# A hybrid algorithm based on Particle Swarm Optimization and Tabu Search for the Maximum Diversity Problem

Edison L. Bonotto[1], Lucídio dos A. F. Cabral[2]

[1] Universidade Federal da Paraíba (UFPB)
R. dos Escoteiros, s/n - Mangabeira, João Pessoa-PB, 58055-000 - Brazil
bonottor@ci.ufpb.br

[2] Universidade Federal da Paraíba (UFPB)
R. dos Escoteiros, s/n - Mangabeira, João Pessoa-PB, 58055-000 - Brazil
lucidiocabral@gmail.com

**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 PSO approach simulates the behavior of a flock of birds in flight with its random movement locally, but also globally determined, to find the local maximum. The use of PSO_TS achieves the great majority of the best results for known instances testing according to literature and improved the known solution in six cases, thus demonstrating to be competitive with the best algorithms in terms of quality of the solutions.

## 1 Introduction

The *maximum diversity problem* (MDP) is a combinatorial optimization problem that can be defined as follows. Given a set $N = \{1, \ldots, n\}$, a diversity measure (e.g., Euclidean distance) $d_{ij}, i, j \in N$, and an integer $m < |N|$, the objective is to determine a subset $M \subset N$, with $|M| = m$, so as to maximize its diversity. This problem is $\mathcal{NP}$-hard [12] and, as pointed out in [22], it is also known in the literature under different names such as *maximum dispersion*, *MAX-AVG dispersion*, *edge-weighted clique*, *remote-clique*, *maximum edge-weighted sub- graph* and *dense k-subgraph*. MDP applications can be found in the pharmaceutical industry [1], in computational biology [19] and, according to the Martì [14], in the location of facilities, in the protection of biological diversity, the formulation of admission policies, the formation of committees, the composition of medical crews, and so on.

A number of metaheuristic algorithms were proposed to solve the MDP. Among them we can cite: greedy randomized adaptive search procedure (GRASP) [10, 16, 18], tabu search (TS) [2, 3, 7, 15, 20], iterated local search (ILS) [8], iterated greedy (IG) [13], scatter search (SS) [9], variable neighborhood search (VNS) [2], [4], simulated annealing (SA) [1] and memetic algorithm (MA) [21]. The reader is referred to [14] for a detailed comparison between various heuristics and metaheuristics for the problem.

This paper presents a hybrid metaheuristic algorithm that combines particle swarm optimization (PSO) with tabu search (TS) to solve the MDP. Extensive computational experiments were carried out in benchmark instances from the literature and the results obtained show that the proposed metaheuristic is capable of finding highly competitive results.

## 2 PSO_TS for MDP

Our hybrid metaheuristic algorithm (PSO_TS) combines a version of the meta-heuristic particle swarm optimization and tabu search to refine the solutions. First, we create $q$ particles of size $m$, where $Nr\_PG\%$ particles - being at least one particle - are created by a greedy algorithm and, in the remaining particles, the elements are chosen randomly. The greedy heuristics are described in *Section 2.1* and the choice of the value of the constant $q$ is shown in *Section 2.3*.

After the creation of the parts, they are ordered according to their maximum diversity and the $Nr\_Ch\_TS\_In$ best particles will serve as starting solutions for a Tabu Search algorithm. If there is improvement in maximum diversity, the returned TS solution replaced the particle and $GBest$ and $PBest$ are updated.

At each iteration $k$ of PSO, the particle moves through the multidimensional space according to *Equation* (1).

