $M_4 = \{3, 5, 6, 8\}$ e em seguida $M_5 = M = \{2, 3, 5, 6, 8\}$. De posse da solução final, realizamos o processo de busca local.

O valor da função objetivo da solução $M = \{2, 3, 5, 6, 8\}$ é $z = 167$. O primeiro passo da busca local é calcular a variação $\Delta z(i, j)$ para cada $i \in M$ e $j \in N - M$. Por exemplo, $\Delta z(2, 9)$ representa a variação na função objetivo de M trocando o item 2 pelo item 9. Se a variação é positiva, houve melhora no valor da função objetivo, caso contrário, piorou. Tal variação é calculada da seguinte maneira:

$$
\begin{aligned}
\Delta z(i,j) &= \sum_{k \in M - \{i\}} (d_{jk} - d_{ik}) \\
\Delta z(2,9) &= \sum_{k \in M - \{2\}} (d_{9k} - d_{2k}) \\
&= \sum_{k \in M - \{2\}} (d_{93} - d_{23}) + (d_{95} - d_{25}) + (d_{96} - d_{26}) + (d_{98} - d_{28}) \\
&= (9-8) + (6-5) + (12-25) + (20-30) \\
&= (1) + (1) + (-13) + (-10) \\
&= -21.
\end{aligned}
$$

Podemos visualizar as outras variações na Tabela 3.2.6.

Tabela 3.2.6: Valores de $\Delta z(i, j)$ para a solução $M = \{2, 3, 5, 6, 8\}$.

	1	**4**	**7**	**9**	**10**
2	-14	-17	-17	-21	-18
3	-4	-17	-7	-31	-32
5	34	35	22	6	18
6	-21	-8	-24	-43	-21
8	-17	-15	-36	-45	-25

Na Tabela 3.2.6 as linhas representam os itens $i \in M$ que serão substituídos; e as colunas, os itens $j \in N - M$ que são incluídos na solução. Calculadas todas as variações, escolhemos a maior variação $\Delta z(i, j) > 0$, que neste caso é $\Delta z(5, 4) = 35$.

Logo, a solução M é atualizada substituindo o item 5 pelo item 4, que resulta na nova solução $M_1 = \{2, 3, 4, 6, 8\}$. Para calcular o valor da função objetivo da nova solução basta somarmos o valor anterior com a variação $\Delta z(5, 4)$, que resulta em $z = 167 + 35 = 202$.

Sendo assim, as variações para a solução $M_1 = \{2, 3, 4, 6, 8\}$ são dadas na Tabela 3.2.7.

Tabela 3.2.7: Valores de $\Delta z(i, j)$ para a solução $M_1 = \{2, 3, 4, 6, 8\}$.

	1	**5**	**7**	**9**	**10**
2	-26	-52	-21	-32	-43
3	8	-52	13	-18	-33
4	-1	-35	-13	-29	-17
6	-13	-43	-8	-34	-26
8	-20	-50	-31	-47	-41

Neste caso a maior variação é $\Delta z(3, 7) = 13$, logo substituímos o elemento 3 pelo elemento 7, obtendo solução $M_2 = \{2, 4, 6, 7, 8\}$ de forma que $z = 202 + 13 = 215$. Apresentamos agora as variações da solução M_2, conforme mostra a Tabela 3.2.8

Tabela 3.2.8: Valores de $\Delta z(i, j)$ para a solução $M_2 = \{2, 4, 6, 7, 8\}$.

	1	**3**	**5**	**9**	**10**
2	-36	-34	-78	-32	-73
4	-5	-26	-55	-23	-41
6	1	-21	-45	-10	-32
7	-5	-13	-65	-31	-46
8	-37	-44	-83	-54	-78

Observa-se na Tabela 3.2.8 que $\Delta z(6, 1) = 1$ é a maior variação. Sendo assim trocamos o elemento 6 pelo elemento 1, obtendo $M_3 = \{1, 2, 4, 7, 8\}$ com $z = 215+1 = 216$. Na Tabela 3.2.9 temos as variações desta solução.

Tabela 3.2.9: Valores de $\Delta z(i, j)$ para a solução $M_3 = \{1, 2, 4, 7, 8\}$.

	3	**5**	**6**	**9**	**10**
1	-22	-46	-1	-11	-33
2	-49	-74	-37	-22	-53
4	-53	-63	-6	-25	-33
7	-27	-60	-6	-20	-25
8	-53	-73	-38	-38	-52

Como todas as variações são tais que $\Delta z(i, j) < 0$, encerramos a fase de busca local. Portanto a solução final do GRASP é $M' = \{1, 2, 4, 7, 8\}$, que é a solução ótima do problema. Na Tabela 3.2.10 listamos todas soluções obtidas durante a fase de busca local com seus respectivos valores de função objetivo:

Tabela 3.2.10: Soluções obtidas durante a fase de busca local.

Solução	**Valor de função objetivo (z)**
$\{2, 3, 4, 6, 8\}$	$z = 202$
$\{2, 4, 6, 7, 8\}$	$z = 215$
$\{1, 2, 4, 7, 8\}$	$z = 216$

3.3 *Path relinking*

É comum, no uso de heurísticas, a aplicação de algum método de intensificação na busca por soluções. Tais métodos visam melhorar a qualidade das soluções obtidas via heurística, buscando regiões que ainda não foram exploradas no espaço de soluções do problema. Entre eles destaca-se o *Path relinking* introduzido em Glover et al. (2000).

Neste método, dadas duas soluções do problema x e y, gera-se um conjunto de soluções, que representa o caminho para obter a solução y, a partir da solução x. Estes caminhos são obtidos através de movimentos que realizam trocas entres os elementos das soluções x e y, de modo que a primeira solução do caminho seja x e a solução final seja y. A escolha de como realizar estes movimentos deve ser feita de modo a minimizar o número de movimentos.

O objetivo do método é encontrar uma solução w entre aquelas geradas pelo caminho, que possua valor de função objetivo melhor que as soluções x e y.

No artigo de Resende et al. (2010), os autores propõem quatro variantes de *path relinking* para o Problema da Diversidade Máxima. Em nosso trabalho destacamos o modelo guloso que está explicado a seguir.

Sejam x e y duas soluções do problema, no *path relinking* guloso, a solução x é transformada na solução y gradualmente, trocando os elementos selecionados em x com os elementos selecionados em y. Durante o processo, os elementos em comum nas duas soluções permanecem inalterados, realizando as trocas somente entre os elementos diferentes.

Os autores definem Sel_x e Sel_y como sendo o conjunto dos elementos selecionados em x e y, respectivamente, logo $Sel_x \cap Sel_y$ é o conjunto dos elementos em comum entre x e y. Além disso, $Sel_x - Sel_y$ é o conjunto dos elementos que estão em x, mas não em y. De maneira análoga tem-se $Sel_y - Sel_x$.

Em cada etapa do método seleciona-se um elemento do conjunto $Sel_x - Sel_y$ para sair da solução e escolhe-se um elemento de $Sel_y - Sel_x$ para compor a solução. Sendo assim, se as soluções x e y diferem em r elementos, são necessárias r trocas, ou seja, se $|Sel_x - Sel_y| = |Sel_y - Sel_x| = r$, são realizados r movimentos ao longo do *path relinking*. Definimos aqui que o símbolo $|\ |$ denota a cardinalidade do conjunto.

Denota-se $p_k(x, y)$, a k-ésima solução do *path relinking* entre as soluções x e y. Como estamos partindo da solução x em direção a y, estabelece-se como primeira solução do caminho $x = p_0(x, y)$ e solução final $y = p_r(x, y)$. As soluções $p_1(x, y), \ldots, p_{r-1}(x, y)$ são as soluções intermediárias do caminho e é entre elas que espera-se encontrar uma solução melhor.

A partir de $p_k(x, y)$, a solução $p_{k+1}(x, y)$ é obtida trocando-se um elemento $i \in Sel_{p_k(x,y)-y}$ por um elemento $j \in Sel_{x-p_k(x,y)}$. São avaliadas todas as possibilidades de

trocas para escolher-se a melhor entre elas. Em nosso caso, a melhor troca é aquela que produz a solução com melhor função objetivo.

Para exemplificar o *path relinking*, consideremos um Problema da Diversidade Máxima, onde deseja-se selecionar 5 elementos num total de 10 ($m = 5$ e $n = 10$).

Sejam as soluções x e y tais que $Sel_x = \{2, 3, 5, 8, 10\}$ e $Sel_y = \{1, 2, 5, 7, 9\}$. Nota-se que $Sel_x \cap Sel_y = \{2, 5\}$, logo os elementos 2 e 5 estão presentes em todas as soluções do caminho de x para y. De acordo com o método, a primeira solução do caminho é $p_0(x, y) = x$. Para obter $p_1(x, y)$ é necessário trocar um elemento de $Sel_{p_0(x,y)-y} = \{3, 8, 10\}$ por um elemento de $Sel_{y-p_0(x,y)} = \{1, 7, 9\}$. Como $|Sel_x - Sel_y| = |Sel_y - Sel_x| = 3$, são necessários três movimentos para obter a solução y a partir de x. Desta forma, em cada movimento é necessário calcular todas as variações na função objetivo e escolher a troca que corresponde a maior variação.

As variações para a solução $p_0(x, y)$ estão representadas na Tabela 3.3.1, onde os números em negrito nas linhas são os índices dos elementos que devem sair, ao passo que os números em negrito nas colunas são os elementos que devem entrar. De acordo com a Tabela 3.3.1, a melhor troca corresponde a trocar o elemento 10 pelo elemento 1.

Tabela 3.3.1: Variações para a solução $p_0(x, y)$.

	1	**7**	**9**
3	4	12	24
8	-6	11	-17
10	52	29	37

Desta forma, temos $p_1(x, y) = \{1, 2, 3, 5, 8\}$. Agora avaliam-se todas as possibilidades de trocas entre os elementos de $Sel_{p_1(x,y)-y} = \{3, 8\}$ com $Sel_{y-p_1(x,y)} = \{7, 9\}$. Todas as possibilidades com suas respectivas variações na função objetivo estão representadas na Tabela 3.3.2.

Tabela 3.3.2: Variações para a solução $p_1(x, y)$.

	7	**9**
3	-22	2
8	1	-15

A melhor opção é trocar o elemento 3 pelo elemento 9. Sendo assim $p_2(x, y) = \{1, 2, 5, 8, 9\}$. Para o último movimento não é necessário avaliar todas as variações, pois para obtermos a solução y só há uma possibilidade de troca, que corresponde a troca de 8 por 7. Portanto $p_3(x, y) = \{1, 2, 5, 7, 9\} = y$.

