# OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS E BUSCA TABU PARA O PROBLEMA DA DIVERSIDADE MÁXIMA

**Edison Luiz Bonotto**
Programa de Pós-Graduação em Informática - PPGI
Universidade Federal da Paraíba - UFPB
58051-900 - Paraíba - PB - Brazil
bonotto@ci.ufpb.br

**Lucídio dos Anjos Formiga Cabral**
Programa de Pós-Graduação em Informática - PPGI
Universidade Federal da Paraíba - UFPB
58051-900 - Paraíba - PB - Brazil
lucidio@ci.ufpb.br

**RESUMO**

Este artigo descreve um algoritmo híbrido (PSO_TS) que utiliza Otimização por Nuvem de Partículas (PSO) e Busca Tabu (TS) para a resolução do Problema da Diversidade Máxima (MDP). O 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. A abordagem PSO simula o comportamento de um bando de pássaros em voo com seu movimento localmente aleatório, mas globalmente determinado, para encontrar máximos locais. A utilização do PSO_TS alcança os melhores resultados encontrados para as instâncias testes presentes na literatura, demonstrando assim ser competitivo com os melhores algoritmos, em termos de qualidade das soluções.

**PALAVRAS CHAVE. Problema da Diversidade Máxima (MDP), Otimização por Nuvem de Partículas (PSO), Busca Tabu (TS).**

**Tópicos: Otimização Combinatória, Meta-heurística**

**ABSTRACT**

This article describes a hybrid algorithm (PSO_TS) that uses Particle Swarm Optimization (PSO) and Tabu Search (TS) to solve the Maximum Diversity Problem (MDP). The MDP is a problem in the area of combinatorial optimization which aims to select a preset number of elements of a given set such that the sum of the differences between the selected elements is the highest possible. The approach PSO simulates the behavior of a flock of birds in flight with its random movement locally, but globally determined to find local maxima. 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.

**KEYWORDS. Maximum Diversity Problem (MDP), Particle Swarm Optimization (PSO), Tabu Search (TS).**

**Paper topics: Combinatorial optimization, Metaheuristics**

## 1. Introdução

Este artigo descreve uma aplicação baseada em um algoritmo de Otimização por Nuvem de Partículas com refinamento das soluções através de um procedimento de busca local baseado em Busca Tabu para tratar o Problema da Diversidade Máxima.

O Problema da Diversidade Máxima encontra aplicações nos mais variados campos da ciência, como na indústria farmacêutica [Agrafiotis, 1997], na biologia, onde através de cruzamentos seletivos pode-se extrair características desejáveis em animais, plantas e outros seres vivos [Takane, 2011], e nas empresas, onde pode ser usado para selecionar equipes de trabalho heterogêneas ou para descobrir qual a combinação de produtos que gera o maior lucro possível.

Há na literatura diversas abordagens para solução do problema com métodos aproximados, dos quais podemos citar a meta-heurística GRASP [Ghosh, 1996] e [Silva et al., 2007], Busca Tabu em [Duarte e Martí, 2007], [Palubeckis, 2007], [Aringhieri e Cordone, 2011] e [Wang et al., 2012], meta-heurística Iterated Local Search em [Duarte et al., 2008], algoritmo Iterated Greedy em [Lozano et al., 2011], Scatter Search em [Gallego et al., 2009], Busca com Vizinhança Variável em [Aringhieri e Cordone, 2011], Otimização por Nuvem de Partículas em [Bonotto e Cabral, 2014] e algoritmo Memético [Wang et al., 2014].

### 1.1. O Problema da Diversidade Máxima

O Problema da Diversidade Máxima (MDP), segundo [Duarte et al., 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 $N = \{1,...,n\}$, de tal forma que a diversidade entre os seus elementos selecionados seja máxima. O MDP é um problema NP-difícil, 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.

Para calcular a diversidade como uma distância é comum usar uma métrica normatizada entre os atributos dos elementos de $N$. Uma métrica de distância normalmente utilizada é a norma Euclidiana. Usamos a *Equação* (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 1*), $a_{ik}$ denota o k-ésimo atributo do elemento $i$ e $a_{jk}$ denota o k-ésimo atributo do elemento $j$.

$$d_{ij} = \sqrt[p]{\sum_{k \in R} (|a_{ik} - a_{jk}|)^p} \tag{1}$$

#### 1.1.1. 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.

| Classificação | Atributos | | | | |
|---|---|---|---|---|---|
| | Sexo | Idade | Região Origem | Áreas de Conhecimento | Experiência(anos) |
| 1 | Masculino | 16-19 | Norte | Ciências Exatas | 0 - 1 |
| 2 | Feminino | 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 Agrárias | 8 - 10 |