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \tag{1}$$

As presente above, $X_{id}^{k}$ is the position of the element $i$ in the dimension $d$ of the particle in the multidimensional space in the $k$-th iteration and $V_{id}^{k}$ denotes the velocity of displacement of the element $i$ in the dimension $d$ of the particle in the $k$-th iteration, and is calculated by *Equation* (2),

$$V_{id}^{k+1} = w * V_{id}^{k} + c_1 * Cr * (P_{id}^{k} - X_{id}^{k}) + c_2 * (1 - Cr) * (P_{gd}^{k} - X_{id}^{k}) \tag{2}$$

where $w$ is a variable that defines the inertial weight that will be applied to the velocity of the particle and the value is chosen through equation $w = 0.9 - ((0.9 - 0.1)/k) * q$. This equation, based on Coelho Filho [6], causes the value of $w$ - between $[0.1, ..., 0.9]$ - to be close to the maximum in the initial iterations and close to the minimum in the final iterations. This causes the particle to move faster at the beginning of processing and to "escape" from optimal locations;

$Cr$ is chosen through the equation $Cr = y * Cr * (1 - Cr)$, described by Chuang [5], instead of $r_1$ and $r_2$ of the original equation of the PSO. Initially, $Cr$ is randomly chosen from $[0, ..., 1]$. The variable $y = 4$ was used, which causes the equation to have a chaotic behavior but generates most numbers in the vicinity of 0 or 1;

**Algorithm 1:** PSO_TS

```
   input  : a $M_{nxn}$ matrix $(d_{ij})$ a given cardinality $m \leq n$
   output: the best solution found $Gb*$
1  $P \leftarrow \{x^1, ..., x^{max(1,int(m*Nr\_PG/100))}\} \leftarrow$ create_greedy()/*Section 2.1*/;
2  $P \leftarrow \{x^{max(1,int(m*Nr\_PG/100))+1}, ..., x^q\} \leftarrow$ create_random();
3  $Pb^* \leftarrow \{x^1, ..., x^q\} \leftarrow P = \{x^1, ..., x^q\}$;
4  $Gb^* \leftarrow argmax\{f(x^i)|i = \{1, ..., q\}$;
5  while $k < Kmax$ do
6      $Nr\_Ch = Nr\_Ch\_TS$;
7      if $k = 0$ then
8          $Nr\_Ch = Nr\_Ch\_TS\_In \times m/100$;
9      end
10     order $P = \{x^1, ..., x^q\}$ (descending by maximum diversity);
11     for $i \leftarrow 0$ to Nr_Ch do
12         $P(i) = TS(P(i))$   /* Section 2.2 */;
13         update $Gb^*$ e $Pb^*(s)$;
14         if time $> t$ then
15             return $Gb^*$
16         end
17     end
18     for $i \leftarrow 0$ to $q - 1$ do
19         move $P[i]$   /* Eq. (1) */;
20         update $Gb^*$ e $Pb^*(i)$;
21     end
22 end
23 return $Gb^*$.
```

$c_1$ and $c_2$ are constants that define the velocity towards the $PBest$ and $GBest$ (cognitive factor and social factor), respectively.

$X_{id}^k$ is the current position of the particle in the *k*-th iteration;

$P_{id}^k$ ($PBest$) is the best position ever visited by the particle up to the *k*-th iteration and $P_{gd}^k$ ($GBest$) is the best position ever visited by either of the particles until the *k*-th iteration.

Eventually, the new position of the $i$ element of the particle occurs outside the vector space. In this case, the element is not moved. There are also situations in which the new position of the element coincides with the position of another element of the same particle. it happens the same with this element, it is not moved.

At each iteration the particles move to the new position and the maximum diversity is calculated. If there is improvement in diversity locally (highest maximum diversity already reached by the particle), $PBest$ is updated. Also, if there is improvement in $GBest$, it is updated.

At the end of each iteration, the particles are sorted according to their diversity. A tabu search is initialized $Nr\_Ch\_TS$ times, having the $Nr\_Ch\_TS$ best particles of the round as a starting solution. The tabu search procedure is described in details in *Section 2.2*. At the end of TS, $GBest$ and $PBest$ are updated.

Finally, the algorithm stops when it reaches a number of $Kmax$ of iterations or reaches the maximum processing time $t$. The last $GBest$ found is the solution to the problem. The pseudocode of the PSO_TS algorithm is shown in *Algorithm 1*.

## 2.1 Constructive Algorithms

Two construction methods were used to create the initial particles: a greedy method and a random method.