Neste trabalho optamos em realizar todos os caminhos possíveis, no entanto tal estratégia não é comum na literatura devido aos tempos excessivos gastos nessa prática.

3.4 BRKGA aplicado ao Problema da Diversidade Máxima

A meta-heurística *Biased Random-Key Genetic Algorithm* (BRKGA) ou Algoritmo Genético com Chaves Aleatórias Viciadas, é baseada na meta-heurística Algoritmos Ge-

néticos (GA) proposta por Holland (1975). Em 1994 Bean propôs a meta-heurística *Random-Key Genetic Algorithm* (RKGA) ou Algoritmo Genético com Chaves Aleatórias, com o objetivo de evitar a geração de soluções infactíveis ao longo do processo. Em 2011 Resende e Gonçalves apresentam a meta-heurística BRKGA que tem por objetivo selecionar elementos de um subconjunto específico de soluções.

A seguir é apresentado um resumo das meta-heurísticas GA, RKGA e finalmente o BRKGA.

3.4.1 Algoritmos Genéticos (GA)

Um Algoritmo Genético ou *Genetic Algorithm* (GA) é uma meta-heurística baseada na Teoria da Evolução das espécies de Darwin (Holland (1975) e Reeves (1997)). A teoria diz que apenas os indivíduos mais aptos sobrevivem em determinado ambiente no decorrer do tempo. Neste caso, diz-se que os seres foram selecionadas pelo ambiente. Indivíduos, ainda que da mesma espécie, possuem genes diferentes, genes estes que definem certas características do mesmo. Estas características são transmitidas aos seus descendentes, porém algumas delas são mais favoráveis ao ambiente do que outras, fazendo com que nem todos os descendentes cheguem a vida adulta. Este é o conhecido processo de seleção natural. Mais detalhes sobre a Teoria da Evolução podem ser encontrados em Lopes e Rosso (2005).

Um Algoritmo Genético tenta simular a transmissão de genes aplicando-a nas soluções do problema. Em cada iteração do método temos um conjunto de soluções que representam os indivíduos de uma população. Uma solução pode ser representada por um vetor, onde cada entrada pode ser interpretada como as características do indivíduo. De maneira geral, em uma solução de boa qualidade (solução com valor de função objetivo próximo do valor da solução ótima) algumas características são preferíveis à outras. Sendo assim é interessante realizar pequenas modificações nas soluções com o intuito de melhorar o valor da função objetivo. Para o caso do Problema da Diversidade Máxima, as características são os elementos a serem escolhidos, pois pode ocorrer que alguns elementos sejam mais propensos a compor soluções de boa qualidade em comparação aos outros. As modificações podem ser feitas combinando as características de duas soluções a fim de criar uma melhor ou aplicando modificações aleatórias em parte dela. Em comparação ao processo biológico, essas modificações são equivalentes ao cruzamento entre indivíduos e o processo de mutação. A ideia é que ao longo das iterações, as piores soluções sejam descartadas e no conjunto de soluções restem apenas as melhores soluções ou as soluções mais "aptas".

No Algoritmo Genético é estabelecida uma analogia entre os termos da área de Otimização Combinatória e os conceitos utilizados em Genética. Tal associação é descrita a seguir.

Uma solução do problema é denominada de *indivíduo*. Para cada indivíduo está associado um *cromossomo*, que é um vetor de n entradas representando uma solução, onde n é o tamanho da instância do problema. Cada entrada do vetor é intitulada *gene* do cromossomo e o valor contido no gene é batizado de *alelo*. A posição que cada gene ocupa no cromossomo é chamada de *locus*, logo a primeira entrada do vetor será denominada de gene no locus 1. Um conjunto de cromossomos recebe o nome de *população*. A Figura 3.1 exemplifica esta analogia.

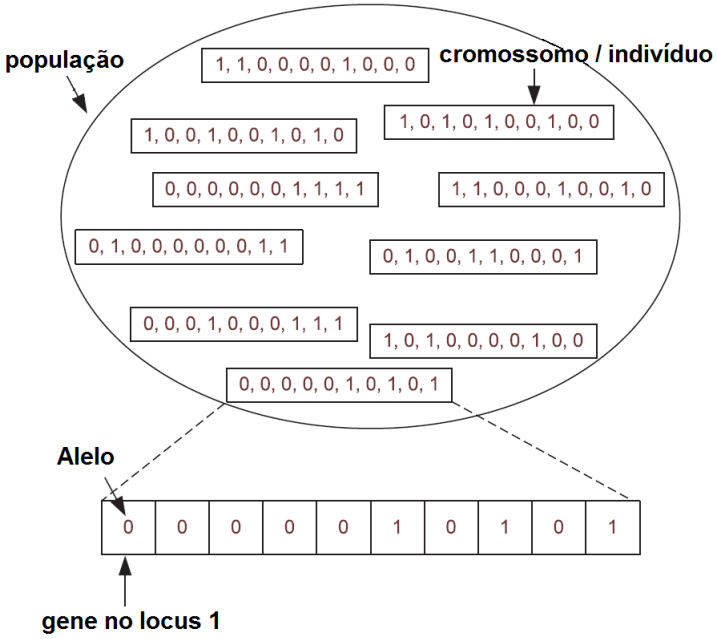

Figura 3.1: Exemplificação dos termos utilizados no Algoritmo Genético (baseada em Zapfel e Braune (2010)).

Em cada iteração do método, a população (ou conjunto de soluções) passa pelos processos de recombinação e modificação de soluções para gerar uma nova população. Estes procedimentos têm o intuito de aumentar o espaço de busca para obtenção de melhores soluções. Em analogia aos processos biológicos, a recombinação simula a *reprodução* e a modificação, a *mutação*. A esta nova população obtida dá-se o nome de *geração*. Dessa maneira, cada iteração do método está associada a uma geração de indivíduos.

Na Biologia, a mutação é o fenômeno da alteração brusca e inesperada do material genético (DNA) de um ser vivo, que pode ser transmitido para os seus descendentes. As mutações surgem a partir de um defeito no processo de multiplicação dos cromossomos que compõem o DNA ou na alteração da sequência de pares de um determinado gene (Lopes e Rosso (2005)).

Na meta-heurística, o processo de mutação pode ser aplicado escolhendo-se aleatoriamente uma entrada do vetor solução para ser modificada. Considerando um problema onde as soluções são representadas por vetores binários, aplica-se o seguinte procedimento após a escolha aleatória da entrada: se entrada for 0, ela é transformada em 1; caso seja 1, torna-se zero. É possível visualizar o processo de mutação na Figura 3.2, onde a quarta entrada foi sorteada para ser mutada:

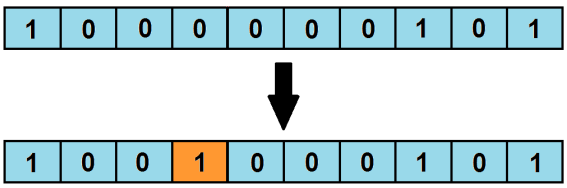

Figura 3.2: Mutação em um vetor solução binário (elaborada pelo autor).

Na natureza, um fenômeno importante da reprodução de dois indivíduos que ocorre no interior da célula é o *crossover*, quando na divisão celular por meiose acontece a quebra

das cromátides irmãs do cromossomo, seguida de uma troca de pedaços entre elas. Tal fato é um dos responsáveis pela variabilidade genética (Lopes e Rosso (2005)).

Nesta meta-heurística, duas estratégias são bastante utilizadas para simular o *crossover* entre dois cromossomos. A primeira é o tradicional operador *crossover* de um ponto. Neste processo, dadas duas soluções s_1 e s_2, define-se um ponto P para "quebrar" a solução em duas partes: I e II. A partir daí são geradas duas novas soluções (ou *descendentes*) d_1 e d_2, onde d_1 é constituída da parte I de s_1 e a parte II de s_2; enquanto d_2 é formada pela parte I de s_2 e parte II de s_1. Tal procedimento pode ser visualizado na Figura 3.3, tendo em vista as soluções de um PDM com $n = 10$ e $m = 3$:

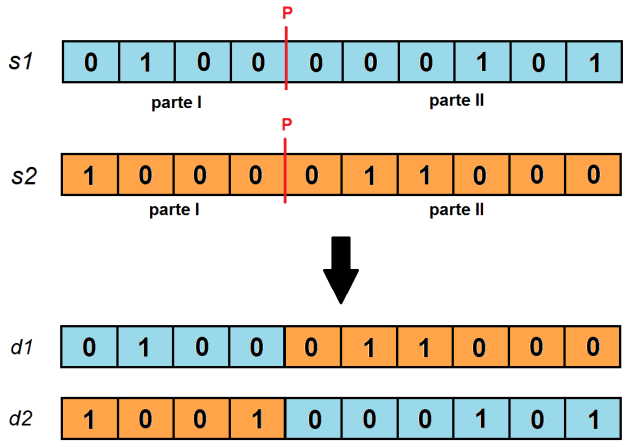

Figura 3.3: Representação do operador *crossover* de um ponto (elaborada pelo autor, baseada em Bean (1994)).

Esta estratégia pode ser estendida para mais de um ponto, neste caso o número de descendentes aumenta, pois pode haver mais possibilidades de permutações entre as partes das soluções.

Uma outra possibilidade é efetuar o *crossover* baseado no lançamento de uma moeda. Neste caso, o resultado obtido no lançamento é utilizado para definir os valores das entradas da nova solução. Por exemplo, definindo-se que o resultado Cara (C) implica na herança da característica da primeira solução (s_1) e Coroa (K) na herança da segunda (s_2), realizam-se n sorteios, onde n é o número de entradas do vetor. Desta forma, o i-ésimo lançamento define o valor da i-ésima entrada da nova solução. Mais uma vez, considerando um PDM com $n = 10$ e $m = 3$, tem-se:

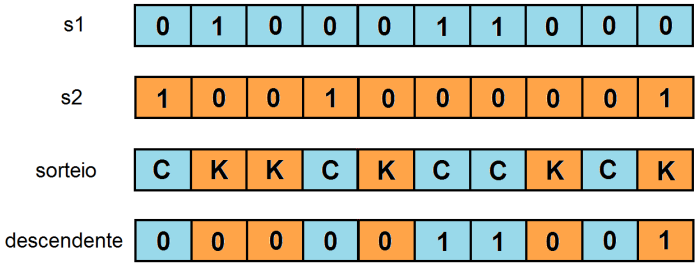

Figura 3.4: *Crossover* baseado no lançamento de uma moeda (elaborada pelo autor, baseada em Bean (1994)).

