

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Otimização por Nuvem de Partículas e Busca Tabu para o
Problema da Diversidade Máxima

Edison Luiz Bonotto

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Computação Distribuída

Lucídio dos Anjos Formiga Cabral
(Orientador)

João Pessoa, Paraíba, Brasil
©Edison Luiz Bonotto, Janeiro de 2017

B719o Bonotto, Edison Luiz.
Otimização por Nuvem de Partículas e Busca Tabu para
Problema da Diversidade Máxima / Edison Luiz Bonotto. - João
Pessoa, 2017.
64 f. : il. -

Orientador: Lucídio dos Anjos Formiga Cabral.
Dissertação (Mestrado) - UFPB/CI

1. Informática. 2. MDP - Problema da Diversidade Máxima.
3. PSO - Otimização por Nuvem de Partículas. 4. Busca Tabu
(TS). 5. Otimização combinatória. 6. Problema $\mathcal{NP}$ -Difícil.
I. Título.

UFPB/BC

CDU: 004(043)



UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA



Ata da Sessão Pública de Defesa de Dissertação de
Mestrado de EDISON LUIZ BONOTTO, candidato ao título
de Mestre em Informática na Área de Sistemas de
Computação, realizada em 31 de janeiro de 2017.

1 Aos trinta e um dias do mês de janeiro, do ano de dois mil e dezessete, às quatorze horas e
2 trinta minutos, no Centro de Informática da Universidade Federal da Paraíba, em
3 Mangabeira, reuniram-se os membros da Banca Examinadora constituída para julgar o
4 Trabalho Final do Sr. Edison Luiz Bonotto, vinculado a esta Universidade sob a matrícula nº
5 2015103939, candidato ao grau de Mestre em Informática, na área de "Sistemas de
6 Computação", na linha de pesquisa "Computação Distribuída", do Programa de Pós-
7 Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora
8 foi composta pelos professores: Lucídio dos Anjos Formiga Cabral (PPGI-UFPB), Orientador
9 e Presidente da Banca, Anand Subramanian (PPGI-UFPB), Examinador Interno ao
10 Programa, Roberto Quirino do Nascimento (UFPB), Examinador Externo ao Programa, e
11 Marcelo Lisboa Rocha, Examinador externo à instituição (UFT). Dando início aos trabalhos, o
12 Presidente da Banca, cumprimentou os presentes, comunicou aos mesmos a finalidade da
13 reunião e passou a palavra ao candidato para que o mesmo fizesse a exposição oral do
14 trabalho de dissertação intitulado "Otimização por nuvem de partículas e busca tabu para o
15 problema da diversidade máxima". Concluída a exposição, o candidato foi arguido pela
16 Banca Examinadora que emitiu o seguinte parecer: "**aprovado**". Do ocorrido, eu, Claurton
17 de Albuquerque Siebra, Coordenador do Programa de Pós-Graduação em Informática,
18 lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora.
19 João Pessoa, 31 de janeiro de 2017.

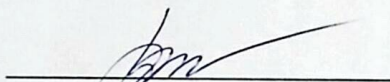
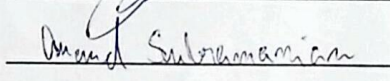
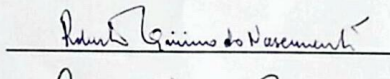
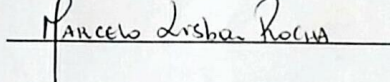
Prof. Dr. Claurton de Albuquerque Siebra

Prof. Dr. Lucídio dos Anjos Formiga Cabral
Orientador (PPGI-UFPB)

Prof. Dr. Anand Subramanian
Examinador interno ao programa (PPGI-UFPB)

Prof. Dr. Roberto Quirino do Nascimento
Examinador externo ao Programa (UFPB)

Prof. Dr. Marcelo Lisboa Rocha
Examinador externo à instituição (UFT)

Resumo

O Problema da Diversidade Máxima (MDP) é um problema da área de Otimização Combinatória que tem por objetivo selecionar um número pré-estabelecido de elementos de um dado conjunto de maneira tal que a soma das diversidades entre os elementos selecionados seja a maior possível. O MDP pertence a classe de problemas $\mathcal{NP}$ -difícil, isto é, não existe algoritmo conhecido que o resolva de forma exata em tempo polinomial. Por apresentarem uma complexidade de ordem exponencial, exigem heurísticas eficientes que forneçam resultados satisfatórios em tempos aceitáveis. Entretanto, as heurísticas não garantem otimalidade da solução encontrada. Este trabalho propõe uma nova abordagem híbrida para a resolução do Problema da Diversidade Máxima e está baseada nas meta-heurísticas de Otimização por Nuvem de Partículas (PSO) e Busca Tabu(TS). O algoritmo foi denominado **PSO_TS**. Para a validação do método, os resultados encontrados são comparados com os melhores existentes na literatura.

Palavras-chave: Problema da Diversidade Máxima (MDP), Otimização por Nuvem de Partículas (PSO), Busca Tabu (TS), Otimização Combinatória, Problema $\mathcal{NP}$ -Difícil.

Abstract

The Maximum Diversity Problem (MDP) is a problem of combinatorial optimization area that aims to select a pre-set number of elements in a given set so that a sum of the differences between the selected elements are greater as possible. MDP belongs to the class of $\mathcal{NP}$ -Hard problems, that is, there is no known algorithm that solves in polynomial time accurately. Because they have a complexity of exponential order, require efficient heuristics to provide satisfactory results in acceptable time. However, heuristics do not guarantee the optimality of the solution found. This paper proposes a new hybrid approach for a resolution of the Maximum Diversity Problem and is based on the Particle Swarm Optimization (PSO) and Tabu Search (TS) metaheuristics, The algorithm is called **PSO_TS**. The use of PSO_TS achieves the best results for known instances testing in the literature, thus demonstrating be competitive with the best algorithms in terms of quality of the solutions.

Maximum Diversity Problem (MDP), Particle Swarm Optimization (PSO), Tabu Search (TS), Combinatorial Optimization, $\mathcal{NP}$ -Hard Problem.

Agradecimentos

A minha mãe, Santa e meu pai, José Lir, que possibilitaram que eu tivesse a educação necessária para cursar este mestrado.

A minha esposa, Daiane, e meus filhos, Lucas e Luiz Felipe, pelo apoio e compreensão nos momentos em que precisei estar ausente e que me acompanharam pacientemente durante todo este processo de mestrado.

Ao Professor Marcelo Lisboa, pela motivação, me fazendo acreditar que era possível.

Ao professor Lucídio Cabral, pela orientação fornecida e por toda ajuda ao longo do processo.

Ao colega Eduardo Queiroga e ao professor Anand Subramanian, que muito contribuíram para a finalização deste trabalho.

Conteúdo

1	INTRODUÇÃO	1
1.1	Motivação	1
1.2	Objetivos	3
1.2.1	Objetivo Geral	3
1.2.2	Objetivos Específicos	3
1.3	Estrutura da Dissertação	3
2	Fundamentação Teórica	5
2.1	O Problema da Diversidade Máxima (<i>Maximum Diversity Problem</i> - MDP)	5
2.1.1	Aplicações Práticas do MDP	6
2.1.2	Exemplo	7
2.1.3	Formulação Matemática	9
2.2	Métodos Heurísticos	12
2.2.1	Heurísticas de Construção	12
2.2.2	Heurísticas de Melhoria	13
2.3	Otimização por Nuvem de Partículas (<i>Particle Swarm Optimization</i> - PSO)	13
2.4	Busca Tabu (<i>Tabu Search</i> - TS)	15
3	Trabalhos Relacionados	20
4	Método Proposto	27
4.1	Algoritmo PSO_TS	27
4.2	Algoritmos de Construção	31

4.2.1	Heurística de Construção Gulosa	31
4.2.2	Heurística de Construção Aleatória	32
4.3	Busca Tabu	33
5	Testes e Resultados	37
5.1	Ambiente de Desenvolvimento	37
5.1.1	Hardware	37
5.1.2	Software	37
5.1.3	Base de dados	38
5.2	Definição dos Parâmetros	39
5.3	Testes e Resultados	40
5.3.1	Resultados das Instâncias de Médio Porte	40
5.3.2	Resultados Instâncias de Grandes Porte	42
6	Conclusão	48
	Referências Bibliográficas	54

Lista de Siglas

MDP	: <i>Maximum Diversity Problem</i>
PSO	: <i>Particle Swarm Optimization</i>
LS	: <i>Local Search</i>
TS	: <i>Tabu Search</i>
ILS	: <i>Iterated Local Search</i>
ITS	: <i>Iterated Tabu Search</i>
IG	: <i>Iterated Greedy</i>
GRASP	: <i>Greedy Randomized Adaptive Search Procedure</i>
MA	: <i>Memetic Algorithm</i>
SA	: <i>Simulated Annealing</i>
SS	: <i>Scatter Search</i>
ME	: <i>Métodos Exatos</i>
FM	: <i>Formulação Matemática</i>

Lista de Figuras

2.1	Interpretação geométrica de uma partícula no plano bi-dimensional . .	15
2.2	Fluxograma do algoritmo de Otimização por Nuvem de Partículas . . .	16
2.3	Representação gráfica de máximos ótimos locais e ótimo global	18
5.1	Comparação entre a melhor solução do PSO_TS e outros algoritmos <i>estado-da-arte</i> (Instâncias MDG-a)	44
5.2	Comparação entre a solução média do PSO_TS e outros algoritmos <i>estado-da-arte</i> (Instâncias MDG-a)	44
5.3	Comparação das instâncias MDG-c entre PSO_TS e outros algoritmos <i>estado-da-arte</i>	46

Lista de Tabelas

2.1	Mapeamento de atributos pessoais	7
2.2	Atributos mapeados dos candidatos	8
2.3	Matriz de distâncias entre cada par de candidatos	8
2.4	Diversidade máxima com todas as soluções possíveis	9
2.5	Comparação entre as Formulações F1, F2 e F3 [Martí, Gallego e Duarte 2010]	12
3.1	Principais abordagens da literatura para o MDP	26
5.1	Características das instâncias MDG-c	38
5.2	Valores definidos para as constantes - Parte 1	39
5.3	Valores definidos para as constantes - Parte 2	40
5.4	Resultados das instâncias SOM-b (Tempo $< 1s$)	41
5.5	Resultados das instâncias MDG-a (Tempo $\leq 16s$)	42
5.6	Resultados instância MDG-c (tempo $\leq 480s$)	45

Lista de Algoritmos

1	Pseudocódigo do algoritmo PSO GBest	16
2	Pseudocódigo de um algoritmo básico de Busca Tabu [Souza 2010] . . .	18
3	Pseudocódigo do Algoritmo PSO_TS	30
4	Pseudocódigo da Heurística de Construção Gulosa	32
5	Pseudocódigo da Heurística de Construção Aleatória	33
6	Pseudocódigo da Busca Tabu (Baseado em [Wang et al. 2014])	35

Capítulo 1

INTRODUÇÃO

1.1 Motivação

Problemas de otimização são comuns na vida diária e aparecem quando procura-se determinar o melhor caminho para chegar a um destino, o nível de produção mais econômico de uma fábrica, a equipe mais heterogênea possível para uma determinada tarefa, etc.

Problemas com alto grau de complexidade surgem todos os dias e os pesquisadores necessitam resolvê-los e tem-se, assim, a necessidade de buscar soluções viáveis. Técnicas de otimização são amplamente utilizadas em problemas como de escalonamento, alocação de lugares e clusterização de dados, exemplos de problemas reais que requerem um tratamento adequado.

Há uma grande dificuldade para encontrar o método adequado na resolução dos problemas de otimização, sendo que os métodos convencionais, exatos (principalmente os que executam buscas exaustiva), geralmente são ineficientes devido ao alto custo computacional para encontrar uma solução ótima para problemas de grandes dimensões. Os métodos avançados de otimização, baseados em meta-heurísticas, tornam os processos ágeis e obtêm soluções de alta qualidade com tempos de processamento viáveis, reduzindo os custos de produção.

Meta-heurística é um termo usado para descrever um subcampo primário da área de otimização estocástica, classe geral de algoritmos e técnicas que empregam algum grau de aleatoriedade para encontrar soluções tão ótimas quanto possível para problemas

reconhecidamente difíceis [Luke 2013]. Utilizam combinação de escolhas aleatórias e conhecimento histórico dos resultados anteriores adquiridos pelo método para se guiarem e realizar suas buscas pelo espaço de pesquisa.

A meta-heurística Busca Tabu, segundo [Souza 2010], é uma técnica para guiar uma heurística de busca local tradicional na exploração do espaço de soluções além da otimalidade local, usando para isso, basicamente, estruturas de memória. Parte de uma solução inicial e se move no espaço de soluções de uma solução para outra que esteja em sua vizinhança.

A Otimização por Nuvem de Partículas é uma meta-heurística que pretende simular graficamente o comportamento de um conjunto de pássaros em vôo com seu movimento localmente aleatório, mas globalmente determinado [Kennedy e Eberhart 1995]. Cada partícula se move pelo espaço de busca sofrendo influência da sua melhor posição já visitada, da melhor posição já visitada por qualquer uma das partículas e pela velocidade de inércia da própria partícula.

O Problema da Diversidade Máxima (MDP) é um problema de otimização combinatória que está relacionado à variedade de características, ideias ou elementos diferentes entre si dentro de um determinado contexto, sendo importante para o pluralismo, heterogeneidade, tolerância mútua e sobrevivência de ideias [Takane 2011]. Existem diversos tipos de diversidades em diferentes áreas do conhecimento humano. Entre eles, pode-se citar a diversidade religiosa, social, linguística, sexual, cultural e biológica [Takane 2011].

Dado que o MDP é um problema $\mathcal{NP}$ -Difícil [Kuo, Glover e Dhir 1993], esse trabalho propõe o desenvolvimento de um algoritmo híbrido baseado na abordagem Otimização por Nuvem de Partículas e um algoritmo de Busca Tabu para a resolução do do referido problema. Experimentos são feitos, sendo o algoritmo executado com diferentes tipos de instâncias e os resultados são comparados com os melhores resultados existentes na literatura e com outras heurísticas estado-da-arte.

1.2 Objetivos

Nesta seção serão apresentados o objetivo geral e os objetivos específicos que orientaram a realização deste trabalho.