The greedy heuristic creates a particle (valid solution) according to the following steps:

(1) randomly chooses an element of the $N$ instance to be part of the $U$ solution;

(2) creates a vector $S$ containing elements that are not yet part of $U$;

(3) stores in $S$ the value that each element $N/U$ would contribute by being part of the partial solution, according to equation $S_i = \sum_{j \in \cup} d_{ij}$;

(4) adds the first element found with the largest contribution in the vector $S$ to the partial solution $U$ and excludes this element from the vector;

(5) The vector $S$ is updated to the equation step *(3)*, as well as the maximum partial solution diversity;

(6) returns to step *(4)* until a valid solution with $m$ elements is generated.

(7) The particle itself is defined as $PBest$. The choice of the first element being random makes the algorithm non-deterministic and allows the generation of more than one particle with different elements.

Nevertheless, in the random heuristics, obviously, all the particles are generated in a random way. After all the particles are created, the $GBest$ is set. .

## 2.2 Tabu Search (TS)

In the tabu search procedure the technique described by Wang [21] was applied, which proved efficient in improving the quality of a solution. The TS consists in a constrained swap operator to exchange a variable having the value of 0. More formally, given a feasible solution $x = \{x_1, ..., x_n\}$, let $U$ and $Z$ respectively denote the set of variables with the value of 1 and 0 in $x$. Given solution $x$, the neighborhood $N(x)$ of $x$ consists of all the solutions obtained by swapping two variables $x_i \in \cup$ and $x_j \in Z$.

The Tabu Search uses a mechanism that rapidly determines the gain of the exchange movement, similar to the used by Aringhieri [3] where a vector $\triangle$ is employed to record the objective variation of moving a variable $x_i$ from its current subset $U/Z$ into the other subset $Z/U$. This vector can be initialized according to *Equation* (3) and the move gain of interchanging two variables $x_i \in U$ and $x_j \in Z$ can be calculated using the formula $\delta_{ij} = \triangle_i + \triangle_j - d_{ij}$.

$$\Delta_i = \begin{cases} \sum_{j \in U} -d_{ij}(x_i \in U) \\ \sum_{j \in U} d_{ij}(x_i \in Z) \end{cases} \tag{3}$$

Once a move is performed, we just need to update a subset of move gains affected by the move. Specifically, the following abbreviated calculation can be performed to update $\triangle$ upon swapping variables $x_i$ and $x_j$ according to *Equation* (4).

$$\triangle_k = \begin{cases} -\triangle_i + d_{ij}(\text{k=i}) \\ -\triangle_j + d_{ij}(\text{k=j}) \\ \triangle_k + d_{ik} + d_{jk}(\text{k \# \{ i,j\}},x_k \in \text{U}) \\ \triangle_k - d_{ik} + d_{jk}(\text{k \# \{ i,j\}},x_k \in \text{Z}) \end{cases} \tag{4}$$

To restrict the analysis for the motions that produce high quality results for each specific operation, the successive filter candidate list strategy of Glover [11] was employed. For the swap move, it is necessary to move the variable $x_i$ from $U$ to $Z$ and move the variable $x_j$ from $Z$ to $U$. Therefore, pick for each component operation the top *cls* variables (*cls* is a parameter and called the candidate list size) in terms of their $\triangle$ values recorded in a non-increasing order to construct the candidate lists $UCL$ and $ZCL$. For the swap moves, only the variables of $UCL$ and $ZCL$ are involved. The candidate list size of each component operation was defined $cls = min(\sqrt{m}, \sqrt{n-m})$.

The Tabu Search incorporates a short-term memory, known as the Tabu List of Glover [11]. Each time two variables $x_i$ and $x_j$ are swapped, two random integers are taken from an interval $tt = [15, 25]$ to prevent any move involving either $x_i$ or $x_j$ from being selected for a $tt$ number of iterations.

To accompany this rule, a simple aspiration criterion is applied, that permits a move to be selected in spite of being tabu if it leads to a solution better than the best solution found so far. The tabu search procedure terminates when the best solution cannot be improved within $6m$ number of iterations.