Uma das dificuldades que pode ocorrer no cruzamento ou mutação de soluções é a produção de soluções infactíveis. Apesar de ter-se obtido somente descendentes factíveis

nos dois exemplos anteriores de *crossover*, nem sempre tal fato ocorre, isso pode ser verificado no exemplo a seguir de um PDM com $n = 10$ e $m = 3$:

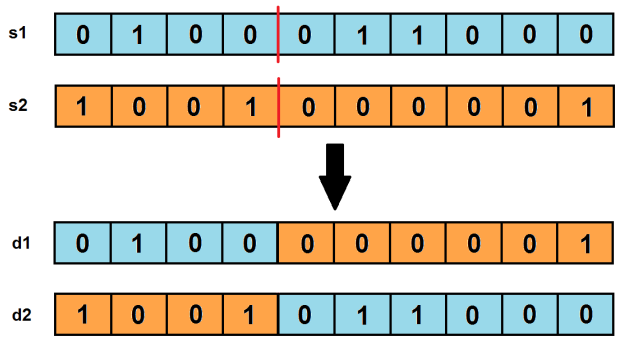

Figura 3.5: Produção de um descendente infactível através do *crossover* de um ponto (elaborada pelo autor, baseada em Bean (1994)).

Dadas duas soluções factíveis s_1 e s_2, aplicando o operador *crossover* de um ponto na quarta casa, adquire-se dois descendentes infactíveis d_1 e d_2. O primeiro descendente é infactível pois tem apenas dois itens selecionados, enquanto o segundo não é factível pois possui quatro itens selecionados.

Fato semelhante pode ser observado também no *crossover* baseado no lançamento de uma moeda:

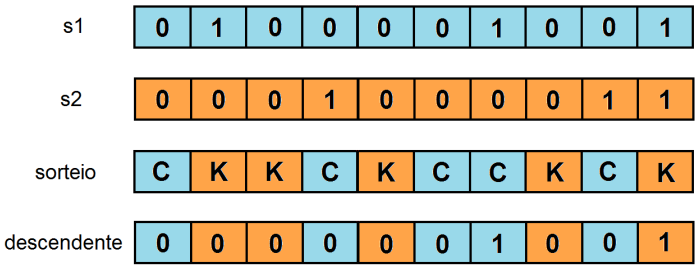

Figura 3.6: Produção de um descendente infactível através do *crossover* baseado no lançamento de uma moeda (elaborada pelo autor, baseada em Bean (1994)).

Neste caso, foi gerada uma solução com apenas dois itens selecionados.

Para contornar este problema da infactibilidade foi desenvolvido o conceito de chaves aleatórias, primeiramente apresentado por Bean (1994).

3.4.2 Algoritmo Genético com Chaves Aleatórias (RKGA)

O Algoritmo Genético com Chaves Aleatórias (*RKGA: Random-Key Genetic Algorithm*), utiliza um algoritmo determinístico, denominado de *decodificador*, responsável por decodificar as soluções. O decodificador recebe como entrada um cromossomo codificado, que é um vetor de n entradas geradas aleatoriamente em um intervalo [0,1] de maneira independente. Estes vetores codificados são denominados de chaves aleatórias. Em seguida o algoritmo associa a chave aleatória à uma solução do espaço de soluções do problema em questão.

Para o PDM por exemplo, baseado na decodificação de Bean (1994) por ordenação, pode-se utilizar o seguinte decodificador: "Dada uma solução com n chaves aleatórias,

ordena-se o vetor em ordem crescente dos valores. Posteriormente verifica-se a posição das m primeiras entradas do vetor ordenado em relação ao vetor inicial. Estas posições ocupadas no vetor inicial recebem valor 1 na decodificação enquanto as demais tem valor 0". Tal decodificação pode ser visualizada na Figura 3.7:

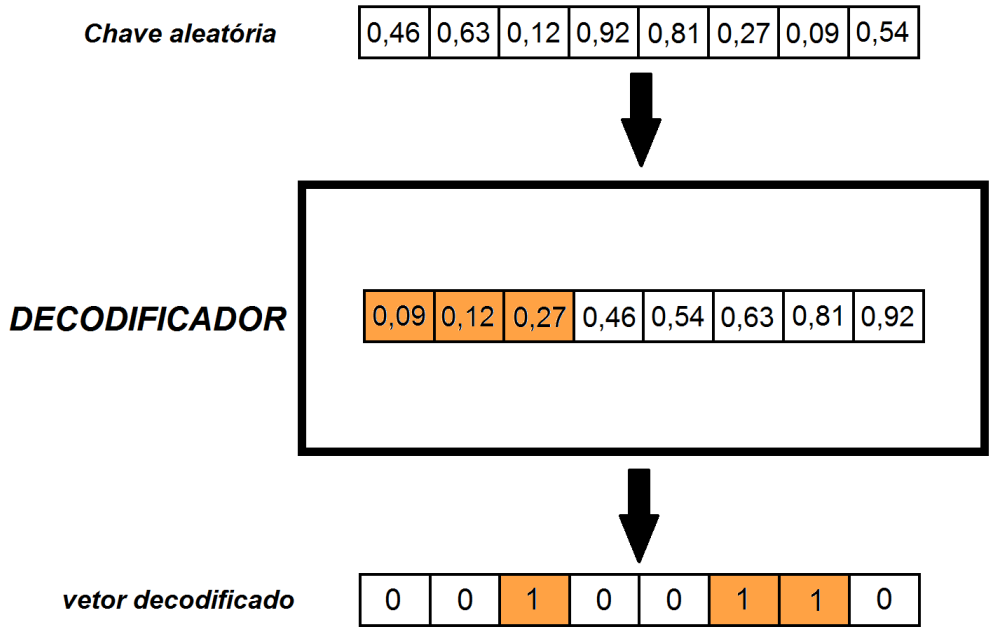

Figura 3.7: Processo de decodificação (elaborada pelo autor).

Neste caso, os processos de mutação e *crossover* são aplicados diretamente no vetor de chaves aleatórias e não no vetor solução do espaço de soluções do problema. Com esta estratégia evita-se a produção de soluções infactíveis.

O primeiro passo da meta-heurística RKGA consiste em gerar uma população inicial de indivíduos, que é composta por p vetores de chaves aleatórias. A seguir, cada um dos indivíduos da população é avaliado e classificado de acordo com uma função *fitness*, que pode ou não estar associada ao valor da função objetivo do problema. De acordo com Goldberg (1988), *fitness* é uma função para medir lucro ou utilidade que queremos maximizar. O termo vem da biologia, onde *fitness* é a habilidade do indivíduo sobreviver ao ataque de predadores na natureza, pestes e outros obstáculos da vida adulta.

Realizada a classificação, os indivíduos são separados em duas populações: População Elite, composta pelos indivíduos com os maiores valores de *fitness*, e a População Não Elite, que contém o restante da população. Definindo p_e como o número de indivíduos da População Elite, então $p - p_e$ é o tamanho da População Não Elite. O valor de p_e é escolhido de forma que $p_e < p - p_e$, ou seja, a População Elite deve ser menor que a População Não Elite, ou ainda, menor que metade da população total ($p_e < \frac{p}{2}$). A escolha é feita deste modo a fim de se evitar a convergência prematura do método.

Classificados os cromossomos, inicia-se o processo de formação da geração $k+1$ a partir da geração k. Primeiramente, todos os indivíduos da população elite da geração k são copiados para a geração $k+1$. Em seguida é estimado um número p_m de indivíduos dentre a população total da geração k que serão mutados para compor a nova população.

Com isso, resta adicionarmos $p-p_e-p_m$ indivíduos para completar a geração $k+1$, isto é feito pelo processo de *crossover*. Para cada *crossover* são sorteados dois indivíduos dentre a população toda. A Figura 3.8 ilustra o processo de formação de geração $k+1$:

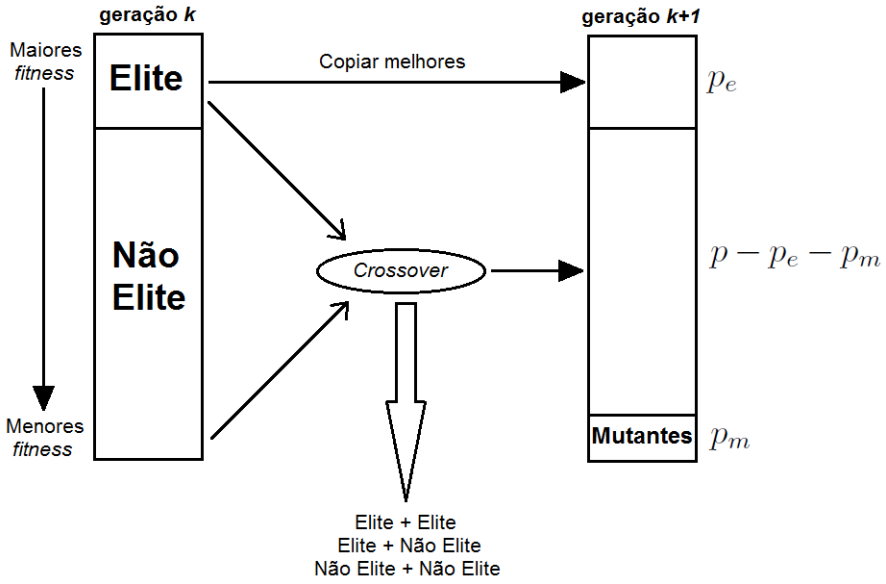

Figura 3.8: Processo de formação da geração $k + 1$ (elaborada pelo autor, baseada em Resende e Gonçalves (2011)).

O procedimento para gerar novas populações é repetido até alcançar uma geração formada quase que exclusivamente por um único tipo de cromossomo. Este cromossomo representa a solução final do problema.

3.4.3 Algoritmo Genético com Chaves Aleatórias Viciadas (BRKGA)

A diferença do BRKGA para o RKGA está na etapa de cruzamento dos indivíduos. Enquanto no RKGA para realizar o *crossover* são selecionados dois indivíduos dentre a população inteira, no BRKGA escolhe-se um indivíduo da população elite e outro da não elite. Isso é feito para assegurar as características boas dos indivíduos elite.