Tabela 1: Mapeamento de atributos pessoais

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. A *Tabela 1* mostra o mapeamento dos atributos e a *Tabela 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: Atributos mapeados dos candidatos

Com base na *Tabela 2*, calculamos a distância entre cada par de candidatos. A *Tabela 3* mostra os valores resultantes do cálculo da distância Euclidiana usando a *Equação* (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 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 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 4: Diversidade máxima com todas as soluções possíveis

### 1.2. Trabalhos Anteriores

[Ghosh, 1996] propôs um algoritmo GRASP para solucionar o Problema da Diversidade Máxima e implementou um algoritmo exato para comparar a eficiência da heurística proposta. Os resultados computacionais mostraram que o GRASP proposto alcança uma solução ótima ou uma solução muito próxima da ótima para instâncias de pequeno porte $(n \leq 40)$.

[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. 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.

[Silva et al., 2005] propuseram três heurísticas de construção e duas de busca local para analisar seus desempenhos na meta-heurística GRASP. Resultados computacionais mostram 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 um algoritmo baseado em Busca Tabu, que incorporava estruturas de memória para as fases de construção e de refinamento. Os experimentos computacionais com instâncias de médio e grande porte mostraram que a proposta superou as melhores heurísticas relatadas na literatura demandando baixos tempos computacionais.

[Aringhieri et al., 2008] propuseram explorar uma abordagem que consiste em aprimorar a fase de Busca Local, através da adoção de um algoritmo de Busca Tabu, mantendo um procedimento de inicialização muito simples. Os resultados computacionais mostraram que a Busca Tabu alcança melhores resultados e tempos computacionais menores em relação aos relatados para o GRASP.

[Duarte et al., 2008] propuseram algoritmos heurísticos baseados nos conceitos 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 resultados computacionais mostram que os algoritmos propostos apresentam um bom desempenho quando comparados com as melhores heurísticas da literatura.

[Lozano et al., 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. 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. Os resultados computacionais mostraram que o método proposto demonstra ser competitivo frente aos métodos exatos descritos na literatura.

[Wang et al., 2014] apresentou um algoritmo Memético altamente eficaz para o problema da diversidade máxima baseado na abordagem Busca Tabu. Análise de comparações com algoritmos do state-of-the-art demonstram estatisticamente que o algoritmo compete muito fortemente com o desempenho dos melhores algoritmos.

## 2. Otimização por Nuvem de Partículas

A Otimização por Nuvem de Partículas (*Particle Swarm Optimization* - PSO) é uma técnica de computação desenvolvida por James Kennedy e Russell Eberhart, [Kennedy e Eberhart, 1995], na qual pretendiam simular graficamente o comportamento de um bando de pássaros em voo, com seu movimento localmente aleatório, mas globalmente determinado.

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.

No algoritmo PSO GBest, 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).

$$V_{ij}^{k+1} = w.V_{ij}^{k} + c_1.r_1.(PBest_{ij}^{k} - X_{ij}^{k}) + c_2.r_2.(GBest_{j}^{k} - X_{ij}^{k}) \quad (2)$$

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, $C_1$ e $C_2$ são constantes de aceleração positivas usadas para balancear a contribuição dos componentes cognitivo e social respectivamente, $r_1$ e $r_2$ são valores randômicos no intervalo $[0, 1]$ distribuídos uniformemente e $w$ é uma constante que representa o peso inercial na velocidade da partícula. A nova posição da partícula se dá pela soma do novo vetor velocidade ao vetor posição corrente.

## 3. Algoritmo PSO_TS

O algoritmo proposto (PSO_TS) utiliza uma versão da meta-heurística PSO combinada com Busca Tabu para refinar as soluções. Seu funcionamento é o seguinte: Inicialmente são geradas $q$ partículas de tamanho $m$, onde $40\%$ delas - sendo no mínimo uma partícula - são criadas através de um algoritmo guloso simples e, as partículas restantes, $60\%$ delas, os $m$ elementos são escolhidos aleatoriamente obedecendo aos limites da instância. A heurística gulosa é descrita na *Seção 3.1* e a escolha do valor da constante $q$ é discutida na *Seção 4.1.1*. A cada iteração $k$, a partícula desloca-se pelo espaço multidimensional conforme a *Equação* (3).