### 2.3 Parameters

For the execution of the algorithm, some constants must be previously defined. Values were tested with *q = .15, .20, .25, .5* and *1*, *Nr_PG = 1%, 10%, 40%, 70%, 80%* and *90%*, $c_1$ *=.8, 1* and *1.2*, $c_2$ *= 1.2, 1.4* and *1.8*, *Nr_Ch_TS= 4, 5, 6, 7, 8* and *9* and *Nr_Ch_TS_In= 0%, 20%, 40%* and *60%*.

The parameters were chosen in an empirical way, based on the results of the tests performed, analyzing the best solutions and average solutions found. *Table 1.* shows the values of the constants, where $q \times m$ = initial particle number, $Nr\_PG$ = percentage of particles created in a greedy manner, $Kmax \times m$ = maximum number of PSO iterations, $c_1$ and $c_2$ = positive acceleration constants, $Nr\_Ch\_TS$ = number of calls to TS at each iteration of the PSO and $Nr\_Ch\_TS\_In$ = Percentage of initial particles that will be subjected to Tabu Search.

| Instances | *q* | *Nr_PG* | *Kmax* | $c_1$ | $c_2$ | $Nr\_Ch\_TS$ | $Nr\_Ch\_TS\_In$ |
|---|---|---|---|---|---|---|---|
| SOM-b | 1 | 80 % | 1 | 1.00 | 1.40 | 7 | 0 |
| MDG-a | .20 | 80 % | 1 | 1.00 | 1.40 | 8 | 0 |
| MDG-c | .20 | 60 % | 4 | 1.00 | 1.40 | 6 | 20 |

Table 1: Values defined for the constants

## 3 Computational experiments

In order to verify the efficiency of the proposed algorithm, we used the medium-sized SOM-b instances, created by [17], which contains a population of size $n$ = 100, $n$ = 200, $n$ = 300, $n$ = 400 and $n$ = 500, and tested solutions with sizes of $m = 10\%n$, $20\%n$, $30\%n$ e $40\%n$. The metaheuristic PSO_TS was executed 10 times with a maximum time of 1s. The best known solutions (*BKS*) were taken from [8]. For the large instances, were used the 20 MDG-a instances created by [7] and 30 executions were performed for a time limit of $16s$ and the 20 MDG-c instances, similar to those used by [15], and 10 executions were performed for a time limit of $480s$. For the maximum diversity, the greatest diversity (*#Best*) and the average diversity (*#Avg*) were analysed.

The algorithm was implemented using the C++ programming language, gcc compiler - compiled with the gcc -O2 option - and Linux OS Ubuntu V. 4.2.0-35. For the tests, a computer with Intel i7 950 @ 3.07GHz processor with 4 cores, 8 GB Mem, HD 2 TB Gb was used. Only 1 processor core was used.

*Table 2* shows the average results obtained for the medium-sized instances SOM-b. In this table, *column 1* indicates the instance used; *column 2*, the size of this instance; *column 3*, the amount of m elements of the particle (solution); *column 4*, the best solution (maximum diversity) known in the literature (BKS); *column 5*, the average solution found in the 10 tests and *columns 6*, 7 and 8 shows, respectively, the minimum, average and maximum processing time until you find the solution.

We can observe that in all the tested instances and in 100% of the 10 tests executed in each one of them, the algorithm PSO_TS found the best solution known in the literature, that is, a null standard deviation. The maximum time to find the *BKS* ranged from 0.00 to 0.52s, depending on the $n$ and $m$ instance value. The average time to find the best solution known in all instances in all runs was 0.08s.

| Instances | n | m | BKS | PSO_TS | | | |
|---|---|---|---|---|---|---|---|
| | | | | Average | Time (seconds) | | |
| | | | | | Min | Avg | Max |
| 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 |
| Average | | | | | 0.02 | 0.08 | 0.17 |

Table 2: Results of SOM-b instances (Time $< 1s$)

Compared with *state-of-the-art* algorithms using the SOM-b instances cited in [14], where they use a longer processing time - 10s or 20s on an Intel Core2 Quad Q8300 2.5GHz processor, the PSO_TS found better or equal average results.