1.2.1 Objetivo Geral

- Demonstrar através de testes e resultados computacionais que a abordagem desenvolvida para a resolução do Problema da Diversidade Máxima (MDP) baseada nas meta-heurísticas Otimização por Nuvem de Partículas (PSO) e Busca Tabu (TS) proporciona melhores resultados, no tocante a qualidade da solução e ao tempo computacional, que as técnicas competidoras da literatura.

1.2.2 Objetivos Específicos

- Realizar estudos e desenvolver um método de Otimização por Nuvem de Partículas aplicado ao MDP.
- Realizar estudos e desenvolver uma técnica de refinamento das soluções através da implantação de um método de Busca Tabu.
- Desenvolver um algoritmo híbrido baseado em PSO e TS (PSO_TS).
- Verificar a eficiência do método proposto verificando os resultados obtidos levando em consideração a qualidade das soluções e o tempo de processamento.
- Avaliar a sensibilidade da parametrização do PSO em termos de qualidade das soluções.

1.3 Estrutura da Dissertação

O trabalho está dividido em cinco capítulos organizado da seguinte forma:

- O Capítulo 1 é referente a introdução.

-
- O Capítulo 2 apresenta a fundamentação teórica do trabalho, com a descrição dos conceitos básicos do método principal, que é a utilização da meta-heurística Otimização por Nuvem de Partículas, estudos do método empregado para o refinamento, a meta-heurística Busca Tabu e de todos os métodos necessários para o andamento deste trabalho.
 - O Capítulo 3 contém a revisão de literatura referente ao Problema da Diversidade Máxima.
 - O Capítulo 4 descreve o método proposto PSO_TS.
 - O Capítulo 5 apresenta os experimentos computacionais realizados e os resultados obtidos.
 - O Capítulo 6 é a conclusão sobre o trabalho realizado.

Capítulo 2

Fundamentação Teórica

Este capítulo está dividido em quatro seções. Na primeira seção tem-se uma descrição do Problema da Diversidade Máxima; na segunda seção, há uma introdução aos métodos Heurísticos; na terceira seção, apresentamos um estudo sobre a Otimização por Nuvem de Partículas e, na quarta seção, faz-se uma apresentação do procedimento de Busca Tabu.

2.1 O Problema da Diversidade Máxima (*Maximum Diversity Problem* - MDP)

O Problema da Diversidade Máxima, segundo [Duarte, Silva e Costa 2008], é um problema de Otimização Combinatória que tem por objetivo selecionar um número pré-estabelecido de elementos de um dado conjunto de tal maneira que a soma das diversidades entre os elementos selecionados seja a maior possível, ou seja, consiste em selecionar um subconjunto M com m elementos de um conjunto N , onde $N = \{1, \dots, n\}$, de tal forma que a diversidade entre os seus elementos selecionados seja máxima.

O MDP é um problema NP-Difícil [Kuo, Glover e Dhir 1993], isto é, não existe algoritmo conhecido que o resolva de forma exata em tempo polinomial e, à medida que o tamanho das instâncias aumenta, torna-se proibitivo o tempo necessário para que estas sejam resolvidas. Segundo [Wu e Hao 2013], o problema também é conhecido por *maximum dispersion*, *MAX-AVG dispersion*, *edge-weighted clique*, *remote-clique*, *maximum*

edge-weighted sub- graph e dense k-subgraph.

Para calcular a diversidade como uma distância é comum usar uma métrica normalizada entre os atributos dos elementos de N . Uma métrica de distância normalmente utilizada é a norma Euclidiana. Usamos a *Equação (2.1)* para definir a distância euclidiana entre os elementos i e j denotada por d_{ij} , onde $p = 2$, k pertence ao conjunto R , R é o conjunto dos atributos (*Tabela 2.1*), a_{ik} denota o k -ésimo atributo do elemento i e a_{jk} denota o k -ésimo atributo do elemento j .

$$d_{ij} = \sqrt[p]{\sum_{k \in R} (|a_{ik} - a_{jk}|)^p} \quad (2.1)$$

2.1.1 Aplicações Práticas do MDP

O problema da Diversidade Máxima é utilizado para resolver uma grande gama de problemas da vida real. Nos itens que seguem há diversos exemplos, os quais foram citados por [Kuo, Glover e Dhir 1993]:

- Políticas de admissão, como na seleção de universitários, onde um pesquisador que deseja pesquisar uma amostra representativa de indivíduos possuindo um amplo espectro de características. Muitas faculdades e universidades, ao formularem suas políticas de admissão, vão além da pontuação do teste e da classificação da turma e também consideram outros fatores na busca de um corpo estudantil diversificado [Bunzel e Au 1987] [Ramirez 1979] *apud* [Kuo, Glover e Dhir 1993].
- Políticas de imigração, como na reforma da imigração norte-americana, onde o congresso estava preocupado em promover a diversidade étnica entre os imigrantes [McConnell 1988] [McConnell 2013].
- Planejamento de mercado, onde frequentemente é desejável maximizar o número e a diversidade dos pontos fortes no perfil de uma marca [Keely 1989].
- Melhoramento de plantas [Porter W. M.. Rawal 1975] *apud* [Kuo, Glover e Dhir 1993].
- Problemas sociais, onde estudos históricos sobre a diversidade étnica podem desenvolver modelos estruturais do sistema institucional de uma nação [Swierenga

1977].

- Preservação ecológica, resolvendo problemas de diversidade genética [Unkel 1985] [Pearce 1987].
- Seleção de portfólios [Ghyczy 1985] [Besanko, Perry e Spady 1990].
- Gestão da mão-de-obra, através do gerenciamento da diversidade como forma de obter uma força de trabalho heterogênea, aumentando a produtividade, o comprometimento, a qualidade e o lucro [Thomas 1990].

A maximização da diversidade também é uma questão importante nas seguintes áreas: contabilidade e auditoria, experimentação química, projeto experimental, pesquisa médica, exploração geológica, seleção de carteira e engenharia estrutural [Kuo, Glover e Dhir 1993]. Também há aplicações práticas no campo de tratamento médico, balanceamento ambiental, serviço de comunicações e engenharia genética [Glover, Kuo. e Dhir 1998].

2.1.2 Exemplo

Nesta seção, mostraremos um exemplo de aplicação do MDP a fim de melhorar a compreensão do problema. Suponha que uma seção de recursos humanos de uma determinada empresa queira selecionar três funcionários dentre cinco pré-selecionados para compor uma equipe de trabalho e deseja que esta equipe tenha os mais diversos atributos possíveis. Inicialmente, são definidos quais atributos são considerados importantes para que esta equipe seja a mais heterogênea possível. Em seguida, define-se um peso para cada atributo, de acordo com sua relevância.

Clas- sifi- cação	Atributos				
	Sexo	Idade	Região de Origem	Áreas de Conhecimento	Experiên- cia (anos)
1	Masc.	16 - 19	norte	ciências exatas	0 - 1
2	Fem.	20 - 23	centro-oeste	ciências humanas	2 - 3
3		24 - 27	nordeste	ciências sociais	4 - 5
4		28 - 31	sul	ciências da saúde	6 - 7
5		31 - 34	sudeste	ciências agrária	8 - 10

Tabela 2.1: Mapeamento de atributos pessoais

A *Tabela 2.1* mostra o mapeamento dos atributos e a *Tabela 2.2* mostra o mapeamento dos atributos dos candidatos, onde cada linha representa um candidato com seus respectivos atributos.

Funcionário	Atributos				
	Sexo	Idade	Região de Origem	Áreas de Conhecimento	Experiência (anos)
A	1	2	2	2	1
B	2	3	3	2	1
C	2	1	1	3	2
D	1	4	5	5	3
E	2	4	5	4	4

Tabela 2.2: Atributos mapeados dos candidatos

Com base na *Tabela 2.2*, calculamos a distância entre cada par de candidatos. A *Tabela 2.3* mostra os valores resultantes do cálculo da distância Euclidiana usando a *Equação (2.1)*.

	A	B	C	D	E
A	0,00	1,73	2,24	5,10	5,20
B	1,73	0,00	3,16	4,36	4,24
C	2,24	3,16	0,00	5,57	5,48
D	5,10	4,36	5,57	0,00	1,73
E	5,20	4,24	5,48	1,73	0,00

Tabela 2.3: Matriz de distâncias entre cada par de candidatos

A solução ótima para o problema é calculada analisando todas as combinações possíveis dos cinco candidatos agrupados de três em três e escolhendo os que apresentarem a maior diversidade entre eles. Neste caso teríamos 10 combinações possíveis.

$$\binom{5}{3} = \frac{5!}{3!2!} = \frac{5 \times 4 \times 3}{3 \times 2 \times 1} = \frac{60}{6} = 10$$

A *Tabela 2.4* apresenta os grupos juntamente com a diversidade total entre eles. A solução do problema (solução ótima) seria a escolha dos candidatos *B*, *C* e *D*, pois apresentam a maior diversidade entre eles.

Combinações		Diversidade Total	
ABC	AB + AC + BC	1,73 + 2,24 + 3,16	7,13
ABD	AB + AD + BD	1,73 + 5,10 + 4,36	11,19
ABE	AB + AE + BE	1,73 + 5,20 + 4,24	11,17
ACD	AC + AD + CD	2,24 + 5,10 + 5,57	12,91
ACE	AC + AE + CE	2,24 + 5,10 + 5,48	12,82
ADE	AD + AE + DE	5,10 + 5,20 + 1,73	12,03
BCD	BC + BD + CD	3,16 + 4,36 + 5,57	13,09
BCE	BC + BE + CE	3,16 + 4,24 + 5,48	12,88
BDE	BD + CE + DE	4,36 + 5,48 + 1,73	11,57
CDE	CD + CE + DE	5,57 + 5,48 + 1,73	12,78

Tabela 2.4: Diversidade máxima com todas as soluções possíveis

2.1.3 Formulação Matemática

O problema de diversidade máxima pode ser modelado como um programa inteiro quadrático 0-1. Tendo como objetivo maximizar a soma das diversidades de um subconjunto de elementos selecionados a partir de um conjunto maior, [Kuo, Glover e Dhir 1993] apresentaram três formulações inteiras para para o MDP. Considerando que $x_i, i \in N$, uma variável binária que toma o valor 1 se o elemento i é selecionado e 0 caso contrário, e $d_{i,j}$ representando a diversidade entre i e $j, \forall i, j \in N$.

Formulação **F1**

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} x_i x_j, \quad (2.2)$$

$$s.a. \sum_{i \in N} x_i = m, \quad (2.3)$$

$$x_i \in \{0, 1\}, \quad \forall i = 1, \dots, n \quad (2.4)$$

Conforme pode-se observar no modelo, a formulação **F1** é um modelo não linear devido a sua função objetivo quadrática. A função objetivo (2.2) maximiza a soma das diversidades entre os m elementos a serem selecionados no conjunto N . Na restrição (2.3) a igualdade exige que exatamente m elementos façam parte de uma solução viável e em (2.4) as restrições indicam que as variáveis do problema devem ser binárias.

[Kuo, Glover e Dhir 1993] também descreveram um modelo com uma abordagem linear 0-1, apresentando o MDP como um Problema de Programação Linear Inteira.

Formulação F2

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} y_{ij}, \quad (2.5)$$

$$s.a. \sum_{i \in N} x_i = m, \quad (2.6)$$

$$x_i + x_j - y_{ij} \leq 1, \quad \forall i, j \in N : i < j, \quad (2.7)$$

$$-x_i + y_{ij} \leq 0, \quad \forall i, j \in N : i < j, \quad (2.8)$$

$$-x_j + y_{ij} \leq 0, \quad \forall i, j \in N : i < j, \quad (2.9)$$

$$y_{ij} \geq 0, \quad \forall i, j \in N : i < j, \quad (2.10)$$

$$x_i \in \{0, 1\}, \quad \forall i = 1, \dots, n \quad (2.11)$$

Na formulação F2 a função objetivo (2.5) maximiza a soma das diversidades dos m elementos de N . A restrição (2.6) exige que exatamente m elementos sejam selecionados. As restrições (2.7) fazem com que as variáveis y_{ij} sejam igual a 1, sempre que x_i e x_j forem iguais a 1. As restrições (2.8) exigem que no caso em que x_i pertencer à solução então deverá existir algum j tal que $y_{ij} = 1$. As restrições (2.9) exigem que no caso em que x_j pertencer à solução então deverá existir algum i tal que $y_{ij} = 1$. As restrições (2.10) restringem o domínio das variáveis y_{ij} a valores não-nulos e as restrições (2.11) indicam que as variáveis do problema devem ser binárias.

Formulação F3

$$\max \sum_{i=1}^{n-1} w_i, \quad (2.12)$$

$$s.a. \sum_{i \in N} x_i = m, \quad (2.13)$$

$$-U_i x_i + w_i \leq 0, \quad \forall i = 1, \dots, n-1 \quad (2.14)$$

$$-\sum_{j=i+1}^n d_{ij} x_j + L_i \times (1 - x_i) + w_i \leq 0, \quad \forall i = 1, \dots, n-1 \quad (2.15)$$

$$w_i \in \mathbb{R}, \quad \forall i = 1, \dots, n \quad (2.16)$$

$$x_i \in \{0, 1\}, \quad \forall i = 1, \dots, n \quad (2.17)$$

Na formulação *F3*, também apresentada por [Kuo, Glover e Dhir 1993], a relaxação linear contínua pode ser resolvida de forma mais eficiente do que formulação *F2*. No entanto, a formulação *F3* não produz uma programação linear tão restritiva quanto a anterior.

Para o MDP, sejam $U_i = \sum_{j=i+1}^n \max(0, d_{ij})$ e $L_i = \sum_{j=i+1}^n \min(0, d_{ij})$ e seja $w_i, i \in N$, uma variável tal que $w_i = \sum_{j=i+1}^n d_{ij} x_j$, se $x_i = 1$ e $w_i \leq 0$, se $x_i = 0$, então a formulação *F1* pode ser escrita conforme a formulação *F3*. Em (2.12), a função objetivo, que maximiza a soma das diversidades dos m elementos de N , garante o mesmo valor que a função objetivo (2.2) da *formulação 1*. A restrição (2.13), assim como nas outras duas formulações, exige que exatamente m elementos sejam selecionados. As restrições (2.14) fazem com que $w_i \leq 0$, quando $x_i = 0$, enquanto que as restrições (2.15) garantem que $w_i = \sum_{j=i+1}^n d_{ij} x_j$, quando $x_i = 1$. As restrições (2.16) definem o domínio da variável $w_i, i \in N$ e as restrições (2.17) indicam que as variáveis do problema devem ser binárias.