O processo de *crossover* nesta meta-heurística se dá através do lançamento de uma moeda viciada, ou seja, uma face tem mais chance de sair do que a outra. Selecionado o indivíduo elite e o não elite, define-se um parâmetro $\rho_e > 0{,}5$ associado à probabilidade do descendente herdar a característica do indivíduo elite.

Suponha que tal valor seja $\rho_e = 0{,}7$. No lançamento i é gerado um número aleatório pertencente a [0,1], se o número sorteado for menor que 0,7, o alelo i do descendente tem o mesmo valor do alelo do indivíduo elite, caso contrário é herdado o alelo do indivíduo não elite. Um exemplo pode ser visualizado na Figura 3.9:

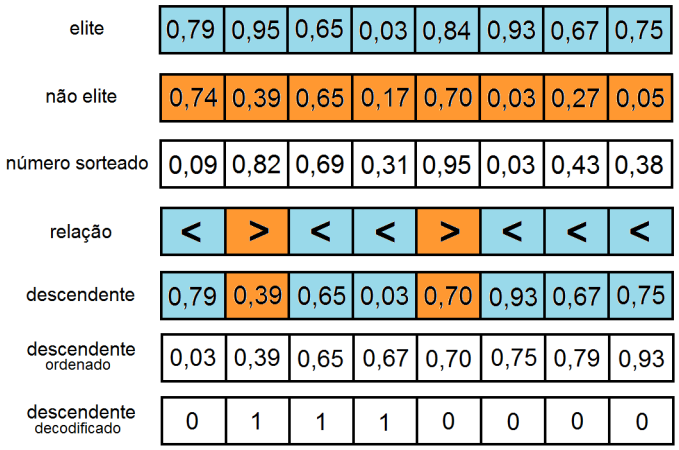

Figura 3.9: Crossover baseado no lançamento de uma moeda viciada (elaborada pelo autor, baseada em Resende e Gonçalves (2011)).

Os dois primeiros vetores são os indivíduos elite e não elite. O terceiro vetor representa o número sorteado em cada lançamento e abaixo dele um vetor indicando a relação entre o número sorteado e a probabilidade 0,7. O quinto vetor equivale ao vetor descendente. Os dois últimos são o vetor descendente ordenado e o vetor descendente decodificado.

Apresentadas as duas meta-heurísticas utilizadas neste trabalho, podemos conferir no Capítulo 4 a aplicação das mesmas para o Problema da Diversidade Máxima, bem como os resultados obtidos nos testes computacionais.

Capítulo 4

Testes computacionais

Neste capítulo apresentamos os resultados dos testes computacionais realizados, onde aplicou-se as meta-heurísticas GRASP e BRKGA ao Problema da Diversidade Máxima e comparou-se com os melhores resultados da literatura. Os problemas testados foram extraídos da biblioteca MDPLIB[1], que é composta por 315 problemas. Para 140 deles, são conhecidos os melhores valores para a função objetivo, obtidos por meio de heurísticas, pois são problemas de grande porte e uma solução exata é desconhecida. Nos demais 175 problemas não consta o valor das soluções ótimas ou melhores soluções conhecidas.

4.1 IBM ILOG CPLEX na resolução do Problema da Diversidade Máxima

Como visto no Capítulo 2, à medida que aumenta o número de possibilidades de selecionar os elementos, torna-se inviável resolver o problema por métodos exatos. Para verificar tal fato utilizamos o IBM ILOG CPLEX[2], que combina um ambiente de desenvolvimento integrado (IDE) com a eficiente *Optimization Programming Language* (OPL) e *solvers* do otimizador ILOG CPLEX de alto desempenho. Foi utilizado o modelo (2.1.4)-(2.1.10) na implementação.

Sabemos que o número de possibilidades depende simultaneamente dos valores de n e m, onde n é o número total de itens e m a quantidade de itens que queremos selecionar. Sendo assim, testamos problemas de pequeno porte, começando em $n = 20$ e aumentamos gradativamente esse valor até o tempo de resolução não ser mais praticável. Para isso, fixado o valor n, variamos o valor de m em 20%, 30%, 40% e 50% com relação ao número total de itens (0,2n; 0,3n; 0,4n e 0,5n). Os tempos de processamento, em segundos, podem ser visualizados na Tabela 4.1.1. Os testes foram realizados em um microcomputador Intel Core i7-4790 CPU 3.60 GHz com 16 GB de memória e sistema operacional Windows 10.

Nos problemas de tamanho $n = 20$ e $n = 30$, pelo fato dos tempos de processamento serem baixos, resolvemos 5 vezes e calculamos a média entre estes tempos. Nos demais

[1]Disponível em <http://www.optsicom.es/mdp/>. Último acesso em 25/02/2017.

[2]Disponível em <http://www-03.ibm.com/software/products/pt/ibmilogcpleoptistud>. Último acesso em 08/03/2017.

Tabela 4.1.1: Tempos de resolução à otimalidade no IBM ILOG CPLEX.

n	m	**Tempo(s)**
20	4	1,50
20	6	1,69
20	8	2,16
20	10	2,43
30	6	22,49
30	9	48,65
30	12	40,93
30	15	39,96
40	8	1518,86
40	12	14104,69
40	16	45954,41
40	20	71529,81
50	10	21928,05
50	15	7713,84
50	20	3954,25
50	25	2812,44

casos o tempo ilustrado corresponde ao tempo de um experimento. A partir de $n = 40$ observamos que o IBM ILOG CPLEX começa a se tornar impraticável, levando horas para obter uma solução ótima.

Nos casos em que $n = 50$, não foi possível obter uma solução ótima para nenhum valor de m, pelo fato do *solver* interface utilizar toda memória do computador. Sendo assim, nos exemplos com $n = 50$ os valores correspondem ao tempo de processamento até o momento em que é reportada a mensagem de falta de memória. Nestes casos, obtivemos os seguintes valores de GAP: 53,32% ($n = 50$ e $m = 10$), 60,91% ($n = 50$ e $m = 15$), 39,57% ($n = 50$ e $m = 20$) e 27,28% ($n = 50$ e $m = 25$).

Podemos observar pelos resultados da Tabela 4.1.1 que é inviável resolver o Problema da Diversidade Máxima através de métodos exatos para problemas com $n \geq 50$. Tal fato sugere a abordagem de métodos heurísticos para resolução em problemas de grande porte. Relatamos agora os testes realizados com as meta-heurísticas GRASP e BRKGA.

4.2 Testes computacionais utilizando a meta-heurística GRASP

A meta-heurística GRASP, conforme descrita na Seção 3.2, foi implementada no MATLAB[3] versão 7.10.0 (R2010a) e os testes realizados em Linux Ubuntu 16.04.4. Como explicado na Seção 3.1, o GRASP conta com uma fase de construção da solução e uma fase de busca local. Além desta busca local, aplicamos nas soluções pós busca local o *path relinking* relatado na Seção 3.3. O objetivo é intensificar a busca por regiões não exploradas, com o intuito de melhorar a qualidade dessas soluções.

[3]Disponível em <https://www.mathworks.com/products/matlab.html>. Último acesso em 30/03/2017.

Os testes foram realizados da seguinte maneira: para cada problema geramos um conjunto de 10 soluções pela fase de construção e busca local. Em seguida, aplicamos o *path relinking* em todos os pares de soluções possíveis desse conjunto, totalizando $\binom{10}{2} = 45$ pares de soluções. Para cada par de soluções (x, y), empregamos o *path relinking* de x para y, bem como o de y para x. Portanto, durante o *path relinking* testamos um total de 90 caminhos diferentes. Este número pode variar dependendo do número de soluções iguais no conjunto das 10 soluções geradas.

4.3 Testes computacionais utilizando a meta-heurística BRKGA

Na meta-heurística BRKGA os testes foram realizados em Linux Ubuntu 16.04.4 a partir do *Application Programming Interface* (API) para BRKGA em C++, denominado aqui de `brkgaAPI` (Toso e Resende (2015)). Este API pode ser aplicado, a princípio, a qualquer problema de otimização combinatória, bastando o usuário implementar um decodificador de soluções para seu problema e elaborar uma função *fitness*, utilizada para classificação da população.

Neste trabalho foi implementado o decodificador descrito na Seção 3.4.2 e como função *fitness* utilizou-se a função objetivo do Problema da Diversidade Máxima, ou seja, a soma das diversidades dos elementos selecionados.

O primeiro passo foi definir os valores dos parâmetros a serem utilizados nos testes dos problemas da biblioteca MDPLIB.

No estudo de heurísticas duas estratégias são fundamentais na busca por soluções: *intensificação* e *diversificação*. Acredita-se que boas soluções são obtidas quando há um equilíbrio entre intensificação e diversificação. No BRKGA estes dois termos estão diretamente ligados com os valores atribuídos aos parâmetros p_e, p_m e ρ_e, que representam respectivamente a proporção da população elite em relação à população total p, a proporção de mutantes com relação a população total p e a probabilidade do descendente herdar gene do indivíduo elite.

Na etapa de cruzamento de soluções (*crossover*), selecionamos um cromossomo da População Elite e outro da População Não Elite. Deste modo quando atribuímos baixos valores a p_e e p_m, e altos valores a ρ_e, estamos intensificando a busca por soluções, pois teremos poucas opções na escolha do indivíduo elite (p_e baixo) e estaremos privilegiando o mesmo no processo de *crossover*, já que terá mais chances de transmitir seus genes aos descendentes (ρ_e alto). Além disso, supõe-se poucas variações na população de uma geração para outra, pelo fato de poucos mutantes serem inseridos na população (baixo p_m). Por outro lado, no caso de valores elevados de p_e e p_m, e valores pequenos de ρ_e, estamos priorizando a diversificação de soluções, pois aumentamos as opções de selecionar indivíduos na População Elite no cruzamento de soluções (p_e alto) e os indivíduos elite terão chances menores de transmitir seus genes aos descendentes (ρ_e baixo). Além do mais, teremos uma variação maior de indivíduos de uma população para outra, pelo fato de haver mais mutantes (p_m alto).

Portanto para um bom desempenho da meta-heurística BRKGA, necessitamos encontrar um equilíbrio entre estes três parâmetros. Sendo assim, selecionamos 10 problemas da biblioteca MDPLIB entre os 140 de melhor solução conhecida e variamos os três pa-

râmetros para analisar quais valores produziam as melhores soluções. As dimensões dos problemas analisados estão descritas na Tabela 4.3.1.