In *Table 3* we have the results found for the large MDG-a instances, where the columns indicate, respectively, the instance used; the best solution known in the literature; the best solution found; the GAP (*BKS-#Best*); the GAP (*BKS-#Avg.*) and the coefficient of variation, which corresponds to the standard deviation divided by the mean, between the 30 runs.

We can observe that in only one instance the algorithm did not found the best known solution and the GAP was only 2 units and in the average of the solutions the GAP was 36.6. The average coefficient of variation between the solutions, obtained by the ratio between standard deviation and the mean, was $0.00027$, which demonstrates the robustness of the algorithm.

| Instances | BKS | PSO_TS | | | | |
|---|---|---|---|---|---|---|
| | | Solution | | GAP | | CV |
| | | #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 |
| Average | | | | 0.1 | 36.6 | 0.00027 |

Table 3: Results of MDG-a instances (Time $\leq 16s$)

The results of the MDG-a instances were compared with the state-of-the-art algorithms, that were reported in [21] where the algorithms were run for 17s on a Xeon E5440 with 2.83 GHz CPU and 8 GB RAM or equivalent machines.

To determine the stopping time of our algorithm, we used the CPU Benchmarks - Single Thread Performance [1] which indicated that our computer has a factor 1.036 faster than those used in benchmarking [21]. Thus, we set the time limit for 16s for the MDG-a instances.

Observing the graph of *Table 1*, we can see that the PSO_TS has obtained better results in all instances than the ITS [15], VNS [4], TIG [13] and LTS_EDA [20] algorithms, comparing the best solutions found and the average of the solutions. In comparison with the TS-MA [21] algorithm, we obtained slightly lower results regarding the quality of the solutions. We believe that this is due to the characteristics of the metaheiristic PSO, which requires a longer processing time to converge to the optimal solutions.

For the analysis of the MDG-c instances, also considered large instances, 10 executions were performed and a maximum processing time of $480$ seconds was used. In *Table 4* we have the results found for these instances, where the columns indicate, respectively, the instance used; the best solution known in the literature; the best solution found in the tests; the average solution found; the GAP of the best solution and the GAP of the average solution in relation to the best known solution. The GAP was calculated according to the equation $GAP = \left(\left(\frac{\#Best}{Soluc}\right) - 1\right) \times 100$, where *#Best* is the best solution found in the

[1]http://www.cpubenchmark.net/singleThread.html

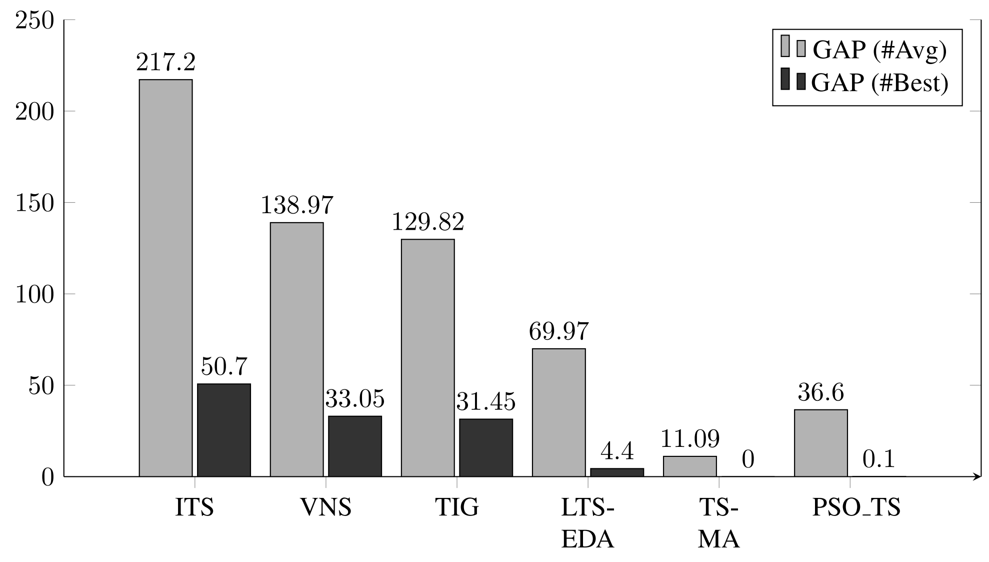

Figure 1: Comparison among PSO_TS and other *state-of-the-art* algorithms (MDG-a instances)

literature and *Soluc* is the analyzed solution. Negative GAPs represent better results. The best solutions known in the literature were taken from [14].