As três formulações foram comparadas por [Martí, Gallego e Duarte 2010], sendo executados 5 testes sobre cada uma das 75 instâncias proposta por [Glover, Kuo. e Dhir 1998] por um tempo máximo de 3600s e utilizando o software CPLEX 8.0. Conforme pode-se observar na *Tabela 2.5*, a formulação *F3* conseguiu melhores resultados, encontrando a solução ótima em todos os testes sobre as instâncias testadas, com um tempo e GAP médio inferiores as formulações *F1* e *F2*.

	Formulação		
	F1	F2	F3
GAP Médio (%)	0,70	0,80	0,00
Soluções Ótimas	65	66	75
Tempo Médio (s)	592,6	711,7	96,9

Tabela 2.5: Comparação entre as Formulações F1, F2 e F3 [Martí, Gallego e Duarte 2010]

2.2 Métodos Heurísticos

Os métodos heurísticos tem por finalidade encontrar uma solução para resolver um determinado problema sem, entretanto, garantir que seja a melhor solução. As heurísticas agem reduzindo o espaço de busca do problema e, apesar de não garantirem a otimalidade na solução, apresentam boa qualidade, levando em conta o esforço computacional. Normalmente são empregadas para resolveram problemas NP-Difíceis, onde não existem algoritmos exatos que os resolvam em tempo polinomial.

O algoritmo proposta utiliza os dois principais tipos de heurísticas, as heurísticas de construção e as heurísticas de melhoria.

2.2.1 Heurísticas de Construção

As heurísticas de construção são métodos que tem por finalidade construir uma solução inicial para um determinado problema, escolhendo um elemento de cada vez, seguindo algum critério heurístico de otimização, até que se tenha uma solução viável.

As maneiras utilizadas para escolha destes elementos varia de acordo com o o problema abordado, sendo que os dois principais métodos de construção são o método aleatório e o método guloso.

Os métodos de construção aleatórios tem como principal características conseguirem produzir soluções diversificadas. A escolha dos elementos que farão parte da solução é feita de maneira totalmente aleatória dentre os elemento que ainda não foram selecionados.

Os métodos de construção gulosos visam produzir soluções iniciais de boa qualidade, escolhendo o elemento a ser inserido na solução em razão da melhoria na função objetivo, calculando qual elemento trará mais contribuição a solução levando-se em conta

os elementos já selecionados.

2.2.2 Heurísticas de Melhoria

As chamadas heurísticas de melhoria, refinamento ou de busca em vizinhança partem, necessariamente, de uma solução inicial viável, por sua vez tida como solução corrente, e tentam melhorar esta solução através de operações de troca, remoção ou inserção de elementos até um determinado critério de parada.

Seja $s \in S$, onde S possui um espaço de soluções possíveis, tem-se que $N(s)$ representa a estrutura de vizinhança ou conjunto de vizinhos de s , tal que $N(s) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de s , onde cada *movimento* é denominado mudança m que transforma uma solução s em s' , pertencente a sua vizinhança. Diferentes estruturas de vizinhanças podem ser definidas para cada problema de otimização combinatória.

2.3 Otimização por Nuvem de Partículas (*Particle Swarm Optimization - PSO*)

A Otimização por Nuvem de Partículas (*Particle Swarm Optimization - PSO*) é, segundo [Souza 2006], uma técnica de computação desenvolvida por James Kennedy e Russell Eberhart, [Kennedy e Eberhart 1995], na qual pretendiam simular graficamente o comportamento colaborativo de um bando de pássaros em voo, com seu movimento localmente aleatório, mas globalmente determinado. É baseada na inteligência de enxames, onde o comportamento coletivo dos indivíduos em uma população causa soluções simples coerentes ou surgem padrões.

O algoritmo de otimização é um tipo de inteligência de enxame inspirado no comportamento de bandos de pássaros. A busca por alimentos e a interação entre aves ao longo do voo são modeladas como um mecanismo de otimização. Fazendo uma analogia, a área sobrevoada é equivalente ao espaço de busca e encontrar o local com comida corresponde a encontrar a solução ótima. O algoritmo é modelado por pássaros (chamados de partículas) que fazem uso de sua experiência e da experiência do próprio

bando para encontrar a melhor região do espaço de busca.

No PSO, cada partícula inicia um voo aleatório ou pré-determinado. Entretanto, a cada iteração, o voo é ajustado de acordo com a sua própria experiência, indo em direção a melhor posição encontrada pela partícula ($PBest$, que representa o fator de individualidade) e da experiência de todas as outras partículas, voando em direção a melhor posição encontrada por qualquer uma das partículas ($GBest$). A experiência externa à partícula é denominada fator de sociabilidade. A nova posição é definida baseada nestes dois fatores e na velocidade de inércia da própria partícula.

O PSO original contém duas versões que se distinguem em relação à vizinhança utilizada: O PSO $GBest$ e o PSO $PBest$. Enquanto no PSO $PBest$ todas as partículas fazem parte da vizinhança, no PSO $GBest$ a vizinhança é composta por bairros que são previamente determinados. Como consequência, o algoritmo que utiliza todas as partículas como vizinhos tende a ter uma convergência mais rápida em direção ao máximo global, mas isto diminui a diversidade das partículas. Como o PSO $PBest$ tem uma maior diversidade, fica menos provável ficar preso em máximos locais.

No algoritmo PSO $GBest$, que servirá de base para a implementação para a proposta deste trabalho, inicialmente as posições das partículas X_{ij} são escolhidas aleatoriamente e a velocidade das partículas V_{ij} são nulas. A velocidade de deslocamento da partícula é feito segundo a Equação (2.18).

$$V_{ij}^{k+1} = w.V_{ij}^k + c_1.r_{1j}.(PBest_{ij}^k - X_{ij}^k) + c_2.r_{2j}.(GBest_j^k - X_{ij}^k) \quad (2.18)$$

Onde:

- V_{ij}^k é a velocidade da partícula i na dimensão j na k -ésima iteração;
- X_{ij}^k é a posição da partícula i na dimensão j na k -ésima iteração;
- c_1 é uma constante de aceleração positiva usada para balancear a contribuição dos componentes cognitivo;
- c_2 também é constante de aceleração positiva, mas é utilizada para balancear a contribuição do componente social;
- r_{1j} e r_{2j} são valores randômicos no intervalo $[0, 1]$ distribuídos uniformemente, e;

- w é uma constante que representa o peso inercial na velocidade da partícula. Esta constante não existia no PSO GBest original.

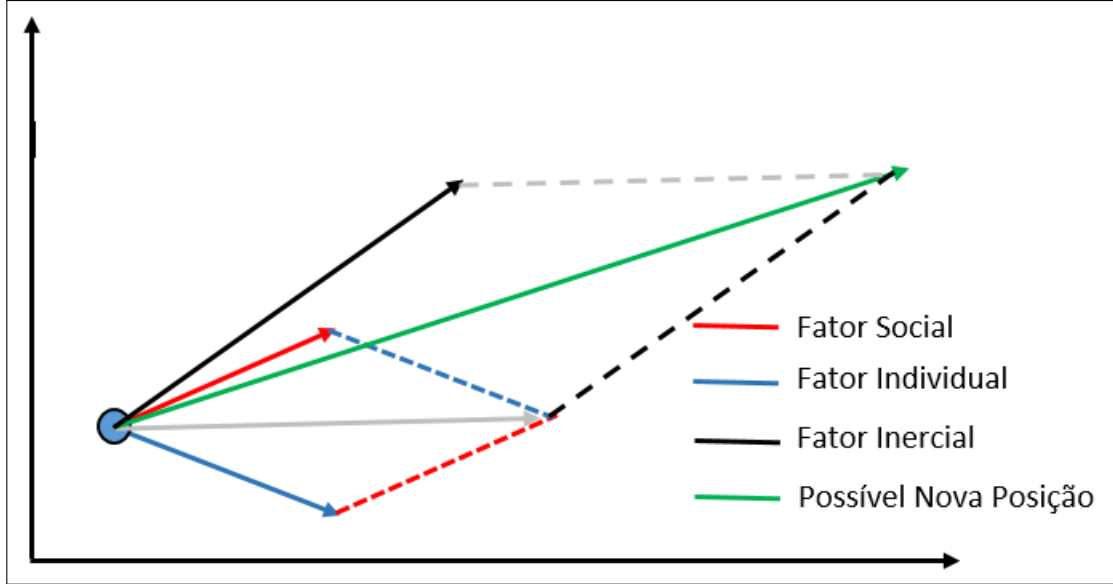


Figura 2.1: Interpretação geométrica de uma partícula no plano bi-dimensional

A nova posição da partícula se dá pela soma do novo vetor velocidade ao vetor posição corrente - Vide *Equação* (2.19).

$$X_{ij}^{k+1} = X_{ij}^k + V_{ij}^{k+1} \quad (2.19)$$

A *Figura 2.1* mostra uma interpretação geométrica no plano bi-dimensional do movimento de uma partícula em especial, onde pode-se observar os três vetores atuantes: O vetor na cor vermelha representa o fator social; o de cor azul, o fator individual; o de cor preta, o fator inercial e o vetor de cor verde, a possível nova posição da partícula após o deslocamento.

O pseudocódigo da Otimização por Nuvem de Partículas, na sua versão PSO GBest, é mostrado na *Algoritmo 1* e seu respectivo fluxograma é visto na *Figura 2.2*.

2.4 Busca Tabu (*Tabu Search* - TS)

A Busca Tabu foi proposta na sua forma atual por [Glover 1986] e tornou-se uma forma bem estabelecida de otimização em diversas áreas de conhecimento devido a sua

Algoritmo 1: Pseudocódigo do algoritmo PSO GBest

```

1  Gerar aleatoriamente  $q$  partículas iniciais  $X_{ij}$ ;
2  Inicializar  $w$ ;
3  Inicializar  $kmax$ ;
4   $V_{ij} \leftarrow 0$ ;
5   $c_1 \leftarrow rand(0 - 1)$ ;
6   $c_2 \leftarrow rand(0 - 1)$ ;
7  Determinar  $PBest_i$  e  $GBest$ ;
8  for  $k \leftarrow 0$  to  $kmax - 1$  do
9      for  $i \leftarrow 0$  to  $q - 1$  do
10         for  $j \leftarrow 0$  to  $n - 1$  do
11             Atualizar  $V_{ij}^k$  (Eq. 2.18);
12             Atualizar  $X_{ij}^k$  (Eq. 2.19);
13         end
14         Calcula aptidão da partícula de acordo com a função objetivo;
15         Atualiza  $PBest_i$ ;
16     end
17     Atualiza  $GBest$ ;
18 end

```

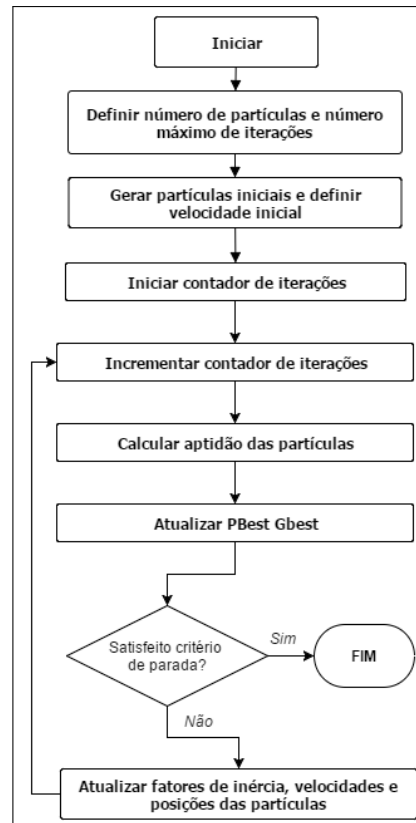


Figura 2.2: Fluxograma do algoritmo de Otimização por Nuvem de Partículas

eficiência na busca de soluções ótimas.

A meta-heurística Busca Tabu é um procedimento adaptativo auxiliar que utiliza uma estrutura de memória para armazenar as soluções geradas, ou características destas, e guiar um algoritmo de busca local na exploração contínua do espaço de busca. A partir de uma solução inicial, move-se de uma solução para uma outra solução, até que se satisfaça um determinado critério de parada.

O espaço de busca é o conjunto de todas as soluções possíveis. Entretanto, a exploração de toda a vizinhança normalmente tem um custo de processamento elevado, o que torna essencial utilizar uma lista de candidatos, a qual restringe os movimentos apenas a soluções promissoras, ou simplesmente reduzindo a vizinhança.

A TS, segundo [Souza 2010], inicia com uma solução válida e, a cada iteração, move-se para uma solução que é sua melhor vizinha, ou seja, explora o espaço de soluções S indo de uma solução $s \in S$ a outra s' que esteja em sua vizinhança $V \subseteq N(s)$. A solução $s' \in N(s)$ é dita vizinha de s se ela for obtida pela aplicação de um movimento m a s , operação esta representada por $s' \leftarrow s \oplus m$. As soluções encontradas são guardadas em uma memória de curto prazo (lista tabu) durante um certo tempo ou número de iterações. A TS não aceita movimentos que a levem a soluções já visitadas, evitando, assim, retornar a um ótimo local visitado previamente, de forma a fugir da otimalidade local e atingir um resultado ótimo ou próximo ao ótimo global. A *Figura 2.3* mostra uma representação gráfica de ótimos de máximos locais e ótimo de máximo global.

