Artificial Bee Clustering Search

Tarcísio Souza Costa and Alexandre César Muniz de Oliveira

Universidade Federal do Maranhão, UFMA - Departamento de Informática,
65.085-580 – Campus do Bacanga, São Luís - MA, Brasil
priestcp@gmail.com, acmo@deinf.ufma.br

Abstract. Clustering Search (*CS) has been proposed as a generic way of combining search metaheuristics with clustering to detect promising search areas before applying local search procedures. The clustering process may keep representative solutions associated to different search subspaces (search areas). In this work, a new approach is proposed, based on Artificial Bee Colony (ABC), observing the inherent characteristics of detecting promissing food sources employed by that metaheuristic. The proposed hybrid algorithm, performing a Hooke & Jeeves based local, is compared against other versions of ABC: a pure ABC and another hybrid ABC, exploring an elitist criteria.

1 Introduction

Modern metaheuristics attempt to balance exploration and exploitation movements explicitly by employing global and local search strategies. The global algorithm plays the role of generator of reference points to diversified search areas, which are more intensively inspected by problem-specific components further. Early hybrid algorithms have demonstrated concern about rationally employing local search. Instead of always applying local search procedures over entire population, the global algorithm runs normally until a promising area is detected, according to a specific criteria, for example, when a new best individual is found (elitist criteria). In a general way, the following action must be better to exploit the expected promising area, in a reduced search domain, by a local search procedure.

Clustering Search (*CS) has been proposed as a generic way of combining search metaheuristics with clustering, aiming to detect promising search areas before applying local search procedures [4]. Its generalized nature is achieved both by the possibility of employing any metaheuristic and by also applying to combinatorial and continuous optimisation problems. Clusters of mutually close solutions provide reference points to relevant areas of attraction in the most of search metaheuristics. Relevant search areas can be treated by problem-specific local search procedures. Furthermore, the cluster center itself is always updated by a procedure involving inner solutions, called assimilation [4].

Artificial Bee Colony (ABC) is a recently proposed optimisation algorithm [1] that simulates the intelligent behavior of honey bees, providing a population-based search procedure in which individuals called foods positions are modified

I. Rojas, G. Joya, and J. Cabestany (Eds.): IWANN 2013, Part II, LNCS 7903, pp. 20–27, 2013.

by the artificial bees, aiming to discover promising food sources and, finally, the one with the optimal amount of nectar [1].

This paper is devoted to propose a new *CS approach, called Artificial Bee Clustering Search (ABCS), applied to unconstrained continuous optimisation, in which promising search areas are associated to promising food sources and the bee movement is performed by the intrinsic *CS operation known as assimilation [4]. The remainder of this paper is organized as follows. Section 2 describes the concepts behind *CS and ABC. ABCS is described in section 3 and computational results are presented in 4. The findings and conclusions are summarized in section 5.

2 Algorithms Foundations

A challenge in hybrid metaheuristic is to employ efficient strategies to cover all the search space, applying local search only in actually promising search areas. The inspiration in nature has been pursued to design flexible, coherent and efficient computational models. In this section, the Clustering Search (*CS) and ABC are briefly described, introducing concepts needed to further explain how to combine them to allow detecting promising search areas before applying the Hooke & Jeeves local search procedure [3].

2.1 Clustering Search

*CS attempts to locate promising search areas by framing them by clusters. A cluster is defined by a *center*, c, that is generally, initialized at random and, posteriorly, it tends to progressively slip along really promising points in the search space. The number of clusters $\mathcal{NC}$ can be fixed a priori [5] or dynamically determined according to the width of the search areas being explorated by the metaheuristic [4]. In the later case, clusters can be created in a way that all candidate solutions are covered by, at least, a cluster. By the other hand, inactive clusters, i.e., clusters not covering any solutions may be eliminated.

The coverage is determined by a *distance metric* that computes the similarity between a given solution and the cluster center and must consider the problem nature. For example, in unconstrained continuous optimization, the similarity has been defined regarding the Euclidean distance [4]. *CS can be splitted off in four conceptually independent parts: a) the search metaheuristic (SM), b) the iterative clustering (IC) component, c) the analyzer module (AM), and d) the local searcher (LS).

The SM component can be implemented by any optimization algorithm that generates diversified solutions of the search space. It must work as a full-time solution generator, exploring the search space by manipulating a set of solutions, according to its specific search strategy. IC component aims to gather similar solutions into groups, maintaining a representative cluster center for them. A *distance metric*, Δ, must be defined, *a priori*, allowing a similarity measure for the clustering process.

AM component examines each cluster, in regular intervals, indicating a probable promising cluster. A *cluster density*, also named *volume*, δ_j, is a measure that indicates the activity level inside the cluster j. For simplicity, δ_j can count the number of solutions generated by SM and grouped into c_j. Whenever δ_j reaches a certain *threshold* λ, meaning that some information template becomes predominantly generated by SM, such cluster must be better investigated to accelerate the convergence process on it. Clusters with lower δ_j can be eliminated or perturbed, as part of a mechanism that allows creating posteriorly other clusters, keeping framed the most active of them.