As we can see, the PSO_TS has found a better solution known in four instances and mainly improved the known results for six cases. In two cases, even its average quality was better than the previously known results reported in the literature.

The results obtained in the MDG-c instances were compared with state-of-the-art algorithms, which were reported in [14] and executed for a processing time of 600 seconds on an Intel Core 2 Quad CPU Q 8300 with 6 GB of RAM. To determine the stopping time of our algorithm, we used the CPU Benchmarks (Single Thread Performance) which indicated that our computer has a factor 1,246 faster than those used in the tests by [14]. Then, we determine the stop time as 480 seconds for the MDG-c instances.

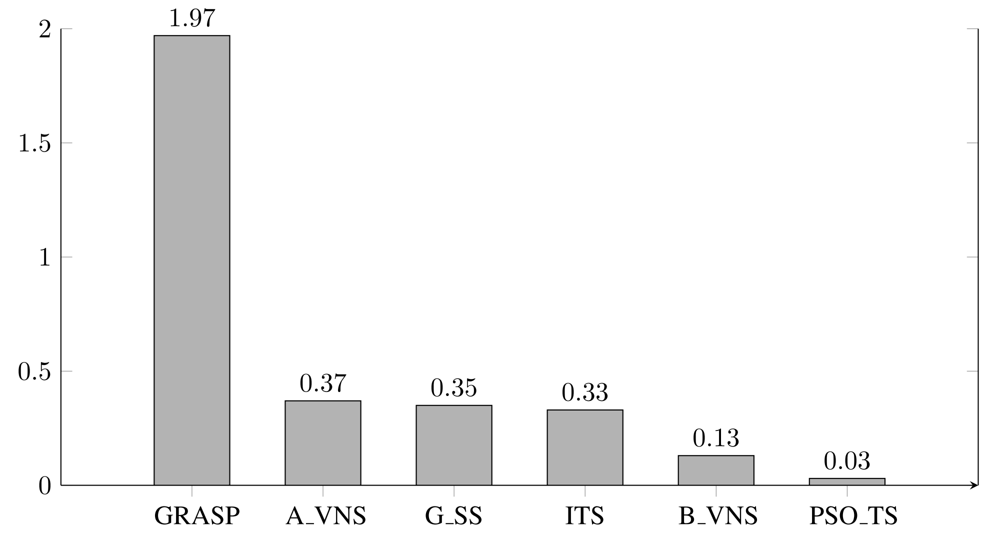

Figure 2: Comparison among PSO_TS and other *state-of-the-art* algorithms (MDG-c instances)

The graph of *Figura 2* shows the comparison between the GAP of the best solution known in the literature and the average of the solutions found in the PSO_TS with the results of the different meta-heuristics. We can observe that the PSO_TS algorithm obtained better results than all the algorithms compared: GRASP_D2+I_LS [7], ITS [15], B_VNS [4], A_VNS [2] and G_SS [9].

| Instncia | #Best | PSO_TS | | | |
|---|---|---|---|---|---|
| | | Soluo | | GAP % | |
| | | Melhor | Mdia | (1) | (2) |
| 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 |
| Average | | | | 0.017 | 0.033 |

Table 4: Results of MDG-c instances (Time $\leq 480s$)

## 4 Conclusions

In this paper, we propose a new approach to solve the *problem of maximum diversity* based on meta-heuristic *particle swarm optimization* along with *tabu search*. For the generation of the initial particles, a greedy and random construction heuristic was implemented.

The method, called PSO_TS, is simple in its implementation and the results demonstrate that the algorithm achieves excellent performance, obtaining, in most tested instances, the best solutions found in the literature and in MDG-c instances, the algorithm improved the known solution in six cases.