Componentes básicos da Busca tabu:

- Critério de escolha da próxima solução vizinha;
- Seleção dos atributos do movimento;
- Memória de curto prazo;

Também chamada de Lista tabu, armazena as regras de proibição e tem por finalidade evitar a geração de uma mesma sequência de soluções já visitadas. Sempre que uma solução é gerada ela é armazenada na lista tabu e fica proibida de tornar-se a solução corrente por um certo número de iterações. A lista tabu clássica tem tamanho fixo. Quando a lista está cheia e uma nova solução entra,

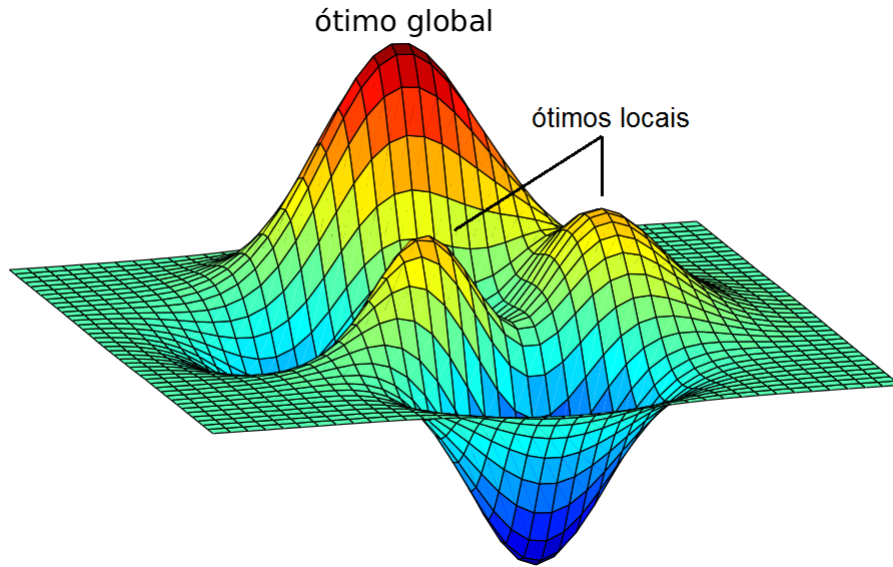


Figura 2.3: Representação gráfica de máximos ótimos locais e ótimo global

Algoritmo 2: Pseudocódigo de um algoritmo básico de Busca Tabu [Souza 2010]

input : $f(\cdot), N(\cdot), |T|, |V|, s$
output: s^*

```

1   $s' \leftarrow s$            {Melhor solução encontrada até então};
2   $Iter \leftarrow 0$        {Contador do número de iterações};
3   $MelhorIter \leftarrow 0$  {Iteração mais recente que forneceu  $s^*$ };
4   $T \leftarrow \emptyset$    {Lista Tabu};
5  while Critério de parada não satisfeito do
6     $Iter \leftarrow Iter + 1$ ;
7    Seja  $s' \leftarrow s \oplus m$  o melhor elemento de  $V \subseteq N(s)$  tal que o movimento  $m$ 
      não seja tabu ( $m \notin T$ ) ou  $s'$  atenda a um critério de aspiração;
8    Atualize a lista tabu;
9     $s \leftarrow s'$ ;
10   if  $f(s) < f(s')$  then
11      $s^* \leftarrow s$ ;
12      $MelhorIter \leftarrow Iter$ ;
13   end
14 end
15 Return  $s^*$ 

```

a mais antiga sai desta lista, tendo em vista que soluções geradas a mais tempo provavelmente estarão distantes da região de espaço sob análise [Souza 2010].

- Numero de iterações que um atributo selecionado ficará proibido (tamanho da lista);

Número de ciclos que uma solução está impedida de tornar-se a solução corrente, evitando a ciclagem. Em uma lista de tamanho T a solução fica impedida por T iterações e após poderá fazer parte novamente da solução corrente. O tamanho da lista tabu é um parâmetro critico da TS: caso seja muito pequeno pode não evitar a ciclagem e se for muito grande pode armazenar desnecessariamente soluções que não estejam ligadas a busca recente, aumentando o custo computacional para verificar a lista.

- Critério de aspiração.

Para evitar o custo computacional, é adotada uma estratégia onde apenas as características da solução são armazenadas. Ao invés de armazenar toda a solução, apenas o movimento realizado (elemento que entre e que sai da solução) é armazenado na lista tabu. Entretanto, esta estratégia pode proibir que uma solução inédita seja gerada. O critério de aspiração é um mecanismo que retira um movimento da lista tabu sobre certas circunstâncias, como por exemplo, se um movimento leva a uma solução global melhor do que todas que já foram encontradas, pois significa que ela ainda não foi gerada.

O pseudocódigo do *Algoritmo 2* apresenta as ideias básicas da busca tabu.

Capítulo 3

Trabalhos Relacionados

Neste capítulo faz-se um levantamento de alguns trabalhos relacionados na literatura, que incluem heurísticas e meta-heurísticas, para a resolução do Problema da Diversidade Máxima.

[Kuo, Glover e Dhir 1993] mostraram que o Problema da Diversidade Máxima é um problema NP-Difícil e apresentaram formulações matemáticas, as quais vimos na *Seção 2.1.3*, para a resolução do problema.

[Ghosh 1996] foi o primeiro a propor um algoritmo GRASP para solucionar o Problema da Diversidade Máxima e implementou um algoritmo exato para comparar eficiência da heurística proposta. Também criou instâncias de testes com população de até 40 elementos onde deveriam ser encontradas soluções com subconjuntos de tamanho 20% e 40% dos elementos totais. Os resultados computacionais mostraram que o GRASP proposto alcança uma solução ótima para instâncias de tamanho $n = 30$ e $m = 6$ e de $n = 25$ e $m = 5$ em uma máquina SPARCstation 2 e sistema operacional SunOS 4.1.1, com uma média de 413, 85 e 32, 17 segundos, respectivamente e uma solução muito próxima da ótima para às instâncias restantes.

[Agrafiotis 1997] apresentou uma nova família de algoritmos de seleção que combinam um motor de busca estocástica com uma função objetivo definida pelo usuário que codifica qualquer critério de seleção desejável e aplicou o método para o problema de maximizar a diversidade molecular, onde selecionava subconjuntos de componentes apropriados para a síntese química e avaliação biológica. O algoritmo baseado em *Simulated Annealing* foi testado em base de dados formadas a partir de métodos esta-

tísticos e os resultados demonstraram que o método é capaz de visitar áreas remotas e pouco povoadas do espaço de amostragem e pode escapar facilmente de áreas de alta densidade local, que é um defeito típico de algumas metodologias de clustering.

[Hassin, Rubinstein e Tamir 1997] apresentaram algoritmos de aproximação para o problema que considera um grafo simétrico $G = (V, E)$ completo, onde a cada aresta é associado um peso e $|V| = n$. Sendo $p = \{2, \dots, n\}$ e $k = \{1, \dots, n/p\}$ devem-se obter k conjuntos disjuntos, com p vértices cada um de tal forma que a soma dos pesos das arestas de cada subconjunto seja máxima. Uma variação do problema, onde $k = 1$, é o MDP.

[Glover, Kuo. e Dhir 1998] *apud* [Duarte, Silva e Costa 2008], propuseram duas heurísticas construtivas e outras duas destrutivas onde, a partir de uma solução heurística com e elementos ocorre uma eliminação progressiva até que se tenha uma solução viável com m elementos. A eficiência dessas heurísticas foi testada através de comparação com os resultados de um método exato e para isso foram criadas diversas instâncias de populações com até 30 elementos. Segundo os autores, as heurísticas propostas obtiveram em média resultados próximos aos do algoritmo exato (a no máximo 2% do ótimo), com um tempo de computação muito menor.

[Silva, Ochi e Martins 2005] propuseram três heurísticas de construção e duas de busca local para analisar seus desempenhos na meta-heurística GRASP. A heurística de construção HA_KDM e a heurística de construção HA_KDM_v2 constroem a solução inicial, elemento por elemento, escolhendo aleatoriamente um elemento a partir de uma lista restrita de candidatos. A diferença entre elas está na maneira como os elementos da lista de candidatos são alterados. A terceira heurística de construção apresentada, HA_IMD , incorpora o conceito clássico de heurística iterativa de inserção mais distante, onde foi modificada para torná-la adaptativa, gulosa e randômica. O método de busca local, (BLP) , utiliza duas estruturas de vizinhança. Na primeira vizinhança (BLG) , a cada passo, um elemento que faz parte da solução é trocado por outro de fora desta. Na segunda, dois elementos por vez são permutados. Para avaliar o desempenho das heurísticas foram criadas instâncias com conjuntos N de tamanho 10, 20, 30, 40 e 50 elementos onde deveriam ser encontradas soluções com subconjuntos de tamanho 10%, 20%, 30% e 40% do tamanho de N . Também foram utilizadas

instâncias encontradas na literatura de população 100,150, 200 e 250 elementos. Os resultados computacionais mostraram que os algoritmos propostos sempre alcançam uma solução ótima quando esta é conhecida e, para instâncias maiores, apresentam um desempenho médio superior quando comparados com as melhores abordagens GRASP da literatura.

[Duarte e Martí 2007] propuseram quatro métodos construtivos com base em Busca Tabu que incorpora estruturas de memória, e quatro construções sem memória com base na metodologia GRASP. Os métodos construtivos proposto por eles podem ser utilizadas como passo inicial para se obter um conjunto de boas e diversificadas soluções. Os oito processos podem ser acoplados com uma fase de melhoria. Propuseram, também, duas variantes: uma busca local eficiente (em comparação com um desenvolvimento anterior) e uma fase de melhoria utilizando Busca Tabu com memória de curto prazo. Em geral, as experiências com 120 exemplos mostraram a eficácia da utilização de memória tanto em construção como em melhoria para resolver este problema.

[Palubeckis 2007] desenvolveu um algoritmo *Iterated Tabu Search* (ITS) para resolver o MDP. Também apresentou um algoritmo de subida mais íngreme, o que é bem adequado em configurações de aplicativos em que são necessárias soluções de qualidade satisfatória serem fornecidas muito rapidamente. Resultados computacionais para instâncias de problemas envolvendo até 5000 elementos mostram que o algoritmo ITS é uma alternativa muito atraente para as abordagens existentes.

[Aringhieri, Cordone e Melzani 2008] propuseram uma abordagem que consiste em aprimorar a fase de Busca Local, através da adoção de um algoritmo de Busca Tabu utilizando memória de curto e longo prazo e mantendo um procedimento de inicialização muito simples. Utilizam os conceitos de lista tabu dinâmica e memória secundária, onde armazenam as segundas melhores soluções encontradas, mas não utilizadas em cada iteração da busca tabu, e as utilizam para reiniciar a busca em casos específicos. Eles fizeram uma comparação entre o algoritmos proposto e as heurísticas GRASP utilizando as instâncias propostas por [Andrade et al. 2003] e [Silva, Ochi e Martins 2004]. Os resultados computacionais mostraram que a Busca Tabu alcança melhores resultados e tempos computacionais menores em relação aos relatados para o GRASP.

[Duarte, Silva e Costa 2008] propuseram algoritmos heurísticos baseados nos concei-

tos da meta-heurística Iterated Local Search (ILS), que consiste em um procedimento iterativo no qual buscas locais são aplicadas às novas soluções de partida, obtidas através de perturbações feitas em soluções de ótimos locais. Os procedimentos implementados no trabalho incluem uma heurística de construção aleatória e gulosa, três algoritmos de busca local baseados em conceitos clássicos e um esquema de perturbação de solução que combinados produzem uma abordagem baseada na meta-heurística ILS. Os resultados computacionais mostram que os algoritmos propostos apresentam um bom desempenho quando comparados com as melhores heurísticas da literatura.

[Brimberg et al. 2009] propuseram vários procedimentos de VNS originalmente dedicados ao *K-subgraph Problem*, que é uma generalização do MDP em que as distâncias são substituídas por pesos. Portanto, qualquer algoritmo projetado para o k-subgrafo é capaz de resolver o MDP. Os autores apresentaram diferentes versões VNS, incluindo *skewed VNS*, *Basic VNS* uma combinação de uma heurística construtiva seguida por VNS. O melhor método global de acordo com a sua experimentação é o VNS básico, *B_VNS*, que consiste em três elementos principais. O primeiro, permite que o algoritmo atualize eficientemente o valor da função objetivo; o segundo, gera soluções na vizinhança da solução atual, realizando *swaps* de vértices aleatórios e o terceiro é um procedimento de busca local baseado em trocas.

[Gallego et al. 2009] desenvolveram procedimentos híbridos dentro da estrutura de busca de dispersão (*Scatter Search - SS*) com o objetivo de descobrir os projetos mais eficazes para resolver o problema. A pesquisa revelou a eficácia da adição de estruturas de memória simples (baseadas na recência e na frequência) aos principais mecanismos de busca de dispersão. Os experimentos extensivos e os testes estatísticos relacionados mostraram que a variante de busca de dispersão mais eficaz superava os métodos estado-da-arte.

[Martí, Gallego e Duarte 2010] propõe um algoritmo *Branch and Bound* para resolver o Problema da Diversidade Máxima de forma exata. Testes foram realizados utilizando um tempo máximo de processamento de 3600 segundos sobre instâncias de [Glover, Kuo. e Dhir 1998], onde conseguem obter soluções ótimas para instâncias de tamanho $n = 100$ e $m = 10$ e sobre as instâncias de [Silva, Ochi e Martins 2004], onde também conseguem soluções ótimas para soluções tamanho $n = 100$ e $m = 10$. Os

resultados empíricos com uma coleção de instâncias anteriormente relatados indicam que o algoritmo proposto é capaz de resolver todas as instâncias de média dimensão (com 50 elementos), bem como alguns casos de grande porte (com 100 elementos). O método foi comparado com a melhor formulação linear inteira anterior resolvida com o software Cplex. Segundo os autores, a comparação favorece o procedimento proposto.

[Lozano, Molina e García-Martínez 2011] apresentaram uma meta-heurística que gera uma sequência de soluções por iteração sobre uma heurística de construção gulosa, usando fases de construção e desconstrução. Segundo o autor, experiências computacionais revelaram que o algoritmo proposto é extremamente eficaz em comparação com as melhores meta-heurísticas para o problema em questão.

[Takane 2011] apresentou uma nova formulação para o problema baseado na Técnica de Reformulação de Linearização. Devido às características da formulação proposta e da dificuldade de resolução, o método de Decomposição de Benders Revisado é aplicado ao problema, assim como uma técnica de pré-processamento de modo a acelerar a sua convergência. Testes são realizados para avaliar o desempenho do método aplicado ao problema e em seguida, uma análise é feita comparando-o com outro algoritmo descrito na literatura. Os resultados computacionais mostram que o método proposto demonstra ser competitivo frente aos métodos exatos descritos na literatura.