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \quad (3)$$

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, e é calculada pela *Equação* (4).

$$V_{id}^{k+1} = W * V_{id}^{k} + c_1 * Cr * (P_{id}^{k} - X_{id}^{k}) + c_2 * (1 - Cr) * (P_{gd}^{k} - X_{id}^{k}) \quad (4)$$

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 $W = 0.9 - ((0.9 - 0.1)/k) * q$. Esta equação, baseada em[Coelho Filho, 2013], faz com que nas $k$ iterações iniciais o valor de $W$ - entre $[0.1, ..., 0.9]$ - 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 $Cr = y * Cr * (1 - Cr)$, descrita por [Chuang et al., 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 $[0, ..., 1]$. 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 de $C_1$ e $C_2$ são discutidos na *Seção 4.1.2*; $X_{id}^{k}$ 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 é posicionado na borda deste espaço. Também acontecem casos em que a nova posição do elemento coincide com a posição de outro elemento da mesma partícula. Sendo assim, uma nova posição desse elemento é escolhida aleatoriamente, evitando que a partícula fique "presa", sem poder deslocar-se no espaço.

A cada iteração $k$ , 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 iniciado. O número de chamadas a TS depende do tamanho das instâncias: Para as instâncias grandes, a partícula com maior diversidade e 01, escolhida aleatoriamente entre as outras 4 melhores partículas, servirão de solução inicial para a TS. Já para as instâncias médias, também é escolhida a melhor partícula, mas, 02 são escolhidas aleatoriamente entre as outras 5 melhores partículas. Esta segunda configuração melhora o tempo de processamento para encontrar as soluções nas instâncias médias. O procedimento TS é descrito com detalhes na *Seção 3.2*. Ao término da Busca Tabu, $GBest$ e $PBest$ são atualizados.

Caso uma partícula não apresente melhora a cada 50 iterações, ela é substituída por outra gerada de maneira totalmente aleatória. O algoritmo termina quando atingir um número $kmax$ de iterações ou atingir o tempo $t$ máximo de processamento. Definimos $kmax = max(100, 30 * m)$ e *t=15s* para as instâncias médias e $t = 900s$ para as instâncias grandes. O último $GBest$ encontrado é a solução do problema. O pseudocódigo do algoritmo PSO_TS é mostrado na *Figura 1*.

1. **Entrada:** Matriz $M_{nxn}$ $(d_{ij})$ com cardinalidade $m \leq n$
2. **Saída:** Melhor solução encontrada $Gb*$
3. $P = \{x^1, ..., x^{max(1,int(m*0.4))}\} \leftarrow$ Gera_particulas_gulosas() /* Seção 3.1 */
4. $P = \{x^{max(1,int(m*0.4))}, ..., x^q\} \leftarrow$ Gera_particulas_aleatorias()
5. $Pb^* = \{x^1, ..., x^q\} \leftarrow P = \{x^1, ..., x^q\}$
6. $Gb^* = argmax\{f(x^i)|i = \{1, ..., q\}$
7. Sem_Melhora$(i_1, ..., i_q) \leftarrow 0; k = 0$
8. While $k < kmax$ do
9. For $i = 0$ até $q - 1$
10. Sem_Melhora(i)++
11. If Sem_Melhora$(i) > 50$ então
12. Reconstroi_Part$(i)$
13. Move $P[i]$ /* Eq. (3) */
14. Atualiza $Gb^*$ e $Pb^*(i)$
15. EndFor
16. Ordena $P = \{x^1, ..., x^q\}$ (Descrescente pela diversidade máxima)
17. $s = 0$;
18. For $i = 0$ até *Nr_Ch_TS*
19. $P(s) = TS(P(s))$ /* Seção 3.2 */
20. Atualiza $Gb^*$ e $Pb^*(s)$
21. $s = rand(1 - 5)$
22. EndFor
23. If Tempo_processamento $> t$ então
24. Break
25. EndWhile
26. Return $Gb^*$

Figura 1: Pseudocódigo do algoritmo PSO_TS

### 3.1. Algoritmos de Construção

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 um

elemento $N \setminus U$ contribuiria ao fazer parte da solução parcial, conforme *Equação* (5):

$$S_i = \sum_{j \in U} d_{ij} \tag{5}$$

(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* (5), assim como a diversidade máxima da solução parcial; (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).

### 3.2. 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 $x = \{x_1, ..., x_n\}$ 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 $x_i \in \cup$ e $x_j \in Z$. A TS utiliza um mecanismo que determina rapidamente o ganho do movimento de troca, similar ao utilizado por [Aringhieri et al., 2008].

