A Cooperative Strategy based on Brain Storm Optimization Algorithm

Ricardo García-Ródenas[1] Luis Jiménez Linares[1] Julio Alberto López Gómez[1]

Universidad de Castilla la Mancha

Abstract. In the last few years, Brain Storm Optimization(BSO) algorithm have arisen as a new metaheuristic based on population which is inspired into the human brainstorming process. This new metaheuristic tries to deal with hard optimization problems, attempting to improve the results obtained by other metaheuristic based on the behaviour of social insects like bacterias, ants, particles... After that, new versions of this algorithm have appeared in order to improve the efficiency and results of BSO, like ADMBSO (Advanced Discussion Mechanism-Based Brain Storm Optimization Algorithm)
This paper introduces a cooperative strategy of ADMBSO with Nelder-Mead (NM) algorithm in order to benefit the advantages in exploration of ADMBSO and the advantages in exploitation of NM.

Keywords: Brain Storm Optimization, Hybridization, Global Optimization, Nelder-Mead, Cooperative Strategy

1 Introduction

Metaheuristics [1],[2] are a set of optimization methods for problem solving which employ a set of strategies to overcome local optimums in search spaces. Traditionally, biological systems have been a source of inspiration which metaheuristics are based in. Thus, there appeared metaheuristics based on social insects like Ant Colony Optimization [3], Bee Colony Algorithm [4], Particle Swarm Optimization [5] or Bacterial Foraging Algorithm [6], among others.
The usefulness of this kind of optimization methods has been successfully proven. This fact has favoured the emergence of new metaheuristics which have arisen, many of them, from the previous. In 2011 Brain Storm Optimization Algorithm(BSO)[7] arises from the inspiration in the human brainstorming process. Brainstorming is a method of ideas generation which people use to deal with complex problems. Based on this metaphor, Brain Storm Optimization algorithm tries to address with complex optimization problems in order to improve the results obtained by the current metaheuristics.
The appearance of BSO involves a paradigm shift in the building of metaheuristics, since traditionally, these were based on biological systems. Now, this state has changed in order to obtain new metaheuristics which theoretically, must be better than the existing ones due to their inspiration in the most inteligent animal: human. In the last few years, this algorithm has been subject to different

modifications in order to improve its results. Thus, ADMBSO(Advanced Discussion Mechanism-Based Brain Storm Optimization Algorithm)[8] appears, which is considered like the reference implementation of Brain Storm Optimization Algorithm.
However, metaheuristics does not use the gradient or Hessian matrix, which can accelerate the local search and improve the current results. In addition, metaheuristics need a set of parameters to be fitted to the problem at hand. Furthermore, although these kind of algorithms has very good capacities of exploration, they are worst in exploitation tasks than exact algorithms which use the gradient, derivates or other mathematical methods to do that. This paper proposes a cooperative strategy for ADMBSO, combining it with Neld-Mead algorithm in order to improve the individual capabilities of these methods.
The rest of the document is structured as follows: In section 2, an overview of brain storm optimization and ADMBSO algorithms is presented. In section 3, the proposal is introduced and described in detail. Section 4 will expose the experiments and results to validate the proposal. Finally, conclusions and further works are given in section 5.

2 Brain Storm Optimization Algorithm

Brainstorming process has been employed in companies and enterprises in order to solve different kind of problems related to business processes. Due to the success of this method, Shi [7] proposed in 2011 a metaheuristic based on this process. This new metaheuristic adopts the idea of brainstorming process like it was an optimization problem in which new ideas are generated from others in order to obtain the best solution of a given optimization problem. Thus, Brain Storm Optimization (BSO) has the special feature of being based on a human process unlike the rest metaheuristics based on population which take some physical or biological aspects belonging to different animal species, like ants, bees or even bacterias.
Moreover, a brainstorming process will consist on the successive ideas generation, until an optimum idea for the problem is reached. For that purpose, best ideas will be kept and the worst will be discarded. The algorithm works as follows:
The Brain Storm Optimization algorithm will depart from a set of random possible solutions (population), which they will be ranked and grouped according to a clustering algorithm -initially k-means - which it is know as *Converging Operation*, since the solutions converge into k clusters. After that, and following a probabilistic scheme, new individuals will be generated departing from the individuals of one cluster or individuals for more than one cluster, which is called, respectively, intra and inter cluster generation, since the idea generation on a brainstorming process is done from one or more ideas, belonging all of them to only one problem owner (cluster) or more. The clusters selection in order to generate new individuals, and the concrete individuals selection (if they are centroids or general individuals) will be done following a set of probabilities which

determine, in each iteration of the algorithm, how the individuals generation will be produced. If the generation of a new individual is departing from one individual of a cluster, whether this is a centroid or a regular individual, the generation of a new individual x_{new} is done through equation 1.