[Wu e Hao 2013] apresentaram um algoritmo híbrido, nomeado MAMDP, para o MDP. O algoritmo utiliza um operador de crossover dedicado para gerar novas soluções e um procedimento de busca tabu com vizinhança restrita para otimização local. MAMDP aplica-se também a estratégia de substituição baseado na distância e qualidade para manter a diversidade da população. Avaliações abrangentes sobre um grande conjunto de 120 casos de benchmark revelam que a abordagem proposta concorre muito favoravelmente com os atuais métodos de estado-da-arte para MDP. Os principais componentes do MAMDP são analisados para identificar a sua influência no desempenho do algoritmo.

[Bonotto e Cabral 2014] apresentaram um algoritmo baseado na abordagem de Otimização por Nuvem de Partículas (PSO), onde a cada iteração a melhor solução servia de ponto de partida para uma meta-heurística de Busca Local. A utilização da PSO para resolver o MDP mostrou-se viável e os resultados computacionais apresentaram

resultados satisfatórios.

[Wang et al. 2014] apresentaram um algoritmo Memético altamente eficaz para o problema da diversidade máxima baseado na abordagem Busca Tabu. O componente de Busca Tabu utiliza uma estratégia de lista de filtro de candidatos sucessivos e o componente de combinação emprega um operador de combinação com base na identificação de variáveis fortemente determinadas e consistentes. A Busca Tabu utiliza uma técnica semelhante a [Aringhieri, Cordone e Melzani 2008] para determinar rapidamente o ganho do movimento de troca (a mudança do objetivo passando da solução corrente para a solução vizinha). As experiências computacionais em três conjuntos de 40 instâncias de *benchmark* populares indicam que a Busca Tabu / algoritmo Memético (TS / MA) pode facilmente obter os melhores resultados conhecidos para todas as instâncias testadas, bem como a melhoria dos resultados para seis instâncias (onde nenhum algoritmo anterior alcançou). Análise de comparações com algoritmos estado-da-arte demonstra estatisticamente que o algoritmo TS / MA compete muito favoravelmente com os melhores algoritmos em desempenho. Os elementos-chave e propriedades de TS / MA também são analisados para divulgar os benefícios da integração de Busca Tabu (usando uma estratégia de lista de filtro de candidatos sucessivos) e combinação de solução (com base em variáveis críticas).

A *Tabela 3.1* apresenta um resumo das principais abordagens encontradas na literatura para resolver o Problema da Diversidade Máxima.

Trabalho / Ano	Abordagem
Kuo, Glover e Dhir 1993	FM
Ghosh 1996	GRASP
Agrafiotis 1997	SA
Silva, Ochi e Martins 2005	GRASP
Duarte e Martí 2007	TS + GRASP
Palubeckis 2007	ITS
Aringhieri, Cordone e Melzani 2008	TS
Duarte, Silva e Costa 2008	ILS
Brimberg et al. 2009	VNS
Gallego et al. 2009	SS

Martí, Gallego e Duarte 2010	ME
Lozano, Molina e García-Martínez 2011	IG
Takane 2011	FM + ME
Wu e Hao 2013	MA + TS
Bonotto e Cabral 2014	PSO + LS
Wang et al. 2014	MA + TS

Tabela 3.1: Principais abordagens da literatura para o MDP

Capítulo 4

Método Proposto

O algoritmo proposto utiliza uma versão da meta-heurística Otimização por Nuvem de Partículas combinada com um algoritmo de Busca Tabu para refinar as soluções. As partículas iniciais a serem utilizadas pelo PSO serão geradas por duas heurísticas de construção, sendo uma aleatória e outra gulosa. O método foi denominado **PSO_TS**.

4.1 Algoritmo PSO_TS

O algoritmo PSO_TS inicia com a geração de q partículas de tamanho m , onde $Nr_PG\%$ delas - sendo no mínimo uma partícula - é (são) criada(s) através de um algoritmo guloso simples e, nas partículas restantes, os m elementos são escolhidos aleatoriamente obedecendo aos limites da instância. A quantidade q de partículas é definida dinamicamente no início da execução do algoritmo através da *Equação* (4.1), e varia de acordo com o número de elementos m que compõem a solução, tendo como um limite mínimo de 25 partículas. Durante a criação das partículas, $GBest$ e $PBest$ vão sendo atualizados.

$$q = \max(25, q \times m) \quad (4.1)$$

Após a criação das partículas, elas são ordenadas de acordo com sua diversidade máxima e as $Nr_Ch_TS_In$ melhores partículas servirão como soluções de partida para um algoritmo de Busca Tabu. Caso haja melhora na diversidade máxima, a

solução retornada da TS substituiu a partícula e $GBest$ e $PBest$ são atualizados.

A heurística gulosa é descrita na *Seção 4.2.2* e o procedimento TS é descrito com detalhes na *Seção 4.3*. Os valores das constantes q e Nr_PG são mostrados na *Tabela 5.2* e os valores de $Nr_Ch_TS_In$ são mostrados na *Tabela 5.3* da *Seção 5.2*.

A cada iteração k , a partícula desloca-se pelo espaço multidimensional conforme a *Equação (4.2)*:

$$X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1} \quad (4.2)$$

Onde:

X_{id}^k é a posição do elemento i na dimensão d da partícula no espaço multidimensional na k -ésima iteração, e;

V_{id}^k é a velocidade de deslocamento do elemento i na dimensão d da partícula na k -ésima iteração.

A velocidade de deslocamento da partícula é calculada pela *Equação (4.3)*:

$$V_{id}^{k+1} = W \times V_{id}^k + c_1 \times Cr \times (P_{id}^k - X_{id}^k) + c_2 \times (1 - Cr) \times (P_{gd}^k - X_{id}^k) \quad (4.3)$$

Onde:

W é uma variável que define o peso inercial que será aplicado à velocidade da partícula e o valor é escolhido através da equação $W = 0.9 - ((0.9 - 0.1)/k) \times$. Esta equação, baseada em [Coelho Filho 2013], faz com que nas k iterações iniciais o valor de W - entre $[0.1, \dots, 0.9]$ - seja próximo do máximo e nas iterações finais fique próximo do mínimo. Isto faz com que a partícula se desloque mais rapidamente no início do processamento e consiga "escapar" de ótimos locais.

Cr é escolhido através da equação $Cr = y \times Cr \times (1 - Cr)$, descrita por [Chuang, Hsiao e Yang 2011], e gera sequências de números caóticos em substituição a r_1 e r_2 da equação original do PSO. Inicialmente Cr é escolhido aleatoriamente entre $[0, \dots, 1]$. Foi usado $y = 4$, o que faz com que a equação tenha um comportamento caótico, mas gere a maioria dos números nas proximidades de 0 ou 1;

c_1 é uma constante que define a velocidade em direção à $PBest$ e expressa o fator

cognitivo da partícula;

c_2 é uma constante que define a velocidade em direção à $GBest$ e expressa o fator social da partícula. Os valores das constantes c_1 e c_2 são mostrados na *Tabela 5.2* da *Seção 5.2*;

X_{id}^k é a posição atual da partícula na k -ésima iteração;

P_{id}^k ($PBest$) a melhor posição já visitada pela partícula até a k -ésima iteração, e;

P_{gd}^k ($GBest$) a melhor posição já visitada por qualquer uma das partículas até a k -ésima iteração.

Em algumas situações, a nova posição do elemento i da partícula ocorre fora do espaço vetorial. Neste caso, o elemento não é deslocado nesta iteração.

Também acontecem casos em que a nova posição do elemento coincide com a posição de outro elemento da mesma partícula. Sendo assim, este elemento também não é deslocado na iteração corrente.

A cada iteração as partículas se movem para a nova posição e a diversidade máxima é calculada. Caso haja melhora na diversidade localmente (maior diversidade máxima já atingida pela partícula), $PBest$ é atualizado. Igualmente, caso haja melhora em $GBest$, ele é atualizado.

Ao final de cada iteração, as partículas são ordenadas em ordem decrescente de acordo com a sua diversidade e um algoritmo de Busca Tabu (TS) é iniciada Nr_Ch_TS vezes. A TS tem como soluções de partida as Nr_Ch_TS partículas com maior diversidade máxima na iteração corrente.

Ao término da Busca Tabu, as partículas que serviram de solução inicial para a TS são substituídas pelas partículas retornadas da Busca Tabu e $GBest$ e $PBest$ são atualizados.

O algoritmo termina quando atingir um número $Kmax$ de iterações ou atingir o tempo t máximo de processamento.

O último $GBest$ encontrado é a solução do problema.

O número máximo de iterações do PSO é definido dinamicamente no início da execução do algoritmo através da *Equação (4.4)* e depende do número de elementos m

da instância e do valor parâmetro $Kmax$.

$$Kmax = \text{máximo}(100, Kmax \times m) \quad (4.4)$$

Os valores das constantes $Kmax$, e Nr_Ch_TS são mostrados na *Tabela 5.2* e na *Tabela 5.3* da *Seção 5.2*;

Algoritmo 3: Pesudocódigo do Algoritmo PSO_TS

input : Matriz $M_{n \times n}$ (d_{ij}) com cardinalidade $m \leq n$
output: Melhor solução encontrada Gb^*

```

1  $P \leftarrow \{x^1, \dots, x^{\max(1, \text{int}(m \times Nr\_PG/100))}\} \leftarrow \text{Gera\_particulas\_gulosas}() \quad /*$ 
    $\text{Seção 4.2.1} */;$ 
2  $P \leftarrow \{x^{\max(1, \text{int}(m \times Nr\_PG/100))+1}, \dots, x^q\} \leftarrow \text{Gera\_particulas\_aleatorias}();$ 
3  $Pb^* \leftarrow \{x^1, \dots, x^q\} \leftarrow P = \{x^1, \dots, x^q\};$ 
4  $Gb^* \leftarrow \text{argmax}\{f(x^i) | i = \{1, \dots, q\}\};$ 
5  $k \leftarrow 0;$ 
6 while  $k < Kmax$  do
7    $Nr\_Ch = Nr\_Ch\_TS;$ 
8   if  $k = 0$  then
9      $Nr\_Ch = Nr\_Ch\_TS\_In \times m/100;$ 
10  end
11   $\text{Ordena } P = \{x^1, \dots, x^q\} \text{ (Descrescente pela diversidade máxima)};$ 
12  for  $i \leftarrow 0$  to  $Nr\_Ch$  do
13     $P(s) = TS(P(s)) \quad /* \text{Seção 4.3} */;$ 
14     $\text{Atualiza } Gb^* \text{ e } Pb^*(s);$ 
15    if  $\text{Tempo\_processamento} > t$  then
16      return  $Gb^*;$ 
17    end
18  end
19  for  $i \leftarrow 0$  to  $q - 1$  do
20     $\text{Move } P[i] \quad /* \text{Eq. (4.2)} */;$ 
21     $\text{Atualiza } Gb^* \text{ e } Pb^*(i);$ 
22  end
23 end
24 return  $Gb^*.$ 

```

O pseudocódigo do algoritmo PSO_TS é mostrado no *Algoritmo 3*, onde as linhas 1 e 2 criam as partículas iniciais; as linhas 3 e 4, definem o PBest e o GBest, respectivamente; O laço da linha 6 à 23 refere-se as iterações do PSO; as linhas 7 à 10 definem o número de vezes que a Busca Tabu será executada na iteração corrente; a linha 11 ordena as partículas em ordem decrescente em função do valor da diversidade

máxima; o laço das linhas 12 à 18 evocam a TS; às linhas 15 à 17 verificam se o critério de parada foi atingido e o laço das linhas 19 à 22 movem as partículas no espaço de soluções.

4.2 Algoritmos de Construção

4.2.1 Heurística de Construção Gulosa

A heurística gulosa cria uma partícula (solução válida) conforme os passos a seguir:

- (1) escolhe aleatoriamente um elemento da instância N para fazer parte da solução U ;
- (2) cria um vetor S contendo os elementos que não fazem ainda parte de U ;
- (3) armazena em S o valor que cada elemento N/U contribuiria ao fazer parte da solução parcial, conforme *Equação (4.5)*;

$$S_i = \sum_{j \in U} d_{ij} \quad (4.5)$$

- (4) adiciona à solução parcial U o primeiro elemento encontrado com a maior contribuição no vetor S e exclui este elemento do vetor;

(5) O vetor S é atualizado conforme a *Equação (4.5)*, assim como a diversidade máxima da solução parcial, e;

- (6) retorna ao passo (4) até que uma solução válida com m elementos seja gerada.

A escolha do primeiro elemento sendo aleatória torna o algoritmo não determinístico e permite a geração de mais de uma partícula com elementos diferentes. Durante a criação das partículas é calculada a diversidade máxima de cada uma delas a fim de que seja escolhido o $GBest$ (partícula com a maior diversidade máxima entre todas as partículas). A própria partícula, na sua criação, é definida como $PBest$ (melhor posição da partícula encontrada até o momento).

O pseudocódigo da Heurística Gulosa é mostrado no *Algoritmo 4*, onde o laço das linhas 1 à 14 definem o número de partículas a serem criadas; a linha 2 inicializa o vetor que armazena a diversidade das partículas; a linha 3 escolhe um elemento aleatoriamente; a linha 4 cria o vetor S , que armazena a contribuição que cada elemento contribuiria ao fazer parte da solução; o laço das linhas 5 à 11 escolhe os m elementos,

selecionados pela linha 6 (escolhe o elemento com melhor contribuição na iteração corrente); o laço das linhas 7 à 9 atualizam a diversidade máxima da partícula; a linha 10, atualiza o vetor S após a escolha do elemento e as linhas 12 e 13 atualizam o GBest e o PBest.

Algoritmo 4: Pseudocódigo da Heurística de Construção Gulosa

```

1 for  $i \leftarrow 0$  to  $\max(1, \text{int}(m * 0.4)) - 1$  do
2    $\text{Diversidade}[i] \leftarrow 0$ ;
3    $P[i][0] \leftarrow \text{rand}() \% n$ ;
4    $S \leftarrow$  valor que cada elemento ( $N/M$ ) contribuiria ao fazer parte da
      solução parcial /* Equação (4.5) */;
5   for  $j \leftarrow 1$  to  $m - 1$  do
6      $P[i][j] \leftarrow$  elemento de  $S$  que apresenta maior contribuição;
7     for  $x \leftarrow 0$  to  $j - 1$  do
8        $\text{Diversidade}[i] \leftarrow \text{Diversidade}[i] + \text{diversidade}[P[i][j], x]$ ;
9     end
10    Atualiza  $S$ ;
11  end
12  Atualiza  $Gb^*$ ;
13  Atualiza  $Pb^*(i)$ ;
14 end
15 return  $P$ .