Inicialmente, é empregado um vetor $\triangle$ para armazenar a variação da função objetivo decorrente do movimento de $x_i$ do subconjunto atual de $U \setminus Z$ para o outro subconjunto $Z \setminus U$. O vetor é inicializado conforme a equação que segue:

$$\Delta_i = \begin{cases} \sum_{j \in U} -d_{ij} (x_i \in U) \\ \sum_{j \in U} d_{ij} (x_i \in Z) \end{cases} \tag{6}$$

O ganho do movimento entre duas variáveis $x_i \in U$ e $x_j \in Z$ pode ser calculado usando a fórmula $\delta_{ij} = \triangle_i + \triangle_j - d_{ij}$. Após realizado o movimento de troca entre as variáveis $x_i$ e $x_j$ o vetor $\triangle$ deve ser atualizado utilizando a *Equação* (7).

$$\triangle_k = \begin{cases} -\triangle_i + d_{ij} (\text{k=i}) \\ -\triangle_j + d_{ij} (\text{k=j}) \\ \triangle_k + d_{ik} + d_{jk} (\text{k \# \{ i,j\}}, x_k \in \text{U}) \\ \triangle_k - d_{ik} + d_{jk} (\text{k \# \{ i,j\}}, x_k \in \text{Z}) \end{cases} \tag{7}$$

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 vector $\triangle$. 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 $\triangle$ 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 $cls$ 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.

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]. A Lista Tabu é definida por $n$ elementos de um vector $T$, a saber, $min(\sqrt{m}, \sqrt{n-m})$. 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 $\alpha$ foi definido como $6m$. A *Figura 2.* mostra o pseudocódigo da Busca Tabu.

1. **Entrada:** Solução PSO
2. **Saída:** Solução melhorada
3. Iter $\leftarrow 0$
4. $T[x_i, ..., x_n] \leftarrow 0$ (Lista Tabu)
5. Criar e inicializar o vetor $\triangle$ de acordo com a *Eq.* (6).
6. Construir a lista de candidatos $UCL$ e $ZCL$, gravadas em ordem decrescente em relação aos valores de $\triangle$
7. Executa o movimento de troca de $x_i$ para $x_j$ que proporcione a melhor troca de acordo com a equação $\delta_{ij} = \triangle_i + \triangle_j - d_{ij}$ e desde que $T[x_i] \leq Iter$ e $T[x_j] \leq Iter$
8. Atualiza o vetor $\triangle$ de acordo com a *Eq.* (7)
9. Atualiza $T[x_i]$ e $T[x_j]$; $Iter++$
10. Se $Iter < \alpha$ retorna ao passo 6

Figura 2: Pseudocódigo do Algoritmo de Busca Tabu

## 4. Testes e Resultados

Os testes para a escolha dos valores para as variáveis críticas foram realizados em máquinas virtuais do *Google* com processador Intel (R) Xeon (R) CPU @ 2.30GHz com 1 núcleo, 3,75 GB de memória RAM, 10 GB de disco rígido (HD), usando o sistema operacional (SO) Debian GNU/Linux 8.3 (jessie) e o compilador g++ (Debian 4.7.2-5). Para os resultados finais foi utilizado um processador Intel (R) Core (TM) i7 @ 3.07GHz com 4 núcleos, 8 GB memória, HD 2 TB Gb e SO Linux Ubuntu versão 4.2.0-35. Apenas 1 núcleo do processador foi utilizado. Os testes foram realizados utilizando as 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 $[0, ..., 10]$, e as 20 instâncias SOM-b, criadas por [Silva et al., 2004], e que contém uma população de tamanho $n = 100$, $n = 200$, $n = 300$, $n = 400$ e $n = 500$. A diversidade entre dois elementos quaisquer nas instâncias foi escolhida aleatoriamente e uniformemente distribuídas com inteiros entre [0,...,9].

### 4.1. Parâmetros

O PSO tem alguns parâmetros que devem ser definidos e que alteram sensivelmente a qualidade das soluções. Neste tópico, analisamos, através de testes sobre as instâncias, a quantidade ideal de partículas $q$ que são criadas ao iniciar o PSO_TS e o melhor valor para cada um dos

parâmetros $C1$ e $C2$, que definem a velocidade com que as partículas se deslocam em direção à $GBest$ e $PBest$ durante as iterações do PSO.