Tabela 4.3.1: Dimensões dos Problemas analisados para definição dos parâmetros.

Problema	n	m
1	100	20
2	100	30
3	200	40
4	200	60
5	300	60
6	300	90
7	400	80
8	400	120
9	500	100
10	500	150

Para cada um destes problemas, fixamos o tamanho da população em $p = 1000$ e variamos os parâmetros p_e, p_m e ρ_e, escolhendo todas as combinações possíveis para a tripla $(p_e;\ p_m;\ \rho_e)$ entre os valores a seguir:

- $p_e = 10\%, 15\%, 20\%$ e 25%;
- $p_m = 10\%, 20\%$ e 30%;
- $\rho_e = 60\%, 65\%, 70\%, 75\%$ e 80%.

Estes valores foram baseados em Gonçalves e Resende (2011), que segundo os autores, os melhores resultados são obtidos para os seguintes intervalos dos parâmetros:

- $10\% \leq p_e \leq 25\%$;
- $10\% \leq p_m \leq 30\%$;
- $50\% < \rho_e \leq 80\%$.

Em cada problema, como realizamos os testes para 4 valores de p_e, 3 valores de p_m e 5 valores de ρ_e, temos $4 \times 3 \times 5 = 60$ triplas de parâmetros a serem analisadas. Para cada tripla resolvemos 5 vezes cada problema com o `brkgaAPI` e calculamos a média dessas 5 resoluções. Temos assim 60 médias para cada problema.

Para classificarmos a qualidade das soluções obtidas por cada tripla, adotamos como medida de desempenho o *Relative Desviation Index* (RDI), proposto por Kim (1993). Essa medida é em geral utilizada quando desconhece-se o valor da solução ótima, tornando impossível adotar a tradicional medida da taxa $\frac{T_H}{T_O}$, que calcula quanto o valor da solução obtida via heurística (T_H) representa em relação ao valor da solução ótima do problema (T_O).

Outro caso em que RDI é utilizado é quando o valor da solução ótima é um número muito pequeno ou muito grande, nestes casos a taxa $\frac{T_H}{T_O}$ pode não refletir a qualidade da solução, uma vez que a pior solução e melhor solução podem ter taxas próximas uma

da outra, mas devido a ordem de grandeza do valor da solução ótima, a diferença na qualidade das soluções é mascarada.

Denotando por S_B e S_W, respectivamente, o melhor e o pior valor obtido pela heurística, o RDI da solução i, com valor de função objetivo S_i é definido como:

$$RDI_i = \frac{S_B - S_i}{S_B - S_W}. \tag{4.3.1}$$

Nota-se que em RDI não utilizamos o valor da solução ótima, uma vez que pode ser difícil obtê-lo. Pela fórmula percebemos que o valor de RDI pertence ao intervalo [0,1], sendo que tem valor 0 para a melhor solução e 1 para a pior solução. Portanto quanto mais próximo de zero o valor de RDI, melhor é a qualidade da solução. Em nosso problema denotamos por S_B e S_W, a maior e menor média do valor da função objetivo obtida entre o conjunto de 60 triplas.

Cada problema tem seus próprios S_W e S_B. Para o Problema 1, por exemplo, o valor da maior média foi 1195, ao passo que a menor média foi 1184,8, logo $S_B = 1195$ e $S_W = 1184,8$. Tomemos agora como exemplo a tripla $M = (p_e = 15\%, p_m = 20\%$ e $\rho_e = 60\%)$ que obteve média $S_M = 1191,2$, logo o RDI dessa tripla é dado por:

$$RDI_M = \frac{S_B - S_M}{S_B - S_W} = \frac{1195 - 1191,2}{1195 - 1184,8} = 0,3725. \tag{4.3.2}$$

Em cada problema calculamos o RDI para cada uma das 60 triplas. Em seguida, tomamos a média do RDI de cada tripla nos 10 problemas. Feito isso, pudemos classificar as triplas de acordo com a média dos RDI. Ilustramos as médias de RDI das 5 melhores triplas e as 5 piores nas Tabelas 4.3.2 e 4.3.3.

Tabela 4.3.2: Triplas com 5 melhores médias de RDI.

p_e	p_m	ρ_e	**Média RDI**
0,20	0,10	0,70	0,1984
0,20	0,20	0,65	0,2474
0,15	0,20	0,65	0,2685
0,25	0,10	0,60	0,2741
0,25	0,20	0,60	0,2786

Tabela 4.3.3: Triplas com 5 piores médias de RDI.

p_e	p_m	ρ_e	**Média RDI**
0,10	0,20	0,75	0,6123
0,15	0,10	0,80	0,6267
0,10	0,20	0,80	0,6546
0,10	0,30	0,80	0,6836
0,10	0,10	0,80	0,7258

Observa-se pela Tabela 4.3.3 que as piores triplas possuem um baixo valor de p_e e alto valor de ρ_e. Já na Tabela 4.3.2 constatamos que todas as triplas possuem alto valor de p_e com exceção da terceira ($p_e = 15\%$), além disso os valores de (ρ_e) são baixos. Isso

comprova a importância em ajustar os parâmetros de forma a mesclar intensificação e diversificação.

De posse desses valores optamos por fixar os parâmetros em $p_e = 20\%$, $p_m = 10\%$ e $\rho_e = 70\%$, pois além de ser a melhor tripla, a diferença de seu respectivo RDI com relação a segunda melhor tripla é considerável, diferentemente do que ocorre com as próximas triplas, que possuem pouca variação de uma para a outra.

4.4 Resultados

4.4.1 Problemas analisados

Como descrito no início deste capítulo, os problemas testados foram extraídos da biblioteca MDPLIB. As instâncias dessa biblioteca estão divididas em 3 conjuntos de problemas, cada uma delas com características particulares. A seguir apresentamos uma breve descrição das classes[4]:

- **SOM**: possui 70 matrizes com números de 0 a 9 gerados de uma distribuição uniforme inteira. Este conjunto ainda está dividido em duas classes:
  - **SOM-a**: contém 50 instâncias gerados por Martí et al. (2010) com um gerador desenvolvido por Silva et al. (2004);
  - **SOM-b**: possui 20 instâncias geradas por Silva et al. (2004) e usada em muitos outros trabalhos, como por exemplo Aringhieri et al. (2008). Os tamanhos de n são 100, 200, 300, 400 e 500, com m variando em 10%, 20%, 30% e 40% dos valores de n.
- **GKD**: Este conjunto consiste de 145 matrizes para as quais os valores da diversidade foram calculados como a distância Euclidiana de coordenadas de pontos geradas aleatoriamente no intervalo 0 a 10. Este conjunto é dividido em três subconjuntos:
  - **GKD-a**: conta com 75 instâncias introduzidas por Glover et al. (1998);
  - **GKD-b**: dispõe de 50 matrizes geradas por Martí et al. (2010);
  - **GKD-c**: possui 20 matrizes de tamanho $n = 500$ e $m = 50$ geradas por Duarte e Martí (2007).
- **MDG**: Este conjunto apresenta 100 matrizes com valores reais selecionados aleatoriamente entre 0 e 10 a partir de uma distribuição uniforme. Ele está classificado em:
  - **MDG-a**: possui 40 matrizes geradas por Duarte e Martí (2007), sendo 20 delas de tamanho $n = 500$ e $m = 50$, e outras 20 com $n = 2000$ e $m = 200$;
  - **MDG-b**: contém o mesmo número de matrizes e os mesmos tamanhos que em MDG-a;
  - **MDG-c**: conjunto composto de 20 matrizes, as maiores da biblioteca, com tamanho $n = 3000$ e $m = 300, 400, 500$ e 600.

[4]Disponível em <http://www.optsicom.es/mdp/>. Último acesso em 25/02/2017.

Entre os conjuntos de problemas citados, optamos em testar aqueles que já possuíam solução conhecida, de forma a comparar os resultados das heurísticas utilizadas neste trabalho com os melhores resultados conhecidos na literatura. Sendo assim selecionamos problemas dos conjuntos: GKD-c, SOM-b, MDG-a e MDG-b. As melhores soluções conhecidas foram obtidas através da meta-heurística *Scatter Search* em Gallego et al. (2009).

4.4.2 GKD-c

Iniciamos a discussão dos resultados do conjunto GKD-c, que podem ser visualizados na Tabela 4.4.1. Na primeira coluna temos o problema analisado, a segunda e terceira coluna descrevem as dimensões do problema, a coluna "Melhor solução" representa a melhor solução conhecida na literatura, enquanto a quinta e sexta coluna apresentam os resultados obtidos pelas meta-heurísticas BRKGA e GRASP respectivamente.

Tabela 4.4.1: Soluções para o conjunto de problemas GKD-c.

Problema	n	m	**Melhor solução** (S_m)	**BRKGA** (S_b)	**GRASP** (S_g)
GKD-c-01	500	50	19485,18750	19485,18697	19485,18697
GKD-c-02	500	50	19701,53711	19701,53477	19701,53477
GKD-c-03	500	50	19547,20703	19547,20685	19547,20685
GKD-c-04	500	50	19596,46875	19596,46838	19596,46838
GKD-c-05	500	50	19602,62500	19602,62199	19602,62199
GKD-c-06	500	50	19421,55078	19421,53908	19421,53908
GKD-c-07	500	50	19534,30664	19534,30358	19534,30358
GKD-c-08	500	50	19487,32031	**19487,32053**	**19487,32053**
GKD-c-09	500	50	19221,63477	19221,62878	19221,62878
GKD-c-10	500	50	19703,35156	19703,33991	19703,33991
GKD-c-11	500	50	19587,12891	19587,12018	19587,12018
GKD-c-12	500	50	19360,23633	19360,22393	19360,22393
GKD-c-13	500	50	19366,69922	19366,69852	19366,69852
GKD-c-14	500	50	19458,56641	19458,56466	19458,56466
GKD-c-15	500	50	19422,15039	19422,14640	19422,14640
GKD-c-16	500	50	19680,20898	19680,20477	19680,20477
GKD-c-17	500	50	19331,38867	19331,38841	19331,38841
GKD-c-18	500	50	19461,39453	**19461,39460**	**19461,39460**
GKD-c-19	500	50	19477,32813	19477,32577	19477,32577
GKD-c-20	500	50	19604,84375	19604,84356	19604,84356

Para o conjunto de problemas GKD-c, as meta-heurísticas GRASP e BRKGA convergiram para soluções com mesmo valor de função objetivo em todos os problemas. No GRASP todas as soluções visualizadas na Tabela 4.4.1 representam a melhor solução entre as 10 obtidas, pois a solução encontrada ao final do *path relinking* não foi melhor que a obtida ao final da busca local.