```

4.2.2 Heurística de Construção Aleatória

A Heurística de Construção Aleatória escolhe aleatoriamente um elemento, que não faça parte da solução, após o outro até que seja gerada uma solução válida com m elementos.

O pseudocódigo da Heurística Aleatória é mostrado no *Algoritmo 5*, onde o laço das linhas 1 à 11 definem o número de partículas a serem criadas; a linha 2 inicializa o vetor que armazena a diversidade das partículas; o laço das linhas 3 à 8 escolhe os m elementos, selecionados pela linha 4 (escolhe aleatoriamente um elemento que ainda não faz parte da solução); o laço das linhas 5 à 7 atualizam a diversidade máxima da partícula e as linhas 9 e 10 atualizam o GBest e o PBest. .

Algoritmo 5: Pseudocódigo da Heurística de Construção Aleatória

```

1 for  $i \leftarrow \max(1, \text{int}(m * 0.4))$  to  $q - 1$  do
2    $\text{Diversidade}[i] \leftarrow 0$ ;
3   for  $j \leftarrow 0$  to  $m - 1$  do
4      $P[i][j] \leftarrow \text{rand}() \% (n/m)$ ;
5     for  $x \leftarrow 0$  to  $j - 1$  do
6        $\text{Diversidade}[i] \leftarrow \text{Diversidade}[i] + \text{diversidade}[P[i][j], x]$ ;
7     end
8   end
9   Atualiza  $Gb^*$ ;
10  Atualiza  $Pb^*(i)$ ;
11 end
12 return  $P$ .

```

4.3 Busca Tabu

A meta-heurística Busca Tabu é um procedimento adaptativo auxiliar que utiliza uma estrutura de memória para armazenar as soluções geradas ou características destas e guiar um algoritmo de busca local na exploração contínua do espaço de busca. A partir de uma solução inicial, move-se de uma solução para uma outra solução vizinha, até que se satisfaça um determinado critério de parada.

A Busca Tabu empregada foi descrita por [Wang et al. 2014] e se mostrou eficiente na melhoria das soluções. Consiste em aplicar um operador de troca restrita para trocar uma variável que tem valor 1 (elemento que faz parte da solução) para outra com valor 0. Formalmente, dada uma solução $x = \{x_1, \dots, x_n\}$ onde U e Z representam respectivamente as variáveis com valor 1 e 0 em x . Os vizinhos $N(x)$ são todas as soluções obtidas pela substituição de duas variáveis, onde $x_i \in U$ e $x_j \in Z$. A TS utiliza um mecanismo que determina rapidamente o ganho do movimento de troca, similar ao utilizado por [Aringhieri, Cordone e Melzani 2008].

Inicialmente, é empregado um vetor Δ para armazenar a variação da função objetivo decorrente do movimento de x_i do subconjunto atual de U/Z para o outro subconjunto Z/U . O vetor é inicializado conforme a *Equação* (4.6) que segue:

$$\Delta_i = \begin{cases} \sum_{j \in U} -d_{ij}(x_i \in U) \\ \sum_{j \in U} d_{ij}(x_i \in Z) \end{cases} \quad (4.6)$$

O ganho do movimento entre duas variáveis $x_i \in U$ e $x_j \in Z$ pode ser calculado usando a seguinte *Fórmula* (4.7):

$$\delta_{ij} = \Delta_i + \Delta_j - d_{ij} \quad (4.7)$$

Após realizado o movimento de troca entre as variáveis x_i e x_j o vetor Δ deve ser atualizado utilizando a seguinte *Equação* (4.8):

$$\Delta_k = \begin{cases} -\Delta_i + d_{ij}(k = i) \\ -\Delta_j + d_{ij}(k = j) \\ \Delta_k + d_{ik} + d_{jk}(k \notin \{i, j\}, x_k \in U) \\ \Delta_k - d_{ik} + d_{jk}(k \notin \{i, j\}, x_k \in Z) \end{cases} \quad (4.8)$$

O tamanho da vizinhança de troca é igual à $m(n - m)$ e pode ser computacionalmente custoso para identificar o melhor movimento a cada iteração da TS. Para superar esse obstáculo, foi empregada a estratégia de filtro aplicada a lista de candidatos sucessivo de [Glover e Laguna 1997] que restringe a análise para os movimentos que produzem resultados de alta qualidade para cada operação específica.

Para o movimento de troca, é necessário mover a variável x_i de U para Z e mover a variável x_j de Z para U . A diferença na função objetivo resultante de cada operação pode ser facilmente obtida a partir do vetor Δ . Para cada operação são escolhidas as principais *cls* variáveis (*cls* é um parâmetro que define tamanho da lista de candidatos) e são gravadas em ordem decrescente em relação aos valores de Δ para construir a lista de candidatos *UCL* e *ZCL*. Finalmente, os movimentos de troca são restritos as variáveis de *UCL* e *ZCL*. O tamanho da lista foi definido como $cls = \min(\sqrt{m}, \sqrt{n - m})$.

O tamanho da *cls* impacta diretamente no procedimento da TS. Um valor muito grande pode incluir alguns movimentos pouco atraentes na exploração da vizinhança,

enquanto um valor muito baixo poderia excluir alguns movimentos atraentes.

Algoritmo 6: Pseudocódigo da Busca Tabu (Baseado em [Wang et al. 2014])

```

input : Solução PSO
output: Solução melhorada

1  $Iter \leftarrow 0$ ;
2  $SemMelhora \leftarrow 0$ ;
3  $T[x_i, \dots, x_n] \leftarrow 0$  (Lista Tabu);
4 Criar e inicializar o vetor  $\Delta$  de acordo com a Eq. (4.6);
5 while  $SemMelhora < \alpha$  do
6   Construir a lista de candidatos UCL e ZCL, gravadas em ordem
     decrecente em relação aos valores de  $\Delta$ ;
7   Executa o movimento de troca de  $x_i$  para  $x_j$  que proporcione a melhor
     troca de acordo com a equação  $\delta_{ij} = \Delta_i + \Delta_j - d_{ij}$  e desde que
      $T[x_i] \leq Iter$  e  $T[x_j] \leq Iter$ ;
8   Atualiza o vetor  $\Delta$  de acordo com a Eq. (4.8);
9   Atualiza  $T[x_i]$  e  $T[x_j]$ ;
10   $Iter++$ ;
11  if Melhorou Solução then
12     $SemMelhora \leftarrow 0$ ;
13  else
14     $SemMelhora++$ ;
15  end
16 end
17 return Solução melhorada.

```

Para garantir que soluções visitadas dentro de um determinado espaço de iterações não sejam revisitadas, a Busca Tabu incorpora uma memória de curto prazo, conhecida como Lista Tabu [Glover e Laguna 1997]. Cada vez que duas variáveis x_i e x_j são trocadas, dois inteiros aleatórios são gerados a partir de um intervalo de $tt = [a, b]$ para evitar qualquer movimento envolvendo x_i ou x_j ser selecionado por um determinado número de iterações. Os números inteiros que definem a gama de tt , $a = 15$ e $b = 25$, são os mesmos empregados por [Wang et al. 2014]. Quando x_i e x_j são trocados, vamos atribuir a soma de um número inteiro aleatório de tt e o número atual da iteração $Iter$ ao elemento $T[i]$ de T e a soma de outro número inteiro aleatório de tt e $Iter$ para $T[j]$. Posteriormente, para qualquer iteração $Iter$, uma variável x_k é proibida de participar de um movimento de troca se $T[k] > Iter$. O número de iterações α foi definido como $6m$. O Algoritmo 6 mostra o pseudocódigo da Busca Tabu, onde as linhas 1 à 3 inicializam as variáveis; a linha 4, cria um vetor onde são armazenadas a contribuição

de cada um dos elementos; o laço das linhas 5 à 16 serve de critério de parada da TS; a linha 6 cria a lista de elementos candidatos a troca; a linha 7 executa o movimento de troca; a linha 8 atualiza o vetor Δ , a linha 9 atualiza a lista tabu; as linhas 10 à 15, atualizam as variáveis de controle e a linha 17, retorna a melhor solução encontrada.

Capítulo 5

Testes e Resultados

Neste capítulo são apresentados os aspectos relativos ao desenvolvimento do trabalho, tais como o ambiente de desenvolvimento, as características das instâncias de dados de referência utilizadas para os testes, os resultados computacionais e os dados referentes à parametrização utilizada.

5.1 Ambiente de Desenvolvimento

5.1.1 Hardware

A execução dos testes finais e para a parametrização, onde escolheu-se os valores das constantes, foram realizados utilizando um computador com processador Intel (R) Core (TM) i7 950 @ 3.07GHz com 4 núcleos, 8 GB memória, HD 2 TB Gb. Apenas 1 núcleo do processador foi utilizado.

5.1.2 Software

O algoritmo foi implementado usando a linguagem de programação C++ e o compilador g++ version 4.7.2 (Debian 4.7.2-5) e foram compilados com a opção g++ -O2. Para os testes de parametrização foi utilizado o Sistema Operacional (SO) Debian GNU/Linux 8.3 (jessie) e os testes para os resultados finais o Sistema Operacional Linux Ubuntu versão 4.2.0-35.

5.1.3 Base de dados

Para demonstrar a eficiência do algoritmo foram realizados experimentos utilizando três conjuntos de instâncias: o primeiro conjunto é considerado de médio porte e os outros dois conjuntos são considerados de grande porte. As instâncias foram retiradas e estão disponíveis para download em <http://www.opticom.es/mdp>.

- Instâncias de médio porte

20 instâncias SOM-b, criadas por [Silva, Ochi e Martins 2004], e que contém uma população de tamanho $n = 100$, $n = 200$, $n = 300$, $n = 400$ e $n = 500$ e testadas soluções de tamanho $m = 10\%n$, $20\%n$, $30\%n$ e $40\%n$. A diversidade entre dois elementos quaisquer nas instâncias foi escolhida aleatoriamente e uniformemente distribuídas com inteiros entre $[0, \dots, 9]$.

- Instâncias de grande porte

- 20 instâncias MDG-a, introduzidas por [Duarte e Martí 2007], com uma população de $n = 2000$ e $m = 200$, onde a diversidade entre dois elementos quaisquer nas instâncias foi escolhida aleatoriamente e uniformemente distribuídas com inteiros entre $[0, \dots, 10]$.
- 20 instâncias MDG-c, similares as usadas por [Palubeckis 2007], com população $n = 3000$ e $m = 300$, $m = 400$, $m = 500$ e $m = 600$. A diversidade entre dois elementos quaisquer nas instâncias foi escolhida aleatoriamente e uniformemente distribuídas com inteiros entre $[0, \dots, 1000]$. A *Tabela 5.1* mostra o tamanho das soluções testadas para estas instâncias.

m	n	Instâncias MDG-c_1 à MDG-c_20				
3000	300	c_1	c_2	c_3	c_4	c_5
3000	400	c_6	c_7	c_8	c_9	c_10
3000	500	c_11	c_12	c_13	c_14	c_15
3000	600	c_16	c_17	c_18	c_19	c_20

Tabela 5.1: Características das instâncias MDG-c

5.2 Definição dos Parâmetros

Para a correta execução do algoritmo alguns parâmetros devem ser previamente definidos. As seguintes constantes foram testadas, sendo seus intervalos definidos observando trabalhos previamente realizados por [Bonotto e Cabral 2014]:

- q - Número de partículas iniciais, definido como: $q \in \{0.15, 0.20, 0.25, 0.50, 1.00\} \times m$, onde m é o número de elementos da solução;
- Nr_PG - Percentual de partículas geradas através do algoritmo guloso, onde $Nr_PG \in \{1\%, 10\%, 40\%, 70\%, 80\%, 90\%\}$;
- $Kmax$ - Número máximo de iterações do PSO, definido da seguinte maneira: $Kmax \in \{0.5, 1, 2, 4\} \times m$;
- $C1$ - Constante usada para balancear a contribuição do componente cognitivo, sendo $C1 \in \{0.8, 1.0, 1.2\}$;
- $C2$ - Constante usada para balancear a contribuição do componente social, sendo $C2 \in \{1.2, 1.4, 1.6, 1.8\}$;
- Nr_Ch_TS - Número de chamadas a Busca Tabu ao final do deslocamento de todas as partículas ao final de uma iteração do PSO, onde $Nr_Ch_TS \in \{4, 5, 6, 7, 8, 9\}$, e;
- $Nr_Ch_TS_In$ - Percentagem de partículas iniciais que serão submetidas à Busca tabu, sendo $Nr_Ch_TS_In \in \{0\%, 20\%, 40\%, 60\}$.

Instâncias	q	Nr_PG	Kmax	C1	C2
SOM-b	1	80 %	1	1.00	1.40
MDG-a	.20	80 %	1	1.00	1.40
MDG-c	.20	60 %	4	1.00	1.40

Tabela 5.2: Valores definidos para as constantes - Parte 1

Os valores das constantes foram escolhidos de maneira empírica, baseado em testes da meta-heurística sendo executada sobre as instâncias SOM-b, MDG-a e MDG-c e variando o valor de algumas constantes. A *Tabela 5.2* e a *Tabela 5.3* mostram os valores definidos para as constantes de acordo com as instâncias utilizadas.

Instâncias	Nr_Ch_TS	$Nr_Ch_TS_In$
SOM-b	7	0%
MDG-a	8	0%
MDG-c	6	20%

Tabela 5.3: Valores definidos para as constantes - Parte 2

5.3 Testes e Resultados

Para verificar a eficiência do algoritmo proposto foram comparados os melhores resultados obtidos para cada instância e também os resultados médios encontrados com os melhores resultados conhecidos na literatura. As melhores soluções conhecidas da literatura para as instâncias SOM-b foram retiradas de [Duarte, Silva e Costa 2008], para as instâncias MDG-a foram retiradas de [Wang et al. 2014] e para as instâncias MDG-c foram retiradas de [Martí et al. 2013].

Dada a característica aleatorizada das metaheurísticas, os resultados foram obtidos de uma média de 10 execuções do PSO_TS para as instâncias de médio porte e uma média de 30 execuções PSO_TS para as instâncias de grande porte, com exceção das instâncias MDG-c onde foi considerado a média de 10 execuções, dada a sua grande dimensionalidade.