### 4.1.1. Número de Partículas $q$

Para definir a quantidade $q$ de partículas que serão criadas foram realizadas 3 execuções do algoritmo, durante 3600 segundos, sobre as 20 instâncias MDG-a e testados valores para $q = 40m$, $q = 30m$, $q = 20m$, $q = 10m$, $q = m$. O GAP médio entre os resultados em relação aos melhores resultados da literatura foram semelhantes. Para $q = 40m$, $q = 30m$, $q = 20m$, $q = 10m$ e $q = m$ foram encontrados os valores $0.0030\%$, $0.0058\%$, $0.0031\%$, $0.0007\%$ e $0.0002\%$, respectivamente. Com base nisto, a escolha de $q$ baseou-se principalmente no tempo médio de processamento para encontrar a solução. Os tempos médios encontrados foram $2434s$, $2348s$, $2046s$, $1542s$ e $1551s$. O valor de $q = 10m$, além de apresentar o menor tempo de processamento, também apresenta um dos GAPs - (*(melhor solução conhecida/melhor solução encontrada)-1)*100* - mais baixo e é, portanto, o valor escolhido. Nas instâncias médias foram testados valores para $q = 10m$ e $q = 30m$, com um mínimo de 200 partículas, e realizados 10 teste sobre as 20 instâncias SOM-b. O tempo máximo necessário para que todos os testes encontrassem a melhor solução conhecida foi de 56,6s para $q = 10m$ e 6,95 segundos para $q = 30m$.

### 4.1.2. Parâmetros $C1$ e $C2$

Foram realizados testes no conjunto de 10 instâncias grandes (MDG-a_21 à MDG-a_30) e testadas combinações de valores para $C1 = 1.00$, $1.20$ e $1.40$ e $C2 = 1.00$, $1.20$, $1.40$, $1.60$, $1.80$ e $2.00$. Conforme podemos verificar no gráfico de barras mostrados na *Figura 3*, onde é analisado o GAP médio entre os resultados encontrados e os melhores da literatura, a melhor combinação foi obtida com $C1 = 1.00$ e $C2 = 1.80$. Para as instâncias menores os testes foram realizados nas 20 instâncias SOM-b e comparado usando o valor de $C1 = 1.05$ e $C2 = 1.20$, conforme [Bonotto e Cabral, 2014], e com $C1 = 1.00$ e $C2 = 1.80$, resultado empregado para as instâncias grandes. O menor tempo médio para encontrar a melhor solução foi de $8,48s$, com $C1 = 1.05$ e $C2 = 1.20$. Já para $C1 = 1.00$ e $C2 = 1.8$ o tempo obtido foi de $14,49s$. Portanto, para as instâncias SOM-b foi utilizado o valor de $C1 = 1.05$ e $C2 = 1.20$.

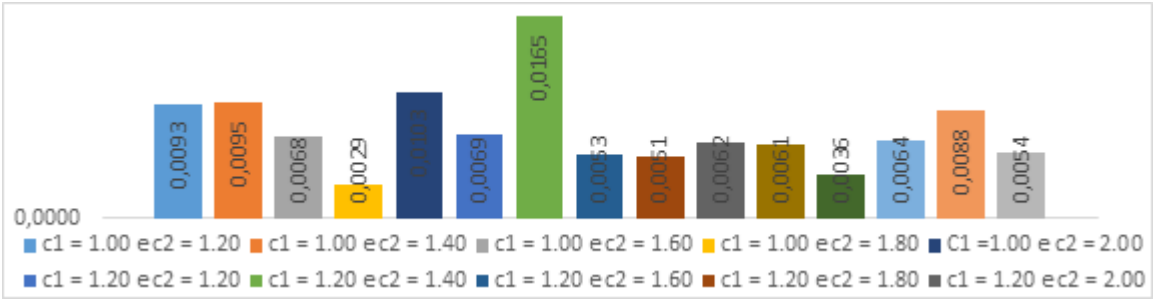


Figura 3: Comparativo dos parâmetros $C1$ e $C2$

## 4.2. Resultados

Para verificar a eficiência do algoritmo proposto foram empregadas as instâncias de médio porte SOM-b e testadas soluções com tamanho $m = 10\%n$, $m = 20\%n$, $m = 30\%n$ e $m = 40\%n$. As melhores soluções conhecidas foram retiradas de [Duarte et al., 2008]. A meta-heurística PSO_TS foi executada 10 vezes com um tempo máximo de processamento de 15 segundos. Para as instâncias de grande porte foram utilizadas as 20 instâncias MDG-a e realizadas 15 execuções para um tempo limite de $900s$ e 10 execuções para 1800 e 3600 segundos. Para a diversidade máxima foi considerado o resultado com a maior diversidade.