Em todos os problemas deste conjunto as soluções se aproximaram muito da melhor solução conhecida, diferindo apenas nas últimas casas decimais. Em GKD-c-08 e GKD-c-18 conseguimos superar as melhores soluções em 0,00022 e 0,0007, respectivamente.

Nos problemas do conjunto GKD-c, o tempo para se obter uma solução pelo BRKGA foi de aproximadamente 5 minutos. Já em GRASP, o tempo médio para se obter uma solução na fase de construção e busca local foi 0,7 segundos. O tempo no *path relinking* depende do número de itens diferentes entre duas soluções, por isso relatamos neste trabalho o tempo médio do *path relinking* para realizar os 90 caminhos de cada problema nos referentes conjuntos. Em GKD-c este tempo foi cerca de 0,04 segundos, um tempo baixo em relação aos outros problemas de mesma dimensão, pois nas 10 soluções obtidas na fase de construção do GRASP, em todos os problemas obteve-se mais de uma solução com mesmo valor de função objetivo.

4.4.3 SOM-b

Neste conjunto de problemas, os tamanhos das instâncias variavam de $n = 100$ a $n = 500$ e os valores das diversidades eram todos números inteiros. Os resultados podem ser visualizados na Tabela 4.4.2.

Tabela 4.4.2: Soluções para o conjunto de problemas SOM-b.

Problema	n	m	**Melhor solução** (S_m)	**BRKGA** (S_b)	**GRASP** (S_g)
SOM-b-01	100	10	333	**333**	301
SOM-b-02	100	20	1195	**1195**	1185
SOM-b-03	100	30	2457	**2457**	**2457**
SOM-b-04	100	40	4142	**4142**	4137
SOM-b-05	200	20	1247	**1247**	1221
SOM-b-06	200	40	4450	**4450**	4432
SOM-b-07	200	60	9437	**9437**	9415
SOM-b-08	200	80	16225	**16225**	16224
SOM-b-09	300	30	2694	2676	2674
SOM-b-10	300	60	9689	9648	9621
SOM-b-11	300	90	20743	20676	20681
SOM-b-12	300	120	35881	35878	35875
SOM-b-13	400	40	4658	4625	4605
SOM-b-14	400	80	16956	16933	16914
SOM-b-15	400	120	36317	36261	36240
SOM-b-16	400	160	62487	62399	62398
SOM-b-17	500	50	7141	7094	7045
SOM-b-18	500	100	26258	26214	26205
SOM-b-19	500	150	56572	56559	56551
SOM-b-20	500	200	97344	97304	97198

Neste caso, a meta-heurística BRKGA conseguiu melhores resultados que o GRASP. Dentre os 20 problemas analisados, BRKGA obteve as melhores soluções conhecidas na literatura em 8 deles (SOM-b-01 a SOM-b-08), cujos tamanhos são $n = 100$ e $n = 200$. A partir de $n = 300$, todas as soluções representam mais de 99% do valor da melhor solução. Esse resultado é obtido a partir da razão entre a solução obtida via BRKGA (S_b) e a melhor solução conhecida da literatura (S_m). Os valores dessas taxas podem ser visualizadas na Tabela 4.4.3. O pior caso foi para o SOM-b-13 que alcançou 99,29%.

Já o GRASP atingiu a melhor solução somente no SOM-b-03. Em compensação nos

Tabela 4.4.3: Taxas com relação a melhor solução conhecida na literatura para o conjunto SOM-b.

Problema	**BRKGA** ($\frac{S_b}{S_m}$)	**GRASP** ($\frac{S_g}{S_m}$)
SOM-b-01	100,00%	90,39%
SOM-b-02	100,00%	99,16%
SOM-b-03	100,00%	100,00%
SOM-b-04	100,00%	99,88%
SOM-b-05	100,00%	97,91%
SOM-b-06	100,00%	99,60%
SOM-b-07	100,00%	99,77%
SOM-b-08	100,00%	99,99%
SOM-b-09	99,33%	99,26%
SOM-b-10	99,58%	99,30%
SOM-b-11	99,68%	99,70%
SOM-b-12	99,99%	99,98%
SOM-b-13	99,29%	98,86%
SOM-b-14	99,86%	99,75%
SOM-b-15	99,85%	99,79%
SOM-b-16	99,86%	99,86%
SOM-b-17	99,34%	98,66%
SOM-b-18	99,83%	99,80%
SOM-b-19	99,98%	99,96%
SOM-b-20	99,96%	99,85%

outros casos, sua taxa com relação a melhor solução conhecida ($\frac{S_g}{S_m}$) foi sempre superior a 99%, com exceção de SOM-b-01, SOM-b-05, SOM-b-13 e SOM-b-17 que atingiram respectivamente 90,39% (pior caso), 97,91%, 98,86% e 98,66%.

Destacamos que os únicos casos em que a solução obtida através do *path relinking* foi maior que a solução pós busca local, foi em SOM-b-19 e SOM-b-20, onde obteve-se uma melhora de 56549 para 56551 e 97191 para 97198, nos respectivos valores de função objetivo.

Em relação aos tempos obtidos na resolução do conjunto de problemas SOM-b, dividimos os relatos de acordo com as dimensões. Nos problemas com $n = 100$, o tempo médio para obtenção de uma solução em BRKGA foi 1 minuto, enquanto que em GRASP foi 0,3 segundos. Já o *path relinking* utilizou em média 0,3 segundos por problema.

Nos problemas de dimensão $n = 200$, BRKGA levou 3 minutos para obter uma solução e GRASP 0,4 segundos. O tempo médio do *path relinking* foi 1,2 segundos.

Para $n = 300$, uma solução foi obtida em 3 minutos no BRKGA e em 0,7 segundos no GRASP. Já o *path relinking* utilizou 3,6 segundos em média para realizar os 90 caminhos.

Nos problemas com $n = 400$, o tempo para obter uma solução em BRKGA foi 4 minutos, à medida que em GRASP foi 2,2 segundos. O *path relinking* gastou 11,7 segundos.

Por fim nos problemas de tamanho $n = 500$, foi necessário 5 minutos para obtenção de uma solução em BRKGA e 3,3 segundos em GRASP. No *path relinking* o tempo

praticado foi 14 segundos por problema.

4.4.4 MDG-a

A análise dos resultados do conjunto de problemas MDG-a está dividida em duas partes: a primeira compreende a análise dos problemas de dimensão $n = 500$ e $m = 50$, já a segunda contempla os problemas de tamanhos $n = 2000$ e $m = 200$. Para os problemas com $n = 500$ e $m = 50$, os resultados obtidos pelas meta-heurísticas podem ser visualizados na Tabela 4.4.4.

Tabela 4.4.4: Soluções para o conjunto de problemas MDG-a com $n = 500$ e $m = 50$.

Problema	n	m	**Melhor solução** (S_m)	**BRKGA** (S_b)	**GRASP** (S_g)
MDG-a-01	500	50	7833,83252	7780,57	7685,31
MDG-a-02	500	50	7771,66162	7709,15	7639,70
MDG-a-03	500	50	7759,35986	7709,15	7699,02
MDG-a-04	500	50	7770,24170	7686,87	7674,63
MDG-a-05	500	50	7755,23096	7673,74	7699,75
MDG-a-06	500	50	7773,70996	7740,37	7682,79
MDG-a-07	500	50	7771,73096	7728,36	7719,66
MDG-a-08	500	50	7750,88135	7717,16	7681,06
MDG-a-09	500	50	7770,07080	7735,38	7705,44
MDG-a-10	500	50	7780,35059	7718,25	7671,50
MDG-a-11	500	50	7770,95068	7696,40	7651,39
MDG-a-12	500	50	7757,65039	7713,67	7718,42
MDG-a-13	500	50	7798,43164	7700,78	7715,97
MDG-a-14	500	50	7795,63135	7752,17	7707,74
MDG-a-15	500	50	7736,84033	7671,57	7679,13
MDG-a-16	500	50	7792,77197	7752,57	7641,53
MDG-a-17	500	50	7787,20068	7688,47	7764,28
MDG-a-18	500	50	7756,26172	7701,64	7631,18
MDG-a-19	500	50	7755,41064	7677,28	7684,87
MDG-a-20	500	50	7733,86133	7685,36	7654,91

Nota-se que o BRKGA obteve melhores resultados que o GRASP na maioria dos problemas. As únicas exceções são os Prolemas MDG-a-05, MDG-a-12, MDG-a-13, MDG-a-15, MDG-a-17 e MDG-a-19.

MDG-a-07 foi o único problema em que o *path relinking* obteve uma solução melhor que a solução pós busca local do GRASP, conseguindo uma melhora de 7717,93 para 7719,66.

Em nenhum caso as meta-heurísticas alcançaram a melhor solução conhecida, entretanto aproximaram-se bastante dela. Na Tabela 4.4.5 ilustramos as taxas em porcentagem, que a solução obtida via heurística representa em relação a melhor solução da literatura. Em ambas meta-heurísticas todas as taxas são maiores que 98%.

Tabela 4.4.5: Taxas com relação a melhor solução conhecida para o conjunto de MDG-a.

Problema	**BRKGA** ($\frac{S_b}{S_m}$)	**GRASP** ($\frac{S_g}{S_m}$)
MDG-a-01	99,32%	98,10%
MDG-a-02	99,20%	98,30%
MDG-a-03	99,35%	99,22%
MDG-a-04	98,93%	98,77%
MDG-a-05	98,95%	99,28%
MDG-a-06	99,57%	98,83%
MDG-a-07	99,44%	99,33%
MDG-a-08	99,56%	99,10%
MDG-a-09	99,55%	99,17%
MDG-a-10	99,20%	98,60%
MDG-a-11	99,04%	98,46%
MDG-a-12	99,43%	99,49%
MDG-a-13	98,75%	98,94%
MDG-a-14	99,44%	98,87%
MDG-a-15	99,16%	99,25%
MDG-a-16	99,48%	98,06%
MDG-a-17	98,73%	99,71%
MDG-a-18	99,30%	98,39%
MDG-a-19	98,99%	99,09%
MDG-a-20	99,37%	98,98%

Neste primeiro conjunto de problemas de MDG-a, o tempo para obtenção de uma solução via BRKGA foi 5 minutos, enquanto que o GRASP levou 0,8 segundos. No *path relinking* foi necessário 5,2 segundos em média para realizar os 90 caminhos em cada problema.