$$x^i_{new} = x^i_{old} + \epsilon(t) * random(t) \tag{1}$$

Where:

- x^i_{new}**:** Is the ith dimension of x_{new} (the new element to be generated)
- x^i_{old}**:** Is the ith dimension of x_{old} (the current element from which the new element will be generated)
- ϵ**:** It is a coefficient that weights the contribution of random value to the new individual. It is defined by equation 2.

$$\epsilon(t) = logsig\left(\frac{\frac{T}{2} - t}{k}\right) \tag{2}$$

where:
  - $logsig$**:** It is a logarithmic sigmoid transfer function.
  - T**:** It is the maximum number of iterations.
  - t**:** It is the current iteration
  - k**:** It is for changing the slope of $logsig()$ function.

  Thus, large ϵ values will facilitate the exploration while small ϵ will facilitate exploitation. In this manner, when global search capability is preferred, which is normal at the beginning of the algorithm, ϵ values will be greater. Meanwhile, at the end of the algorithm, when exploitation around a local environment is preferred, ϵ values will be lower.

For another part, when a new individual x_{new} is generated from more than one individuals, for example, x_{old1} and x_{old2}, the equations which regulate the individuals generation can be seen in equations 3 and 4.

$$x^i_{new} = x^i_{old} + \epsilon(t) * random(t) \tag{3}$$

$$x^i_{old} = w_1 * x^i_{old1} + w_2 * x^i_{old2} \tag{4}$$

Where:

- x^i_{old1}, x^i_{old2}**:** They are the ith dimension of the two ideas used to create x_{new}.
- w_1, w_2**:** They are two coefficients which weight the contribution of the two individuals used to create x_{new}.
- x^i_{old}**:** Represents the summation of the ith dimension of the individuals used to create the new individual and which will be used in equation 3 to calculate the ith dimension of the generated individual.

In this way, the algorithm is capable of generating new individuals which are generated by piggybacking from more than one individual. These operations are known as *Diverging Operation*, where individuals who were grouped in different clusters diverge from their neighbours through the added noise to them. However, in order to reduce the complexity of the algorithm, the original version of BSO only allow it to build new ideas departing from a maximum of two ideas.

Moreover, few years later, Shi [8] proposed a new metaheuristic based on BSO called ADMBSO. This algorithm removes the mutation of evolutionary algorithms, in order to improve the results of BSO, and incorporate a way of controlling the effect of the probabilities in order to control the exploration and exploitation, encouraging the exploration in the first iterations and exploitation in the last iterations. To do that, the intra and inter cluster probabilities generation are set as shown below in equations 5 and 6.

$$P_{intra} = P_{low} + P_{high}\frac{N_{cur}gen}{N_{max}gen} \tag{5}$$

Where:

- $N_{cur}gen$: is the current generation.
- $N_{max}gen$: is the maximum number generation.
- P_{low} and P_{high} are set apriori.

$$P_{inter} = 1 - P_{intra} \tag{6}$$

Furthermore, ADMBSO algorithm incorporates a new way of ideas generation both in intra and inter cluster generation.

- **Intra-Cluster Generation:** On intra-cluster generation, ADMBSO allows to build a new idea combining two ideas from the selected cluster.
- **Inter-Cluster Generation:** On inter-cluster generation, ADMBSO includes the possibility of building a new idea randomly depart from the current idea.

In order to compare the performance of BSO and ADMBSO algorithms, table 1 shows the results for two hard examples: Sphere and Rastrigin functions with 30 dimensions. These experiments have been executed with 50 simulations and 2000 iterations per simulation. It involves 100.000 function evaluations, which means a large simulation. Furthermore, figure 1 shows graphically the performance of these two algorithms. To sum up, the pseudocodes for BSO and ADMBSO algorithms are shown in algorithms 1 and 2.