A *Tabela 5* mostra os melhores resultados obtidos para as instâncias de médio porte SOM-b, onde a *coluna 1* indica a instância utilizada; a *coluna 2*, a dimensão $n$ desta instância; a *coluna*

| Instâncias | n | m | Solução Conhecida | PSO_TS | | | |
|---|---|---|---|---|---|---|---|
| | | | | Média | Tempo (segundos) | | |
| | | | | | Mín | Méd | Máx |
| SOM-b_1_n100_m10 | 100 | 10 | 333 | 333 | 0,00 | 0,00 | 0,00 |
| SOM-b_2_n100_m20 | 100 | 20 | 1195 | 1195 | 0,00 | 0,00 | 0,01 |
| SOM-b_3_n100_m30 | 100 | 30 | 2457 | 2457 | 0,00 | 0,01 | 0,02 |
| SOM-b_4_n100_m40 | 100 | 40 | 4142 | 4142 | 0,02 | 0,04 | 0,09 |
| SOM-b_5_n200_m20 | 200 | 20 | 1247 | 1247 | 0,00 | 0,05 | 0,22 |
| SOM-b_6_n200_m40 | 200 | 40 | 4450 | 4450 | 0,05 | 0,18 | 0,67 |
| SOM-b_7_n200_m60 | 200 | 60 | 9437 | 9437 | 0,07 | 0,09 | 0,16 |
| SOM-b_8_n200_m80 | 200 | 80 | 16225 | 16225 | 0,14 | 0,17 | 0,20 |
| SOM-b_9_n300_m30 | 300 | 30 | 2694 | 2694 | 0,03 | 0,44 | 2,16 |
| SOM-b_10_n300_m60 | 300 | 60 | 9689 | 9689 | 0,09 | 0,67 | 1,74 |
| SOM-b_11_n300_m90 | 300 | 90 | 20743 | 20743 | 0,26 | 0,48 | 0,96 |
| SOM-b_12_n300_m120 | 300 | 120 | 35881 | 35881 | 0,56 | 1,16 | 2,89 |
| SOM-b_13_n400_m40 | 400 | 40 | 4658 | 4658 | 0,10 | 0,82 | 2,13 |
| SOM-b_14_n400_m80 | 400 | 80 | 16956 | 16956 | 0,23 | 1,48 | 2,76 |
| SOM-b_15_n400_m120 | 400 | 120 | 36317 | 36317 | 0,51 | 1,95 | 4,92 |
| SOM-b_16_n400_m160 | 400 | 160 | 62487 | 62487 | 1,35 | 4,00 | 6,84 |
| SOM-b_17_n500_m50 | 500 | 50 | 7141 | 7141 | 0,27 | 1,44 | 2,47 |
| SOM-b_18_n500_m100 | 500 | 100 | 26258 | 26258 | 0,58 | 1,60 | 3,04 |
| SOM-b_19_n500_m150 | 500 | 150 | 56572 | 56572 | 0,92 | 2,14 | 6,80 |
| SOM-b_20_n500_m200 | 500 | 200 | 97344 | 97344 | 2,35 | 3,98 | 6,95 |

Tabela 5: Resultados das instâncias SOM-b (Tempo $< 15s$)