The PSO_TS was quite robust in relation to the average quality behavior of the obtained solutions, with a null coefficient of variation for the medium-sized instances. In large instances MDG-a, the coefficient of variation was $0,00027$ and only $0,00012$ for the MDG-c instances. In summary, a null mathematical variability.

As future steps, in order to improve the average quality of the solutions and, mainly, to reduce the processing time, we have as main suggestions a study of the adjustment of parameters used in the algorithm seeking to improve the performance, the exploration of new neighborhood structures for the particles, where the PBest is chosen from the best neighbor and the development of parallel approaches to the PSO_TS to solve the large instances.

## References

[1] D.K. Agrafiotis. Stochastic algorithms for maximizing molecular diversity. *Journal of Chemical Information and Modeling*, 37:841–851, 1997.

[2] R Aringhieri and R Cordone. Comparing local search metaheuristics for the maximum diversity problem. *Journal of the Operational Research Society*, 62:266–280, 2011.

[3] Roberto Aringhieri, Roberto Cordone, and Yari Melzani. Tabu search versus GRASP for the maximum diversity problem. *4OR*, 6(1):45–60, 2008.

[4] Jack Brimberg, Nenad Mladenović, Dragan Urošević, and Eric Ngai. Variable neighborhood search for the heaviest k-subgraph. *Computers and Operations Research*, 36(11):2885–2891, 2009.

[5] Li-Yeh Chuang, Chih-Jen Hsiao, and Cheng-Hong Yang. Chaotic particle swarm optimization for data clustering. *Expert Systems with Applications*, 38(12):14555–14563, 2011.

[6] Osires Pires Coelho Filho. 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*, 53(9):1689–1699, 2013.

[7] Abraham Duarte and Rafael Martí. Tabu search and GRASP for the maximum diversity problem. *European Journal of Operational Research*, 178(1):71–84, apr 2007.

[8] Ivan Luis Duarte, Geiza Cristina da Silva, and Tatiana Alves Costa. Algoritmos heurísticos para o problema da diversidade máxima. *Anais do XL SBPO. João Pessoa, PB*, pages 1126–1137, 2008.

[9] Micael Gallego, Abraham Duarte, Manuel Laguna, and Rafael Martí. Hybrid heuristics for the maximum diversity problem. *Computational Optimization and Applications*, 44(3):411–426, 2009.

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

[11] F. Glover and M. Laguna. Tabu search. *Kluwer Academic Publishers*, 1997.

[12] Ching-Chung Kuo, Fred Glover, and Krishna S. Dhir. Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming. *Decision Sciences*, 24(6):1171–1185, 1993.

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

[14] Rafael Martí, Micael Gallego, Abraham Duarte, and Eduardo G. Pardo. Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics*, 19(4):591–615, 2013.

[15] Gintaras Palubeckis. Iterated tabu search for the maximum diversity problem. *Applied Mathematics and Computation*, 189(1):371–383, jun 2007.

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

[17] G.C. Silva, L.S. Ochi, and S.L. Martins. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. *Lecture Notes on Computer Science*, pages 498–512, 2004.

[18] Geiza C. Silva, Marcos R Q De Andrade, Luiz S. Ochi, Simone L. Martins, and Alexandre Plastino. New heuristics for the maximum diversity problem. *Journal of Heuristics*, 13(4):315–336, 2007.

[19] Bruno Takane. Um algoritmo exato para o problema da diversidade máxima. Dissertação de m.sc., UFMG, Belo Horizonte, MG, Brasil, 2011.

[20] Jiahai Wang, Ying Zhou, Yiqiao Cai, and Jian Yin. Learnable tabu search guided by estimation of distribution for maximum diversity problems. *Soft Computing*, 16(4):711–728, 2012.

[21] Yang Wang, Jin-Kao Hao, Fred Glover, and Zhipeng Lü. A tabu search based memetic algorithm for the maximum diversity problem. *Engineering Applications of Artificial Intelligence*, 27:103–114, 2014.

[22] Qinghua Wu and Jin Kao Hao. A hybrid metaheuristic method for the Maximum Diversity Problem. *European Journal of Operational Research*, 231(2):452–464, 2013.