5.3.1 Resultados das Instâncias de Médio Porte

Os testes realizados nas instâncias consideradas de tamanho médio consistiram em executar a meta-heurística PSO_TS, por um tempo máximo de processamento de 1 segundos, em cada uma das 20 instâncias SOM-b. Para a diversidade máxima foi considerada apenas a média dos resultados obtidos.

A *Tabela 5.4* mostra os resultados médios obtidos para as instâncias de médio porte SOM-b. Nesta tabela, a *coluna 1* indica a instância utilizada; a *coluna 2*, a dimensão n desta instância; a *coluna 3*, a quantidade m de elementos da partícula (solução); a *coluna 4*, a melhor solução (diversidade máxima) conhecida na literatura (BKS); a *coluna 5*, a solução média encontrada nos 10 testes; as *colunas 6 e 7 e 8*, respectivamente, o tempo mínimo, médio e máximo de processamento até encontrar a solução.

Conforme pode-se observar, nas instâncias de médio porte, em todas as instâncias e em 100% dos 10 testes executados, o algoritmo PSO_TS encontrou a melhor solução conhecida literatura, ou seja, um desvio padrão nulo. O tempo máximo para encontrar a melhor solução variou de 0 a 0.52 segundos, dependendo do valor de n e m da instância e o tempo médio para encontrar as soluções foi de 0,08 segundos.

Comparamos os resultados com algoritmos *estado-da-arte* que utilizaram as instancias SOM-b para os testes. Várias meta-heurística foram analisadas no trabalho de [Martí et al. 2013], que realizaram os testes por um tempo de processamento de 10s e 20s em um processador Intel Core2 Quad Q8300 2.5GHz. Não foi encontrado na literatura testes com tempo de processamento inferior a 10 segundos. O PSO_TS sempre encontrou resultados melhores ou iguais, mesmo sendo executado com um limite de tempo de apenas 1 segundo.

Instâncias	n	m	BKS	PSO_TS			
				Média	Tempo (segundos)		
					Mín	Méd	Máx
SOM-b_1	100	10	333	333	0.00	0.00	0.00
SOM-b_2	100	20	1195	1195	0.00	0.00	0.00
SOM-b_3	100	30	2457	2457	0.00	0.00	0.00
SOM-b_4	100	40	4142	4142	0.00	0.00	0.01
SOM-b_5	200	20	1247	1247	0.00	0.02	0.08
SOM-b_6	200	40	4450	4450	0.00	0.02	0.05
SOM-b_7	200	60	9437	9437	0.00	0.01	0.01
SOM-b_8	200	80	16225	16225	0.01	0.01	0.02
SOM-b_9	300	30	2694	2694	0.00	0.04	0.12
SOM-b_10	300	60	9689	9689	0.01	0.10	0.29
SOM-b_11	300	90	20743	20743	0.01	0.06	0.19
SOM-b_12	300	120	35881	35881	0.04	0.09	0.15
SOM-b_13	400	40	4658	4658	0.02	0.09	0.15
SOM-b_14	400	80	16956	16956	0.02	0.14	0.37
SOM-b_15	400	120	36317	36317	0.05	0.10	0.23
SOM-b_16	400	160	62487	62484	0.09	0.23	0.31
SOM-b_17	500	50	7141	7141	0.04	0.26	0.52
SOM-b_18	500	100	26258	26258	0.03	0.12	0.31
SOM-b_19	500	150	56572	56572	0.06	0.08	0.16
SOM-b_20	500	200	97344	97344	0.11	0.26	0.52
Média					0.02	0.08	0.17

Tabela 5.4: Resultados das instâncias SOM-b (Tempo < 1s)

5.3.2 Resultados Instâncias de Grandes Porte

Para a análise das instâncias MDG-a, consideradas instâncias de grandes porte, foi utilizado um tempo máximo de processamento de 16 segundos.

Na *Tabela 5.5* tem-se os resultados encontrados para estas instâncias, onde as colunas indicam, respectivamente, a instância usada; a melhor solução conhecida na literatura; a melhor solução encontrada nos testes; o GAP(1) ($BKS - \#Best$); o GAP(2) ($BKS - \#Avg.$) e o coeficiente de variação. Uma vez que os valores absolutos da solução são muito grandes, relatamos a diferença (GAP) entre os melhores e médios valores da solução para os melhores resultados conhecidos.

Instância	BKS	PSO_TS				CV
		Solução		GAP		
		#Best	#Avg.	#Best	#Avg.	
MDG-a_21	114271	114271	114237.6	0	33	0.00018
MDG-a_22	114327	114327	114298.8	0	28	0.00047
MDG-a_23	114195	114195	114157.4	0	38	0.00029
MDG-a_24	114093	114091	114042.7	2	50	0.00022
MDG-a_25	114196	114196	114108.1	0	88	0.00035
MDG-a_26	114265	114265	114232.0	0	33	0.00016
MDG-a_27	114361	114361	114351.9	0	9	0.00016
MDG-a_28	114327	114327	114280.0	0	47	0.00054
MDG-a_29	114199	114199	114169.6	0	29	0.00018
MDG-a_30	114229	114229	114193.6	0	35	0.00024
MDG-a_31	114214	114214	114160.4	0	54	0.00025
MDG-a_32	114214	114214	114151.8	0	62	0.00034
MDG-a_33	114233	114233	114189.9	0	43	0.00028
MDG-a_34	114216	114216	114181.9	0	34	0.00033
MDG-a_35	114240	114240	114229.1	0	11	0.00013
MDG-a_36	114335	114335	114313.3	0	22	0.00025
MDG-a_37	114255	114255	114218.5	0	37	0.00029
MDG-a_38	114408	114408	114398.6	0	9	0.00015
MDG-a_39	114201	114201	114183.7	0	17	0.00031
MDG-a_40	114349	114349	114297.5	0	52	0.00038
Média				0.1	36.6	0.00027

Tabela 5.5: Resultados das instâncias MDG-a (Tempo $\leq 16s$)

O coeficiente de variação foi obtido pela razão entre desvio padrão e a média das

soluções encontradas nas execuções em cada uma das instâncias - *Equação (5.1)*,

$$CV = \frac{\sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}}{\bar{x}} \quad (5.1)$$

onde:

CV = coeficiente de variação;

$\bar{x}$ = média dos termos;

n = número de termos.

Pode-se observar que, com exceção de uma instância (MDG-24), o algoritmo encontrou a melhor solução conhecida na literatura. Na instância MDG-24, o GAP foi de apenas 2 unidades. O GAP médio foi de 36.6 unidades. O coeficiente de variação médio foi de 0.00027, o que demonstra a robustez do algoritmo.

Os resultados obtidos nas instâncias MDG-a foram comparados com algoritmos *estado-da-arte*, que foram relatados em [Wang et al. 2014] e foram executados por um tempo de 17 segundos em um computador Xeon E5440 com 2.83 GHz CPU e 8 GB RAM ou máquinas equivalentes.

Para determinar o tempo de parada do nosso algoritmo, nós usamos o *CPU Benchmarks - Single Thread Performance*¹ que indicou que nosso computador tem um fator 1.036 mais rápido que aqueles usados nos testes por [Wang et al. 2014]. Então, nós determinamos o tempo de parada como 16 segundos para as instâncias MDG-a. A utilização do fator que indica quanto o computador utilizado neste trabalho é mais rápido que o utilizado nas técnicas competidoras visa tornar a comparação entre os tempos computacionais um pouco mais justa, apesar da diferença na arquitetura e no conjunto de instruções dos processadores.

O gráfico da *Figura 5.1* faz uma comparação entre o GAP médio ($BKS - \#Best$) das melhores soluções encontradas para cada instância. Observa-se que o algoritmo PSO_TS obteve melhores resultados que os algoritmos ITS [Palubeckis 2007], VNS [Brimberg et al. 2009], TIG [Lozano, Molina e García-Martínez 2011] e LTS_EDA [Wang et al. 2012]. Na comparação com o algoritmo TS-MA [Wang et al. 2014], obtivemos resultados ligeiramente inferiores quanto à qualidade das soluções.

¹<http://www.cpubenchmark.net/singleThread.html>

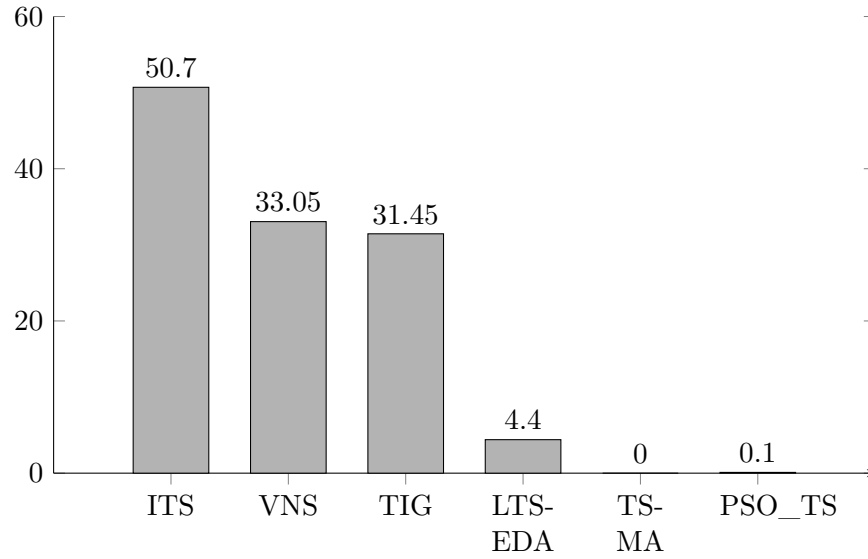


Figura 5.1: Comparação entre a melhor solução do PSO_TS e outros algoritmos *estado-da-arte* (Instâncias MDG-a)

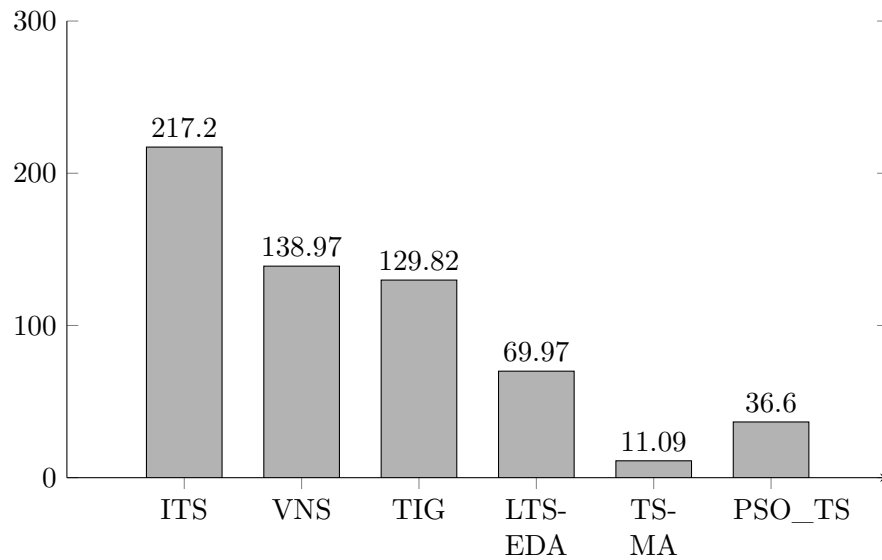


Figura 5.2: Comparação entre a solução média do PSO_TS e outros algoritmos *estado-da-arte* (Instâncias MDG-a)

O gráfico da *Figura 5.2* faz uma comparação entre o GAP médio ($BKS - \#Avg$) das soluções médias encontradas para cada instância. Observa-se que, como na comparação com o GAP da melhor solução encontrada, o algoritmo PSO_TS obteve melhores resultados que os algoritmos ITS [Palubeckis 2007], VNS [Brimberg et al. 2009], TIG [Lozano, Molina e García-Martínez 2011] e LTS_EDA [Wang et al. 2012] e na comparação com o algoritmo TS-MA [Wang et al. 2014], o resultado foi ligeiramente

inferior. Nós acreditamos que isto se deve a característica da meta-heurística PSO que necessita de um tempo maior de processamento para convergir para as melhores soluções conhecidas.

Para a análise das instâncias MDG-c, também consideradas instâncias de grandes porte, foi utilizado um tempo máximo de processamento de 480 segundos.

Instância	#Best	PSO_TS			
		Solução		GAP %	
		Melhor	Média	Melhor	Média
MDG-c_1	24924685	24926344	24918989	-0.0067	0.0229
MDG-c_2	24909199	24912546	24903365	-0.0134	0.0234
MDG-c_3	24900820	24896090	24890210	0.0190	0.0426
MDG-c_4	24904964	24909710	24900660	-0.0191	0.0173
MDG-c_5	24899703	24895704	24890870	0.0161	0.0355
MDG-c_6	43465087	43444251	43436138	0.0480	0.0666
MDG-c_7	43477267	43477267	43466250	0.0000	0.0253
MDG-c_8	43458007	43462452	43458214	-0.0102	-0.0005
MDG-c_9	43448137	43448137	43444905	0.0000	0.0074
MDG-c_10	43476251	43465561	43460248	0.0246	0.0368
MDG-c_11	67009114	67021132	67015085	-0.0179	-0.0089
MDG-c_12	67021888	67014170	66999552	0.0115	0.0333
MDG-c_13	67024373	67016057	67011065	0.0124	0.0199
MDG-c_14	67024804	67027878	67022870	-0.0046	0.0029
MDG-c_15	67056334	67056334	67050797	0.0000	0.0083
MDG-c_16	95637733	95637733	95633768	0.0000	0.0041
MDG-c_17	95645826	95569412	95556782	0.0800	0.0932
MDG-c_18	95629207	95532141	95518634	0.1016	0.1158
MDG-c_19	95633549	95594864	95590270	0.0405	0.0453
MDG-c_20	95643586	95580853	95573081	0.0656	0.0738
Média				0.017	0.033

Tabela 5.6: Resultados instância MDG-c (tempo $\leq 480s$)

Na *Tabela 5.6* tem-se os resultados encontrados para estas instâncias, onde as colunas indicam, respectivamente, a instância usada; a melhor solução conhecida na literatura; a melhor solução encontrada nos testes; a solução média encontrada; o GAP da melhor solução e o GAP da solução média em relação a melhor solução conhecida. GAPs negativos representam melhores resultados. As melhores soluções conhecidas na literatura foram retiradas de [Martí et al. 2013]. O GAP, para estas instâncias, foi

calculado de acordo com a *Equação* (5.2),

$$GAP = \left(\left(\frac{\#Best}{Soluc} \right) - 1 \right) \times 100 \quad (5.2)$$

onde:

$\#Best$ = melhor solução encontrada na literatura;

$Soluc$ = solução analisada.