Para o segundo conjunto de problemas ($n = 2000$ e $m = 200$) obtivemos resultados opostos em relação ao primeiro conjunto de problemas ($n = 500$ e $m = 50$). Enquanto no primeiro conjunto BRKGA sobressaiu-se melhor que o GRASP na maioria dos casos, no segundo conjunto, o GRASP superou o BRKGA em todos os problemas. Tal fato pode ser verificado na Tabela 4.4.6.

Destacamos que em cinco problemas houve melhora no valor da função objetivo após a aplicação do *path relinking*. São eles: MDG-a-21 (com melhora de 113197 para 113199), MDG-a-24 (113211 para 113220), MDG-a-26 (113498 para 113500), MDG-a-28 (113274 para 113276) e MDG-a-29 (113408 para 113410).

Observamos pela Tabela 4.4.7 que em BRKGA, as taxas correspondem entre 98% e 99% do valor da função objetivo da melhor solução conhecida. Já em GRASP essas taxas são todas superiores a 99%.

No segundo conjunto de problemas de MDG-a, o BRKGA gastou 30 minutos em média para obter uma solução, ao passo que GRASP levou 53,5 segundos. Já no *path relinking* o tempo foi de 6 minutos em média para cada problema.

Tabela 4.4.6: Soluções para o conjunto de problemas MDG-a com $n = 2000$ e $m = 200$.

Problema	n	m	**Melhor solução** (S_m)	**BRKGA** (S_b)	**GRASP** (S_g)
MDG-a-21	2000	200	114259	112926	113199
MDG-a-22	2000	200	114327	112739	113640
MDG-a-23	2000	200	114123	112855	113217
MDG-a-24	2000	200	114040	112659	113220
MDG-a-25	2000	200	114064	112875	113244
MDG-a-26	2000	200	114204	112993	113500
MDG-a-27	2000	200	114338	112790	113464
MDG-a-28	2000	200	114158	112674	113276
MDG-a-29	2000	200	114132	112670	113410
MDG-a-30	2000	200	114197	112706	113292
MDG-a-31	2000	200	114139	112814	113362
MDG-a-32	2000	200	114092	112993	113428
MDG-a-33	2000	200	114124	112558	113289
MDG-a-34	2000	200	114203	112994	113195
MDG-a-35	2000	200	114180	112943	113216
MDG-a-36	2000	200	114252	112787	113375
MDG-a-37	2000	200	114213	112919	113294
MDG-a-38	2000	200	114378	113059	113435
MDG-a-39	2000	200	114201	112534	113369
MDG-a-40	2000	200	114191	112857	113351

Tabela 4.4.7: Taxas com relação a melhor solução conhecida na literatura para o segundo conjunto de problemas de MDG-a.

Problema	**BRKGA** ($\frac{S_b}{S_m}$)	**GRASP** ($\frac{S_g}{S_m}$)
MDG-a-21	98,83%	99,07%
MDG-a-22	98,61%	99,40%
MDG-a-23	98,89%	99,21%
MDG-a-24	98,79%	99,28%
MDG-a-25	98,96%	99,28%
MDG-a-26	98,94%	99,38%
MDG-a-27	98,65%	99,24%
MDG-a-28	98,70%	99,23%
MDG-a-29	98,72%	99,37%
MDG-a-30	98,69%	99,21%
MDG-a-31	98,84%	99,32%
MDG-a-32	99,04%	99,42%
MDG-a-33	98,63%	99,27%
MDG-a-34	98,94%	99,12%
MDG-a-35	98,92%	99,16%
MDG-a-36	98,72%	99,23%
MDG-a-37	98,87%	99,20%
MDG-a-38	98,85%	99,18%
MDG-a-39	98,54%	99,27%
MDG-a-40	98,83%	99,26%

4.4.5 MDG-b

A exemplo do conjunto MDG-a, também dividimos a discussão dos resultados em dois grupos. O primeiro grupo contém problemas com $n = 500$ e $m = 50$, enquanto o segundo grupo possui problemas de dimensões $n = 2000$ e $m = 200$. Para o primeiro grupo temos os resultados ilustrados na Tabela 4.4.8.

Tabela 4.4.8: Soluções para o conjunto de problemas MDG-b com $n = 500$ e $m = 50$.

Problema	n	m	**Melhor solução** (S_m)	**BRKGA** (S_b)	**GRASP** (S_g)
MDG-b-01	500	50	778030,625	774674,50	771444,13
MDG-b-02	500	50	779963,6875	770271,73	769738,42
MDG-b-03	500	50	776768,4375	772498,45	769582,12
MDG-b-04	500	50	775394,625	772505,52	771886,80
MDG-b-05	500	50	775611,0625	770621,52	768218,11
MDG-b-06	500	50	775153,6875	768430,60	768656,55
MDG-b-07	500	50	777232,875	772822,05	775336,40
MDG-b-08	500	50	779168,75	775827,28	778378,78
MDG-b-09	500	50	774802,1875	768912,55	765212,55
MDG-b-10	500	50	774961,3125	771307,93	768039,06
MDG-b-11	500	50	777468,875	773717,43	773418,29
MDG-b-12	500	50	775492,9375	768745,26	769575,78
MDG-b-13	500	50	780191,875	779457,62	767390,11
MDG-b-14	500	50	782232,75	780711,32	779886,86
MDG-b-15	500	50	780300,375	776374,41	765244,48
MDG-b-16	500	50	775436,25	773929,49	768924,88
MDG-b-17	500	50	776619,125	771327,89	768815,40
MDG-b-18	500	50	775850,75	771671,41	768336,04
MDG-b-19	500	50	778802,9375	773855,55	772631,79
MDG-b-20	500	50	778644,8125	775098,18	773761,54

Novamente o BRKGA obteve melhores resultados que o GRASP na maioria deles. Somente nos Problemas MDG-b-06, MDG-b-07, MDG-b-08 e MDG-b-12 o GRASP alcançou valores de função objetivo maiores. A melhor solução conhecida para estes problemas não foi superada por nenhuma das duas meta-heurísticas estudadas neste trabalho.

O único problema em que foi possível melhorar a qualidade da solução através do *path relinking* foi MDG-b-09, onde o valor de função objetivo aumentou de 765212,55 para 765432,73. A Tabela 4.4.9 apresenta as taxas que cada solução representa com relação à melhor solução conhecida.

Nos problemas de 1 a 20 do conjunto MDG-b, o tempo médio para obtenção de uma solução no BRKGA foi 5 minutos, enquanto no GRASP foi 0,7 segundos. O tempo praticado no *path relinking* foi 4,9 segundos em cada problema.

Tabela 4.4.9: Taxas com relação a melhor solução conhecida na literatura para o primeiro conjunto de problemas de MDG-b.

Problema	**BRKGA** ($\frac{S_b}{S_m}$)	**GRASP** ($\frac{S_g}{S_m}$)
MDG-b-01	99,57%	99,15%
MDG-b-02	98,76%	98,69%
MDG-b-03	99,45%	99,07%
MDG-b-04	99,63%	99,55%
MDG-b-05	99,36%	99,05%
MDG-b-06	99,13%	99,16%
MDG-b-07	99,43%	99,76%
MDG-b-08	99,57%	99,90%
MDG-b-09	99,24%	98,76%
MDG-b-10	99,53%	99,11%
MDG-b-11	99,52%	99,48%
MDG-b-12	99,13%	99,24%
MDG-b-13	99,91%	98,36%
MDG-b-14	99,81%	99,70%
MDG-b-15	99,50%	98,07%
MDG-b-16	99,81%	99,16%
MDG-b-17	99,32%	99,00%
MDG-b-18	99,46%	99,03%
MDG-b-19	99,36%	99,21%
MDG-b-20	99,54%	99,37%

Vejamos agora os resultados para o segundo conjunto de problemas.

Neste caso, onde as dimensões dos problemas são maiores, os resultados obtidos via GRASP foram melhores que aqueles obtidos via BRKGA em todos os problemas, conforme pode ser verificado na Tabela 4.4.10. Destacamos ainda que o *path relinking* obteve soluções melhores que aquelas pós busca local, em MDG-b-33 e MDG-b-34, com os respectivos acréscimos no valor de função objetivo de 11209498,48020 para 11209691,88512 e 11224825,37475 para 11225031,73859. Mais uma vez não foi possível alcançar o melhor solução conhecida.

Nota-se na Tabela 4.4.11 que em GRASP todas as taxas foram superiores a 99%, enquanto em BRKGA, tal fato ocorreu somente em MDG-b-24, MDG-b-36 e MDG-b-38. Nos demais problemas as taxas encontram-se no intervalo de 98% a 99%.

Nos problemas 21 a 40, o tempo na obtenção de uma solução foi 31 minutos no BRKGA e 55,8 segundos no GRASP. O tempo no *path relinking* foi em média 6 minutos por problema.

Tabela 4.4.10: Soluções para o conjunto de problemas MDG-b com $n = 2000$ e $m = 200$.

Problema	n	m	**Melhor solução** (S_m)	**BRKGA** (S_b)	**GRASP** (S_g)
MDG-b-21	2000	200	11299894,85774	11153637,68317	11216721,30003
MDG-b-22	2000	200	11286775,58894	11167250,23316	11210929,69157
MDG-b-23	2000	200	11299940,86847	11167792,79279	11220182,94250
MDG-b-24	2000	200	11290874,15963	11178946,99337	11203974,74325
MDG-b-25	2000	200	11296066,66070	11178222,54811	11209374,17967
MDG-b-26	2000	200	11292295,88629	11153192,00245	11210951,72447
MDG-b-27	2000	200	11305676,84757	11184380,20130	11208725,19539
MDG-b-28	2000	200	11279916,05595	11152204,39416	11189460,40665
MDG-b-29	2000	200	11297188,33456	11155303,42761	11217481,63976
MDG-b-30	2000	200	11296414,92982	11173438,81211	11232252,57071
MDG-b-31	2000	200	11288901,49168	11152227,71048	11191167,20045
MDG-b-32	2000	200	11279820,12875	11153647,07114	11201626,82164
MDG-b-33	2000	200	11296297,93395	11164346,67471	11209498,48020
MDG-b-34	2000	200	11281245,49982	11151061,68223	11224825,37475
MDG-b-35	2000	200	11307424,25576	11178697,69991	11221786,36026
MDG-b-36	2000	200	11289469,27410	11193494,50504	11229730,57858
MDG-b-37	2000	200	11290544,79258	11149879,28123	11206065,39971
MDG-b-38	2000	200	11288570,85243	11180165,66119	11212438,34579
MDG-b-39	2000	200	11295054,20265	11143042,13626	11226452,59673
MDG-b-40	2000	200	11307104,80347	11156590,00433	11231903,74372

Tabela 4.4.11: Taxas com relação a melhor solução conhecida para o segundo conjunto de problemas de MDG-b.