Function	**Dimension**	**Algorithm**	**Mean**	**Best**	**Worst**	**Variance**
Sphere Function	30	**BSO**	3.412e-45	0	64.872	4.687e-44
Sphere Function	30	**ADMBSO**	0	0	116.993	0
Rastrigin Function	30	**BSO**	37.363	25.869	317.95	816.91
Rastrigin Function	30	**ADMBSO**	2.122	0	334.48	267.64

Table 1. Comparative between BSO and ADMBSO results

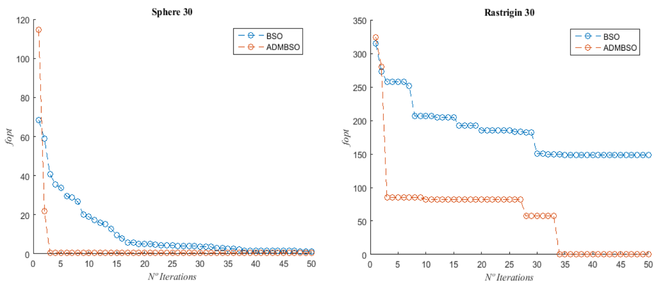

Fig. 1. Results of BSO and ADMBSO for Sphere and Rastrigin functions with 30 dimensions.

3 A Cooperative Strategy based on ADMBSO algorithm

As many authors have pointed along the state of art of metaheuristics, they encourages a lot the exploration capabilities in order to overcome local optimums in complex functions. However, the metaheuristic's exploitation cabilities are not as good as exploitation cabilities of exact methods, which employs the gradient and Hessian matrix concepts in order to obtain a global optimum.
For that reason, it is possible to consider whether these different methods can cooperate in any way in order to improve the individual performance of these methods. In order to do that, cooperative strategies [11] arise as a concept derived from agent-based computation where different agents execute the same or different methods successively or simultaneosly, depending of it is a sequential or a parallell strategy. Furthermore, a coordinator agent who receives the solutions in each iteration of each agent evaluates the solutions and take actions in order to improve the results of each agent.
In this work, two cooperative strategies have been developed in order to compare the different criteria used. These two cooperative strategies are based on:

- **Successive Improvements:** In this first strategy, the coordinator has a rule related to the number of successive improvements of each agent. When an agent a_i sends to the coordinator the best solution in n_c consecutive iterations, it forwards the solution to the rest of agents, in order to they redirect their search departing from the best solution. This rule is shown below.

```
if BSO_sol < NM_sol n_c times
    Coordinator sends BSO_sol to NM
elseif NM_sol < BSO_sol n_c times
    Coordinator sends NM_sol to BSO
```

Algorithm 1 BSO Algorithm

```
 1: BEGIN BSO
 2: while $n_{iter} \neq max_{iter}$ do
 3:     /* Update $\epsilon$ */
 4:     $\epsilon(t) = logsig\left(\frac{\frac{max_{iter}}{2} - n_{iter}}{k}\right) * random(t)$
 5:     BEGIN Converging Operation
 6:     Cluster population in $k$ clusters
 7:     END Converging Operation
 8:     Centroid Mutation
 9:     /* Generate new individuals (ideas) */
10:     for $i$ in population do
11:         BEGIN Diverging Operation
12:         if $rand() < p_{intra}$ then
13:             /* Generate individuals from one cluster*/
14:             $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
15:         else
16:             /*Generate individuals from two clusters*/
17:             $x^i_{old} = w_1 * x^i_{old1} + w_2 * x^i_{old2}$
18:             $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
19:         end if
20:         END Diverging Operation
21:         /* Update Solution */
22:         if $f(x_{new}) < f(x_{old})$ then
23:             $x_{old} = x_{new}$
24:         end if
25:     end for
26: end while
27: END BSO
```

– **Fuzzy Rules:** In this case, the coordinator has a fuzzy rule base in order to evaluate the solutions produced by each agent. To do that, the solutions provided by each agent will be evaluated as well as the ratio of improvement. Thus, two fuzzy sets are proposed in order to define a bad solution and a bad improvement ratio. The decision making process is made through the use of a minimum membership value to these concepts (given a_0 as the minimum value of the distribution and a_2 the maximum). The fuzzy bad solution and bad ratio improvement sets are defined in equations 7 and 8 while the fuzzy rule is shown below.

```
if BSO_sol is bad & Ratio_Improvement_BSO is bad
    Coordinator sends NM_sol to BSO
elseif NM_sol is bad & Ratio_Improvement_BSO is bad
    Coordinator sends BSO_sol to NM
```