Pode-se ver, o PSO_TS encontrou a melhor solução conhecida em 04 instâncias e, principalmente, melhorou os resultados conhecidos para seis casos. Em dois casos, até mesmo a sua qualidade média foi melhor do que os resultados conhecidos anteriormente relatados na literatura.

Os resultados obtidos nas instâncias MDG-c foram comparados com algoritmos *estado-da-arte*, que foram relatados em [Martí et al. 2013] e que foram executados por um tempo de processamento de 600 segundos em um computador Intel Core 2 Quad CPU Q 8300 with 6 GB de RAM.

Para determinar o tempo de parada do nosso algoritmo, nós usamos o *CPU Benchmarks - Single Thread Performance* que indicou que nosso computador tem um fator 1.246 mais rápido que aqueles usados nos testes por [Martí et al. 2013]. Então, nós determinamos o tempo de parada como 480 segundos para as instâncias MDG-c.

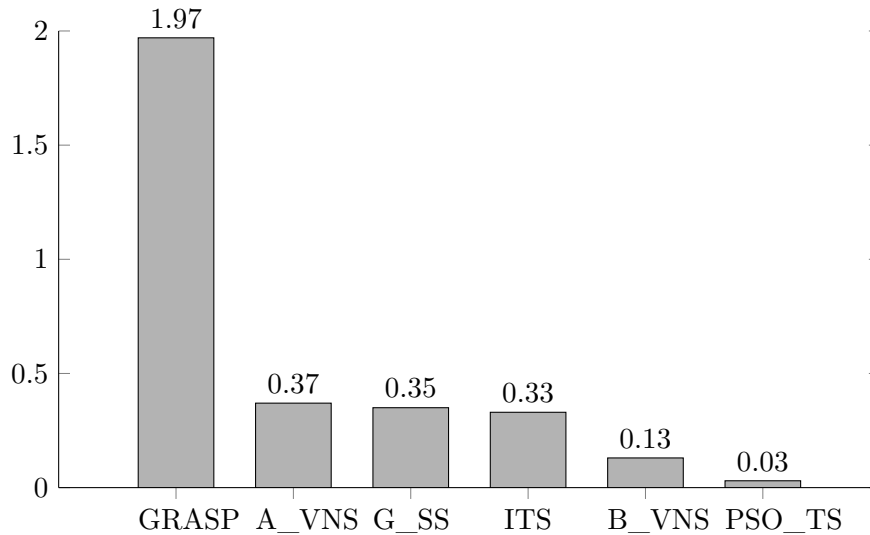


Figura 5.3: Comparação das instâncias MDG-c entre PSO_TS e outros algoritmos *estado-da-arte*

O gráfico da *Figura 5.3* mostra a comparação entre o GAP da melhor solução conhecida na literatura e a média das soluções encontrada no PSO_TS com os resultados das diferentes meta-heurísticas. Pode-se observar que o algoritmo PSO_TS obteve melhores resultados que todos os algoritmos comparados: GRASP_D2+I_LS [Duarte e Martí 2007], ITS [Palubeckis 2007], B_VNS [Brimberg et al. 2009], A_VNS [Aringhieri e Cordone 2011] e G_SS [Gallego et al. 2009].

Capítulo 6

Conclusão

Neste trabalho foi proposta uma nova abordagem para a resolução do Problema da Diversidade Máxima baseada na meta-heurística Otimização por Nuvem de Partículas juntamente com Busca Tabu. Para a geração das partículas iniciais foram implementadas uma heurística de construção gulosa e outra aleatória.

O método, denominado PSO_TS, mostra-se simples na sua implementação e os resultados demonstram que o algoritmo consegue excelente desempenho, obtendo, em todas as instâncias de médio porte testadas, as melhores soluções encontradas na literatura. Nas instâncias de grande porte MDG-a apenas em uma instância este resultado não foi alcançado. Nas instâncias MDG-c, o algoritmo melhorou a solução conhecida em seis casos e encontrou a melhor solução em outros quatro casos.

O PSO_TS apresentou-se bastante robusto em relação ao comportamento médio da qualidade das soluções obtidas, com um coeficiente de variação nulo para as instâncias de médio porte SOM-b. Nas instâncias de grande porte MDG-a, o coeficiente de variação foi de 0,00027 e de apenas 0,00012 para as instâncias MDG-c. Em resumo, uma variabilidade muito baixa.

Como passos futuros, visando melhorar a qualidade média das soluções e, principalmente, reduzir o tempo de processamento, têm-se como principais sugestões:

- Estudo do ajuste de parâmetros usados no algoritmo buscando melhorar o desempenho.
- Explorar novas estruturas de vizinhança para as partículas, onde o PBest é esco-

lhido a partir do melhor vizinho.

- Desenvolver abordagens paralelas para o PSO_TS para resolver as instâncias grandes.

Bibliografia

- [Agrafiotis 1997]AGRAFIOTIS, D. Stochastic algorithms for maximizing molecular diversity. *Journal of Chemical Information and Modeling*, Pennsylvania, v. 37, p. 841–851, 1997.
- [Andrade et al. 2003]ANDRADE, P. M. D. et al. GRASP for the maximum diversity problem. *Proceedings of the fifth metaheuristics international conference*, 2003.
- [Aringhieri e Cordone 2011]ARINGHIERI, R.; CORDONE, R. Comparing local search metaheuristics for the maximum diversity problem. *Journal of the Operational Research Society*, v. 62, p. 266–280, 2011.
- [Aringhieri, Cordone e Melzani 2008]ARINGHIERI, R.; CORDONE, R.; MELZANI, Y. Tabu search versus GRASP for the maximum diversity problem. *4OR*, v. 6, n. 1, p. 45–60, 2008. ISSN 16194500.
- [Besanko, Perry e Spady 1990]BESANKO, D.; PERRY, M. K.; SPADY, R. H. The logit model of monopolistic competition: Brand diversity. *The Journal of Industrial Economics*, Wiley, v. 38, n. 4, p. 397–415, 1990. ISSN 00221821, 14676451.
- [Bonotto e Cabral 2014]BONOTTO, E. L.; CABRAL, L. d. A. F. Algoritmo de otimização por nuvem de partículas aplicado ao problema da diversidade máxima. *Anais XLVI SBPO, Salvador, BA*, p. 2304–2314, 2014.
- [Brimberg et al. 2009]BRIMBERG, J. et al. Variable neighborhood search for the heaviest k-subgraph. *Computers and Operations Research*, v. 36, n. 11, p. 2885–2891, 2009.

- [Bunzel e Au 1987]BUNZEL, J. H.; AU, J. K. D. Diversity or Discrimination? Asian Americans in College. *The Public Interest*, v. 87, n. spring, p. 49–62, 1987.
- [Chuang, Hsiao e Yang 2011]CHUANG, L.-Y.; HSIAO, C.-J.; YANG, C.-H. Chaotic particle swarm optimization for data clustering. *Expert Systems with Applications*, Elsevier Ltd, v. 38, n. 12, p. 14555–14563, 2011. ISSN 09574174.
- [Coelho Filho 2013]COELHO FILHO, O. P. Uma nova abordagem híbrida do algoritmo de otimização por enxame de partículas com busca local iterada para o problema de clusterização de dados. *Journal of Chemical Information and Modeling*, v. 53, n. 9, p. 1689–1699, 2013. ISSN 1098-6596.
- [Duarte e Martí 2007]DUARTE, A.; MARTÍ, R. Tabu search and GRASP for the maximum diversity problem. *European Journal of Operational Research*, v. 178, n. 1, p. 71–84, apr 2007. ISSN 03772217.
- [Duarte, Silva e Costa 2008]DUARTE, I. L.; SILVA, G. C. da; COSTA, T. A. Algoritmos heurísticos para o problema da diversidade máxima. *Anais do XL SBPO. João Pessoa, PB*, p. 1126–1137, 2008.
- [Gallego et al. 2009]GALLEGO, M. et al. Hybrid heuristics for the maximum diversity problem. *Computational Optimization and Applications*, v. 44, n. 3, p. 411–426, 2009. ISSN 09266003.
- [Ghosh 1996]GHOSH, J. B. Computational aspects of the maximum diversity problem. *Operations Research Letters*, v. 19, n. 4, p. 175–181, 1996. ISSN 01676377.
- [Ghyczy 1985]GHYCZY, T. V. Product diversity and proliferation as a new mode of competing in the motor car. *International Journal of Vehicle Design*, v. 6, n. 4-5, p. 423–425, 1985.
- [Glover 1986]GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, v. 13, n. 5, p. 533–549, 1986. ISSN 03050548.

- [Glover, Kuo. e Dhir 1998]GLOVER, F.; KUO., C.; DHIR, K. Heuristic algorithms for the maximum diversity problem. *Journal of Information and Optimization Sciences*, v. 19, n. 1, 1998.
- [Glover e Laguna 1997]GLOVER, F.; LAGUNA, M. Tabu search. *Kluwer Academic Publishers*, 1997.
- [Hassin, Rubinstein e Tamir 1997]HASSIN, R.; RUBINSTEIN, S.; TAMIR, A. Approximation algorithms for maximum dispersion. *Operations Research Letters*, v. 21, n. 3, p. 133–137, 1997. ISSN 01676377.
- [Keely 1989]KEELY, A. From experience-maxi-niching the way to a strong brand positioning according to systemic dynamics. *Journal of Product Innovation Management*, v. 6(3), p. 202–206, 1989.
- [Kennedy e Eberhart 1995]KENNEDY, J.; EBERHART, R. Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, v. 4, p. 1942–1948 vol.4, 1995. ISSN 19353812.
- [Kuo, Glover e Dhir 1993]KUO, C.-C.; GLOVER, F.; DHIR, K. S. Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming. *Decision Sciences*, v. 24, n. 6, p. 1171–1185, 1993. ISSN 0011-7315.
- [Lozano, Molina e García-Martínez 2011]LOZANO, M.; MOLINA, D.; GARCÍA-MARTÍNEZ, C. Iterated greedy for the maximum diversity problem. *European Journal of Operational Research*, v. 214, n. 1, p. 31–38, oct 2011. ISSN 03772217.
- [Luke 2013]LUKE, S. *Essentials of Metaheuristics*. [S.l.: s.n.], 2013. 1–235 p. ISSN 1389-2576. ISBN 9781300549628.
- [Martí, Gallego e Duarte 2010]MARTÍ, R.; GALLEGO, M.; DUARTE, A. A branch and bound algorithm for the maximum diversity problem. *European Journal of Operational Research*, v. 200, n. 1, p. 36–44, 2010. ISSN 03772217.
- [Martí et al. 2013]MARTÍ, R. et al. Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics*, v. 19, n. 4, p. 591–615, 2013. ISSN 13811231.

- [McConnell 1988]MCCONNELL, S. The new battle over immigration. *Fortune*, v. 117(10), p. 89–102, 1988.
- [McConnell 2013]MCCONNELL, S. The new battle over immigration. *Fortune*, June, 16, 2013.
- [Palubeckis 2007]PALUBECKIS, G. Iterated tabu search for the maximum diversity problem. *Applied Mathematics and Computation*, v. 189, n. 1, p. 371–383, jun 2007.
- [Pearce 1987]PEARCE, D. Economic economics and genetic diversitys. *Futures*, v. 19(6), December 1987.
- [Porter W. M.. Rawal 1975]PORTER W. M.. RAWAL, K. M. R. K. . W. H. C. . W. R. C. Cowpea germplasm catalog no. 1. *International Institute of Tropical Agriculture*, 1975.
- [Ramirez 1979]RAMIREZ, A. Cream of the crop? *Wall Street Journal*, p. 29, april, 6, 1979.
- [Silva, Ochi e Martins 2005]SILVA, C. G.; OCHI, L. S.; MARTINS, L. S. Proposta e avaliação de heurísticas GRASP para o problema da diversidade máxima. *Universidade Federal Fluminense (UFF)*, p. 321–360, 2005.
- [Silva, Ochi e Martins 2004]SILVA, G.; OCHI, L.; MARTINS, S. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. *Lecture Notes on Computer Science*, p. 498–512, 2004.
- [Souza 2006]SOUZA, G. R. Uma Abordagem por Nuvem de Partículas para Problemas de Otimização Combinatória. *Dissertação de Mestrado*, UFRN, Natal, RN, 2006.
- [Souza 2010]SOUZA, M. J. F. Manual de Computação Evolutiva e Heurística. Notas de Aula, p. 1–40, 2010.
- [Swierenga 1977]SWIERENGA, R. P. Ethnicity in historical perspective. *Social Science*, Pi Gamma Mu, International Honor Society in Social Sciences, v. 52, n. 1, p. 31–44, 1977. ISSN 00377848, 23320400.

- [Takane 2011]TAKANE, B. Um algoritmo exato para o problema da diversidade máxima. *Dissertação de Mestrado*, UFMG, Belo Horizonte, MG, Brasil, 2011.
- [Thomas 1990]THOMAS, R. R. From affirmative action to affirming diversity. *Harvard Business Review*, v. 68(2), p. 107–117, 1990.
- [Unkel 1985]UNKEL, W. C. Natural diversity and national forest planning. *Natural Areas Journal*, p. 8–13, october, 5, 1985.
- [Wang et al. 2012]WANG, J. et al. Learnable tabu search guided by estimation of distribution for maximum diversity problems. *Soft Computing*, v. 16, n. 4, p. 711–728, 2012. ISSN 14327643.
- [Wang et al. 2014]WANG, Y. et al. A tabu search based memetic algorithm for the maximum diversity problem. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 27, p. 103–114, 2014.
- [Wu e Hao 2013]WU, Q.; HAO, J. K. A hybrid metaheuristic method for the Maximum Diversity Problem. *European Journal of Operational Research*, Elsevier B.V., v. 231, n. 2, p. 452–464, 2013. ISSN 03772217.