| Instâncias | Solução | PSO_TS | | | | | |
|---|---|---|---|---|---|---|---|
| | | Solução | | C. Variação | Tempo (segundos) | | |
| | Conhecida | Melhor | Média | | Mín | Méd | Máx |
| MDG-a_21 | 114271 | 114271 | 114258,5 | 0,00012 | 139,9 | 496,0 | 900,0 |
| MDG-a_22 | 114327 | 114327 | 114317,9 | 0,00029 | 130,0 | 426,8 | 890,2 |
| MDG-a_23 | 114195 | 114195 | 114179,9 | 0,00015 | 199,1 | 429,1 | 805,6 |
| MDG-a_24 | 114093 | 114093 | 114071,0 | 0,00016 | 186,6 | 503,0 | 872,4 |
| MDG-a_25 | 114196 | 114196 | 114165,3 | 0,00021 | 246,3 | 554,0 | 836,9 |
| MDG-a_26 | 114265 | 114265 | 114252,0 | 0,00011 | 119,0 | 379,8 | 715,3 |
| MDG-a_27 | 114361 | 114361 | 114361,0 | 0,00000 | 158,0 | 280,9 | 526,0 |
| MDG-a_28 | 114327 | 114327 | 114297,3 | 0,00041 | 164,6 | 514,9 | 900,0 |
| MDG-a_29 | 114199 | 114199 | 114190,5 | 0,00012 | 210,2 | 495,2 | 894,5 |
| MDG-a_30 | 114229 | 114229 | 114221,6 | 0,00007 | 69,7 | 489,3 | 875,2 |
| MDG-a_31 | 114214 | 114214 | 114194,1 | 0,00017 | 198,8 | 458,7 | 859,9 |
| MDG-a_32 | 114214 | 114214 | 114191,9 | 0,00021 | 194,2 | 525,0 | 859,1 |
| MDG-a_33 | 114233 | 114233 | 114225,8 | 0,00010 | 277,2 | 586,1 | 862,2 |
| MDG-a_34 | 114216 | 114216 | 114211,7 | 0,00007 | 235,4 | 522,2 | 882,2 |
| MDG-a_35 | 114240 | 114240 | 114235,6 | 0,00005 | 104,3 | 578,7 | 896,7 |
| MDG-a_36 | 114335 | 114335 | 114328,7 | 0,00007 | 166,8 | 417,0 | 849,5 |
| MDG-a_37 | 114255 | 114255 | 114240,8 | 0,00010 | 110,6 | 537,4 | 899,8 |
| MDG-a_38 | 114408 | 114408 | 114407,2 | 0,00001 | 176,5 | 406,5 | 717,1 |
| MDG-a_39 | 114201 | 114201 | 114198,3 | 0,00006 | 89,9 | 432,2 | 793,3 |
| MDG-a_40 | 114349 | 114349 | 114328,0 | 0,00027 | 221,0 | 589,9 | 830,0 |

Tabela 6: Resultados das instâncias MDG-a (Tempo $\simeq 900s$)

*3*, a quantidade $m$ de elementos da partícula (solução); a *coluna 4*, a melhor solução (diversidade máxima) conhecida na literatura; a *coluna 5*, a média das soluções encontradas nos testes e as *coluna 6, 7 e 8*, respectivamente, o tempo mínimo, médio e máximo de processamento até encontrar a solução. Podemos observar que em todas as instâncias testadas e em $100\%$ dos 10 testes executados em cada uma das delas, o algoritmo PSO_TS encontrou a melhor solução conhecida na literatura, ou seja, um desvio padrão de $0\%$. O tempo mínimo para encontrar a melhor solução variou de 0 a 6,95 segundos, dependendo do valor de $n$ e $m$ da instância.

Na *Tabela 6* temos os resultados encontrados para as instâncias de grande porte MDG-a, onde a *coluna 1* indica a instância utilizada; a *coluna 2*, a melhor solução (diversidade máxima) conhecida na literatura; as *colunas 3 e 4*, a melhor solução e a média das soluções encontradas nos testes, respectivamente; a *coluna 5*, mostra o coeficiente de variação, que corresponde ao desvio padrão dividido pela média, entre as 15 execuçõese as *colunas 6, 7 e 8*, respectivamente, o tempo mínimo, médio e máximo de processamento até encontrar a solução. Podemos observar que todos os melhores resultados existentes na literatura foram encontrados. Estes melhores resultados foram encontrados em $47\%$ dos 15 testes executados em cada uma das instâncias. O coeficiente de variação médio entre as soluções, obtido pela razão entre desvio padrão e média, ficou em $0,00014$. Na instância MDG-a_27, todos os 15 testes realizados sobre ela obtiveram os melhores resultados da literatura.

Foram, também, realizados 10 testes em cada uma das instâncias MDG-a com um tempo de processamento maior: $1800s$ e $3600s$. Em ambos os testes as melhores soluções da literatura foram encontradas. Para os testes com $1800s$, o coeficiente de variação encontrado foi de $0,00017$ e a percentagem de vezes que a melhor solução foi encontrada foi de $61\%$. Já para os testes com limite de tempo de $3600s$, o coeficiente de variação foi ínfimo, de apenas $0,4x10^{-4}$, em $77\%$ dos testes foi encontrada a melhor solução e, em $45\%$ das instâncias, todos os 10 testes encontraram a melhor solução (*Tabela 7*).

| Tempo de processamento | (1) | (2) | (3) |
|---|---|---|---|
| 900s (15 minutos) | 0,00014 | 47% | 5% |
| 1800s (30 minutos) | 0,00017 | 61% | 10% |
| 3600s (1 hora) | 0,00004 | 77% | 45% |

(1) Coeficiente de Variação (2) Percentagem de vezes que a melhor solução foi encontrada
(3) Percentagem das instâncias em que todos os testes encontraram a melhor solução

Tabela 7: Comparativos dos resultados das instâncias MDG-a com $t = 900s$, $1800s$ e $3600s$

## 5. Conclusões

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 testadas, as melhores soluções encontradas na literatura. 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 e de $0,0001$ para as instâncias grandes.