Problema	**BRKGA** ($\frac{S_b}{S_m}$)	**GRASP** ($\frac{S_g}{S_m}$)
MDG-b-21	98,71%	99,26%
MDG-b-22	98,94%	99,33%
MDG-b-23	98,83%	99,29%
MDG-b-24	99,01%	99,23%
MDG-b-25	98,96%	99,23%
MDG-b-26	98,77%	99,28%
MDG-b-27	98,93%	99,14%
MDG-b-28	98,87%	99,20%
MDG-b-29	98,74%	99,29%
MDG-b-30	98,91%	99,43%
MDG-b-31	98,79%	99,13%
MDG-b-32	98,88%	99,31%
MDG-b-33	98,83%	99,23%
MDG-b-34	98,85%	99,50%
MDG-b-35	98,86%	99,24%
MDG-b-36	99,15%	99,47%
MDG-b-37	98,75%	99,25%
MDG-b-38	99,04%	99,33%
MDG-b-39	98,65%	99,39%
MDG-b-40	98,67%	99,33%

Capítulo 5

Conclusões

Neste trabalho estudamos o Problema da Diversidade Máxima, que busca selecionar entre um conjunto de elementos, um subconjunto que seja o mais diverso possível. Detalhamos a formulação quadrática e a formulação linear inteira mista, onde notamos que a linear possui um número bem maior de variáveis e restrições em comparação com a formulação quadrática.

Observamos ainda que o Problema da Diversidade Máxima é classificado como NP-difícil, o que torna comum o uso de heurísticas para obtenção de uma solução. Através da resolução de alguns problemas, observamos que o tempo de resolução através de métodos exatos cresce exponencialmente conforme os valores de n e m aumentam. Para isso utilizamos o IBM ILOG CPLEX e notamos que para problemas com $n = 40$ e $m = 20$ foram necessárias aproximadamente 20 horas para obter a solução ótima. No caso $n = 50$, não foi possível encontrar uma solução ótima devido ao uso total de memória.

Realizamos uma breve revisão bibliográfica das meta-heurísticas já aplicadas ao problema. A partir dela constatamos que não haviam trabalhos utilizando BRKGA na resolução do PDM. Sendo assim, optamos em estudar uma maneira de aplicá-la ao nosso problema. A outra meta-heurística utilizada, GRASP, foi escolhida devido a sua grande aplicação em diversos trabalhos e os bons resultados apresentados.

Com o objetivo de explorar mais o espaço de soluções do problema, aplicamos o *path relinking*, que busca conectar as soluções obtidas na fase de contrução através de caminhos. Em nosso trabalho adotamos um modelo guloso de *path relinking*, onde a cada etapa realizamos a melhor troca entre os elementos que não são comuns nas duas soluções analisadas. As implementações de GRASP e *path relinking* foram feitas no software MatLab.

Em BRKGA utilizamos o API em C++ desenvolvido por Toso e Resende, implementando uma função *fitness* e um decodificador de soluções adaptados ao Problema da Diversidade Máxima. A função *fitness* adotada é a soma das diversidades entre os elementos selecionados e a decodificação de soluções é baseada na ordenação de chaves aleatórias.

Antes de aplicar BRKGA nos testes computacionais, realizamos uma calibração dos parâmetros p_e, p_m e ρ_e da meta-heurística, onde adotamos o RDI como medida de desempenho. Testamos 60 triplas de parâmetros em um conjunto de 10 problemas, no qual a tripla $p_e = 20\%$, $p_m = 10\%$ e $\rho_e = 70\%$ obteve os melhores resultados, e por isso

utilizamos estes parâmetros nos testes computacionais.

Os problemas utilizados nos testes foram extraídos da biblioteca MDPLIB. Comparamos os resultados obtidos entre as meta-heurísticas GRASP e BRKGA com os melhores resultados da literatura. Os problemas estão divididos em 4 conjuntos: GKD-c, SOM-b, MDG-a e MDG-b.

De maneira geral, tanto GRASP quanto BRKGA obtiveram bons resultados nos problemas analisados, atingindo soluções com valor de função objetivo correspondendo a 98% ou mais da melhor solução conhecida.

Somente em dois problemas obtivemos uma solução com valor de função objetivo maior que a melhor solução conhecida. Ambos fazem parte do conjunto GKD-c e foram obtidos tanto em GRASP, quanto em BRKGA, uma vez que neste conjunto de problemas os valores de função objetivo das soluções de GRASP e BRKGA coincidem em todos os casos. A melhora para estes problemas foi muito pequena, aumento de 0,00022 e 0,0007.

Para os problemas de médio e pequeno porte, cujas dimensões variam de $n = 100$ à $n = 500$, BRKGA atingiu melhores resultados que GRASP na maioria dos casos. Em particular no conjunto SOM-b, para as dimensões $n \leq 200$, BRKGA alcançou a melhor solução conhecida na literatura em todos os casos.

Nos problemas de grande porte, GRASP obteve desempenho melhor que BRKGA na maioria dos casos. Tal fato pode ser evidenciado nos conjuntos MDG-a e MDG-b, onde ele perde para BRKGA nos problemas de tamanho $n = 500$, mas supera-o nos problemas com $n = 2000$.

Notamos que à medida que as dimensões dos problemas aumentam, BRKGA não consegue obter a melhor solução, mas atinge soluções próximas da melhor. Por outro lado, o GRASP tem um efeito contrário, pois não gera a melhor solução sequer nos problemas de pequeno porte. Entretanto, nos problemas de grande porte consegue superar as soluções obtidas via BRKGA na maioria dos casos.

Em apenas 11 problemas de 140, o *path relinking* conseguiu melhorar o valor da função objetivo da melhor solução obtida via fase de construção e busca local. Nestes casos a variação foi muito pequena comparada ao tempo utilizado nessa busca. Tal fato sugere que a busca local do GRASP tem sido eficaz explorando o espaço de busca, encontrando um máximo local na maioria dos casos.

Como trabalhos futuros, podemos citar a utilização de outras heurísticas na fase de construção da solução do GRASP. Propomos tornar mais rápido o API para BRKGA utilizado neste trabalho, ordenando somente as m primeiras chaves aleatórias durante a avaliação da função *fitness*.

Em relação ao *path relinking* sugerimos a sua aplicação somente no par de soluções com maior diferença no valor da função objetivo, uma que que não obtivemos melhoras significativas aplicando em todos os pares possíveis. Além disso, a aplicação do *path relinking* também nas soluções obtidas via BRKGA.

Propomos também a utilização de programas específicos para a calibração dos parâmetros da meta-heurística BRKGA.

Como última sugestão, apontamos o uso de métodos exatos para alimentação das meta-heurísticas aplicadas ao problema.

Referências Bibliográficas

Agrafiotis, D. K. (1997). Stochastic algorithms for maximizing molecular diversity. *Journal of chemical information and computer sciences*, 37(5):841–851.

Aringhieri, R., Cordone, R., e Melzani, Y. (2008). Tabu search versus grasp for the maximum diversity problem. *4OR: A Quarterly Journal of Operations Research*, 6(1):45–60.

Bonotto, E. L. e Cabral, L. d. A. F. (2014). *Algoritmo de otimização por nuvem de partículas aplicado ao problema da diversidade máxima.* Anais do Simpósio Brasileiro de Pesquisa Operacional.

Dhir, K., Glover, F., e Kuo, C.-C. (1993). Optimizing diversity for engineering management. In *Engineering Management Conference, 1993. Managing Projects in a Borderless World. Pre Conference Proceedings., 1993 IEEE International*, pages 23–26. IEEE.

Dorigo, M., Birattari, M., e Stützle, T. (2006). Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39.

Duarte, A. e Martí, R. (2007). Tabu search and grasp for the maximum diversity problem. *European Journal of Operational Research*, 178(1):71–84.

Feo, T. A. e Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71.

Gallego, M., Duarte, A., Laguna, M., e Martí, R. (2009). Hybrid heuristics for the maximum diversity problem. *Computational Optimization and Applications*, 44(3):411–426.

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

Glover, F. (1977). *Selecting subsets of maximum diversity.* Business Research Division, Graduate School of Business Administration, University of Colorado.

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.

Glover, F., Laguna, M., e Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684.

Glover, F., Laguna, M., Taillard, E., e de Werra, D. (1993). *Tabu search*. Baltzer Basel.

Gonçalves, J. F. e Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Harary, F. (1969). *Graph theory*. Addison-Wesley, Reading, MA.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press.

Kim, Y.-D. (1993). Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of the Operational Research Society*, 44(1):19–28.

Kincaid, R. K. (1992). Good solutions to discrete noxious location problems via metaheuristics. *Annals of Operations Research*, 40(1):265–281.

Kirkpatrick, S., Vecchi, M. P., et al. (1983). Optimization by simmulated annealing. *Science*, 220(4598):671–680.

Kochenberger, G. e Glover, F. (1999). *Diversity data mining*. University of Mississippi.

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

Laguna, M. e Martí, R. (2012). *Scatter search: methodology and implementations in C*. Springer Science & Business Media.

Lopes, S. e Rosso, S. (2005). Biologia: volume único. *São Paulo: Saraiva*.

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.

Reeves, C. R. (1997). Feature article-genetic algorithms for the operations researcher. *INFORMS Journal on Computing*, 9(3):231–250.

Resende, M. G., Martí, R., Gallego, M., e Duarte, A. (2010). GRASP and path relinking for the max–min diversity problem. *Computers & Operations Research*, 37(3):498–508.

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

Toso, R. F. e Resende, M. G. (2015). A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93.

Wolsey, L. A. (1998). *Integer programming*. Wiley New York.

Zäpfel, G. e Braune, R. (2010). *Metaheuristic search concepts: A tutorial with applications to production and logistics*. Springer Science & Business Media.