Algorithm 2 ADMBSO Algorithm

```
1: BEGIN ADMBSO
2: while $n_{iter} \neq max_{iter}$ do
3:   $P_{intra} = P_{low} + P_{high} \frac{N_{curgen}}{N_{maxgen}}$
4:   $P_{inter} = 1 - P_{inter}$
5:   /* Update $\epsilon$ */
6:   $\epsilon(t) = logsig\left(\frac{\frac{max_{iter}}{2} - n_{iter}}{k}\right)$
7:   BEGIN Converging Operation
8:   Cluster population in $k$ clusters
9:   END Converging Operation
10:  for $i$ in population do
11:    BEGIN Diverging Operation
12:    if $rand() < p_{intra}$ then
13:      /* Generate individuals from one cluster*/
14:      if $rand() < p_{centroid}$ then
15:        /* Add step to center of the cluster */
16:        $c^i_{new} = c^i_{old} + \epsilon(t) * random(t)$
17:      else if $rand() < p_{individuals}$ then
18:        /*Combine two ideas from this cluster and add with step */
19:        $x^i_{old} = w_1 * x^i_{old1} + w_2 * x^i_{old2}$
20:        $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
21:      else
22:        /*Add step to the idea*/
23:        $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
24:      end if
25:    else
26:      /*Generate individuals from two clusters*/
27:      if $rand() < p_{rnd}$ then
28:        /*Generate randomly an idea depart from two individuals of two clus-
           ters/*
29:        Select $x_{old1}$ and $x_{old2}$
30:        $x^i_{old} = w_1 * x^i_{old1} + w_2 * x^i_{old2}$
31:        $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
32:      else if $rand() < p_{cens}$ then
33:        /*Generate randomly an idea depart from two individuals of two clus-
           ters/*
34:        Select $x_{old1}$ as the current idea and $x_{old2}$
35:        $x^i_{old} = w_1 * x^i_{old1} + w_2 * x^i_{old2}$
36:        $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
37:      else
38:        /*Select two clusters $c_{old1}$, $c_{old2}$, combine the centers and add with step/*
39:        $x^i_{old} = w_1 * c^i_{old1} + w_2 * c^i_{old2}$
40:        $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
41:      end if
42:    end if
43:    END Diverging Operation
44:    /* Update Solution */
45:    if $f(x_{new}) < f(x_{old})$ then
46:      $x_{old} = x_{new}$
47:    end if
48:  end for
49: end while
50: END ADMBSO
```

$$\mu_{BadSol}(x) = \begin{cases} 1 & si \quad x \leq a_0 \\ \frac{a_2 - x}{a_2 - a_1} & si \; a_1 < x < a_2 \\ 0 & si \quad x \geq a_2 \end{cases} \tag{7}$$

$$\mu_{Badir}(ratio_i) = \begin{cases} 1 & si \quad x \leq a_0 \\ \frac{a_2 - ratio_i}{a_2 - a_1} & si \; a_1 < x < a_2 \\ 0 & si \quad x \geq a_2 \end{cases} \tag{8}$$

4 Experiments and Results

Once the strategies have been defined in the previous section, the experiments and results are shown in this section. In order to demonstrate the performance of the cooperative methods against the individual capabilities of ADMBSO, Nelder-Mead (NM) algorithm has been used to cooperate with ADMBSO. This is so, because Nelder-Mead is an heuristic method which has a quikly convergence to local minimums and non-stationary points whose efficiency has been proven [9],[10]. Furthermore, this algorithm have been widely used in nonlinear optimization problems for which derivates may not be known.
In this manner, it has two methods with different features: on the one hand, ADMBSO algorithm, like other metaheuristics, encourages exploration in order to overcome local optimums, although it does not have very strong exploitation capabilities. On the other hand, NM algorithm encourages a lot exploitation like other exact methods, but it does not provide good enough exploration skills. It motivates the development of a cooperative strategy in order to hybridize these methods to improve their performance.
The experiments done in this section use the benchmark functions used in [7] in order to study the pros and cons of this approach. In this case, it has been defined an experiment with a smaller function evaluations number in order to reduce the complexity of the experiment. In this case, ten simulations with 4000 iterations have been done, which involves a total of 40.000 function evaluations. Thus, the current experiments done in this paper reproduce the simulations made by Shi [7], [8] but applying the cooperative approach developed here. The number of agents executed by the cooperative strategy is 2, one of them executing ADMBSO algorithm and the other one executing NM, which is the simplest strategy that it can be executed. This strategy has been developed sequentially and the parameters have been fixed according to tables 2 and 3 for successive improvements and fuzzy logic respectively. In order to compare these three approaches, figure 2 shows graphically a comparative about them.