Como passos futuros, instâncias maiores deverão ser testadas com o objetivo de ajustar os parâmetros usados no algoritmo visando melhorar a qualidade média das soluções e reduzir o tempo de processamento.

## Referências

Agrafiotis, D. (1997). Stochastic algorithms for maximizing molecular diversity. *Journal of Chemical Information and Modeling*, 37:841–851.

Aringhieri, R. e Cordone, R. (2011). Comparing local search metaheuristics for the maximum diversity problem. *Journal of the Operational Research Society*, 62:266–280.

Aringhieri, R., Cordone, R., e Melzani, Y. (2008). Tabu search versus GRASP for the maximum diversity problem. *4OR*, 6(1):45–60. ISSN 16194500.

Bonotto, E. L. e Cabral, L. d. A. F. (2014). Algoritmo de otimização por nuvem de partículas aplicado ao problema da diversidade máxima. *Anais XLVI SBPO, Salvador, BA*, p. 2304–2314.

Chuang, L.-Y., Hsiao, C.-J., e Yang, C.-H. (2011). Chaotic particle swarm optimization for data clustering. *Expert Systems with Applications*, 38(12):14555–14563. ISSN 09574174.

Coelho Filho, O. P. (2013). Uma nova abordagem híbrida do algoritmode otimização por enxame de partículascom busca local iteradapara o problema de clusterização de dados. *Journal of Chemical Information and Modeling*, 53(9):1689–1699. ISSN 1098-6596.

Duarte, A. e Martí, R. (2007). Tabu search and GRASP for the maximum diversity problem. *European Journal of Operational Research*, 178(1):71–84. ISSN 03772217.

Duarte, I. L., da Silva, G. C., e Costa, T. A. (2008). Algoritmos heurísticos para o problema da diversidade máxima. *Anais do XL SBPO. João Pessoa, PB*, p. 1126–1137.

Gallego, M., Duarte, A., Laguna, M., e Martí, R. (2009). Hybrid heuristics for the maximum diversity problem. *Computational Optimization and Applications*, 44(3):411–426. ISSN 09266003.

Ghosh, J. B. (1996). Computational aspects of the maximum diversity problem. *Operations Research Letters*, 19(4):175–181. ISSN 01676377.

Glover, F. e Laguna, M. (1997). Tabu search. *Kluwer Academic Publishers*.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 4:1942–1948 vol.4. ISSN 19353812.

Lozano, M., Molina, D., e García-Martínez, C. (2011). Iterated greedy for the maximum diversity problem. *European Journal of Operational Research*, 214(1):31–38. ISSN 03772217.

Palubeckis, G. (2007). Iterated tabu search for the maximum diversity problem. *Applied Mathematics and Computation*, 189(1):371–383. ISSN 00963003.

Silva, C. G., Ochi, L. S., e Martins, L. S. (2005). Proposta e avaliaç ao de heurísticas grasp para o problema da diversidade máxima. *Universidade Federal Fluminense (UFF)*, p. 321–360.

Silva, G., Ochi, L., e Martins, S. (2004). Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. *Lecture Notes on Computer Science*, p. 498–512.

Silva, G. C., De Andrade, M. R. Q., Ochi, L. S., Martins, S. L., e Plastino, A. (2007). New heuristics for the maximum diversity problem. *Journal of Heuristics*, 13(4):315–336. ISSN 13811231.

Takane, B. (2011). Um algoritmo exato para o problema da diversidade máxima. Dissertação de m.sc., UFMG, Belo Horizonte, MG, Brasil.

Wang, J., Zhou, Y., Cai, Y., e Yin, J. (2012). Learnable tabu search guided by estimation of distribution for maximum diversity problems. *Soft Computing*, 16(4):711–728. ISSN 14327643.

Wang, Y., Hao, J.-K., Glover, F., e Lü, Z. (2014). A tabu search based memetic algorithm for the maximum diversity problem. *Engineering Applications of Artificial Intelligence*, 27:103–114.