Function	n_{agents}	$n_{simulations}$	$n_{iterations}$	**n**	**d**	n_c
Sphere Function	2	10	4000	100	30	3
Rastrigin Function	2	10	4000	100	30	3

Table 2. Parameters for successive improvements strategy

Function	n_{agents}	$n_{simulations}$	$n_{iterations}$	**n**	**d**	α
Sphere Function	2	10	4000	100	30	0.7
Rastrigin Function	2	10	4000	100	30	0.7

Table 3. Parameters for Fuzzy Logic Strategy

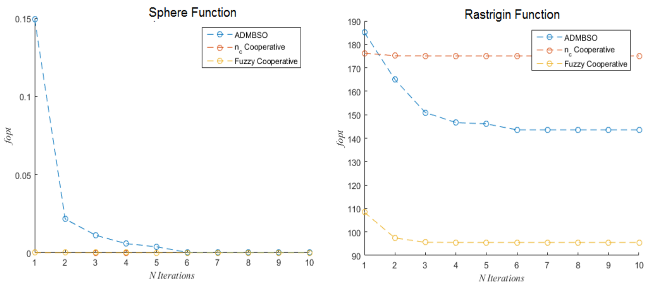

Fig. 2. Comprative between sucessive improvements and fuzzy logic strategies.

5 Conclusions and Further Works

In this paper, two cooperatives strategies for ADMBSO algorithm have been proposed in order to advantage the individual capabilities of these two algorithms. As it was seen, the results obtained improves the individual results of these algorithms. Concretely, it has been proven that the use of fuzzy logic rules improve significantly the performance of these methods.

Function	**Dimension**	**Algorithm**	**Mean**	**Best**	**Worst**	**Variance**
Sphere Function	30	**ADMBSO**	0.0192	0.0001	0.143	0.0021
Sphere Function	30	**Cooperative nc**	4.65e-08	5.12e-21	4.5e-07	2.01e-14
Sphere Function	30	**Cooperative Fuzzy**	5.38e-06	2.27e-15	5.37e-05	2.88e-10
Rastrigin Function	30	**ADMBSO**	142.608	140.383	153.84	17.92
Rastrigin Function	30	**Cooperative nc**	184,233	183.071	193.182	10.108
Rastrigin Function	30	**Cooperative Fuzzy**	101.249	97.364	109.746	10.214

Table 4. Comparative between ADMBSO and Cooperative Strategies Results

However, there is still a lot of work in order to improve the results of ADMBSO, trying with different fuzzy clustering algorithms, parallelizing the developed strategies or fitting the parameters of the presented metaheuristic and the cooperative strategies proposed here, fitting their parameters, increasing the number of agents and the algorithms which cooperate...

Acknowledgment

The authors would like to express his thanks to the project *Desarrollo, implementación y validación de una metodología para la estimación de matrices origen-destino a partir de volúmenes de tráfico y matrices desactualizadas* and to the authorities that give their support to its development, the FEDER and the *Junta de Comunidades de Castilla la Mancha*.

References

1. Voss, S., Martello, S., Osman, I., Roucairol, C.: Meta-Heuristics Advances and Trends in Local Search Paradigms for Optimization. 1998.
2. Gendreau, M., Potvin J.: Handbook of Metaheuristics. Springer. 2010.
3. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst., Man, Cybern. B. 26(2), 29,41 (1996)
4. Karaboga D., Akay B.: A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation. 2009.
5. Shi,Y., Eberhart, R.C.: A modified Particle Swarm Optimizer. In: 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, May 4-9 (1998)
6. Passino, K.M.:Bacterial Foraging Optimization. International Journal of Swarm Intelligence Research 1(1), 1-16 (2010)
7. Shi, Y: Brain Storm Optimization Algorithm. Advances in Swarm Intelligence. Springer. 2011
8. Yang, Y., Shi, Y., Shunren X.: Advanced discussion mechanism–based brain storm optimization algorithm. Soft Computing. 2014.
9. Angulo E., Castillo E., García-Ródenas R., Sánchez-Vizcaíno J.: Determining highway corridors. Journal of Transportation Engineering 2011;137(5): 557–70.
10. Panigrahi B., Pandi V: Bacterial foraging optimization: Nelder-Mead hybrid algorithm for economic load dispatch. 2008
11. Masegosa A, Pelta D, Verdegay J.: A centralised cooperative strategy for continuous optimisation: The influence of cooperation in performance and behaviour.