At last, the LS component is an internal searcher module that provides the exploitation of a supposed promising search area, framed by a cluster. In unconstrained continuous optimisation, the component LS has been commonly implemented by a Hooke-Jeeves direct search [3].

2.2 Artificial Bee Colony

In the ABC algorithm, the colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts. A bee waiting on the dance area for making decision to choose a food source, is called an onlooker and a bee going to the food source visited by itself previously is named an employed bee. A bee carrying out random search is called a scout. In the ABC algorithm, first half of the colony consists of employed artificial bees and the second half constitutes the onlookers. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source is exhausted by the employed and onlooker bees becomes a scout.

At each cycle ABC perform three basic operations: place employed bees on their food sources and evaluate their nectar amounts; after sharing the nectar information of food sources, onlookers select food source depending on their nectar amounts; send scout bees to explore new sources randomly on the search space. These main steps are repeated through a predetermined Maximum Cycle Number ($\mathcal{MCN}$) or until a termination criterion is satisfied.

Employed bees share the nectar amount of food sources on the dance area that will be used by onlooker bees as parameter to decide which source is better to be explored. The decision to be done by onlooker bees is governed by Equation 1:

$$p_i = \frac{fit_i}{\sum_{n=1}^{\mathcal{SN}} fit_n} \tag{1}$$

where p_i is the probability associated with food source i, fit_i is the fitness value of solution i and $\mathcal{SN}$ is the total number of food sources which is equal to the number of employed bees or onlooker bees. Both employed and onlooker bees modify their food sources, if the modification increase nectar amounts of those sources, onlooker and employed bees memorize the best source and forget previous one. Thus, to produce a candidate food position from the old one in memory is done by following equation:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{2}$$

where $k \in \{1, 2...SN\}$ and $j \in \{1, 2...D\}$ are randomly chosen indexes, and D is the dimension of variables.

A food source is assumed to be abandoned if its fitness cannot be improved through certain number of cycles. The value of pre-determined number of cycles is called "limit" for abandonment [2].

3 Artificial Bee Clustering Search

The Artificial Bee Clustering Search (ABCS) algorithm is based on potencial characteristic of detecting promissing areas presented by ABC algorithm. Such potencial characteristic can be seen on basic operations of ABC, such as the exploration and abandonment of a food source. An onlooker bee can be compared with a solution that must be assimilated by a cluster on a *CS-based algorithms, because this bee will increase a source's fitness after it has explored and found a better position to this food source. ABCS is proposed to represent all food sources as central regions of clusters and onlooker bees are solutions that can be or cannot be assimilated by these clusters.

```
program ABCS
   begin
     Initialize Population;
     repeat
       Place the employed bees on their food sources;
       Place the onlooker bees on the food sources depending on
       their nectar amounts;
       Memorize the quantity of activations (accounting of
       density) of each source;
       Perform Local Search in promissing areas (top activated
       food sources);
       Send the scouts to the search area for discovering new
       food sources;
       Memorize the best food source found so far;
     until requirements are met
   end;
end.
```

(Pseudo-code adapted from [2])

The accounting of density of a food source i (δ_i) is a parameter to perform local search. This parameter is another metric of a food source, that will inform which food sources are most promissing, to detect these promissing areas δ_i increases whenever an onlooker bee access the food source and gets a new better position. If a food source reach a certain threshold there is a time to perform local search in such source to find better solutions. This threshold is known in *CS as cluster density(Section 2).

$$\delta_i \geq \mathcal{PD} \cdot \frac{\mathcal{NP}}{|\mathcal{SN}|} \tag{3}$$

where $\mathcal{PD}$ is a pressure of density, $\mathcal{NP}$ is the colony size and $\mathcal{SN}$ is the total number of food sources. Pressure of density is a well known metric of *CS based algorithms that allows controlling the sensibility of the component AM.

ABCS is proposed to add to ABC some clustering elements without modifying the main structure of original algorithm. This work proposes to add two elements in ABCS: the amount of activations of each food source and local search routines. The algorithm used to perform local search in ABCS is the Hooke & Jeeves' algorithm [3].

4 Computational Experiments

Some functions were minimized to test the competitiveness of ABCS with others: A pure ABC, HABC (Hybrid Artificial Bee Colony) and ECS (Evolutionary Clustering Search [4]). The results in tables 3 and 4 were obtained allowing all algorithms to perform up to Maximum Function Evaluations ($Max\ FE$) (Table 1) in each one of 30 trials. The list of parameters of all algorithms are summarized in Table 1, where the value 80 of $\mathcal{NP}$ was used based on [2], the other values were empirically tested.

Table 1. Parameters of all algorithms

<table>
<tr><td></td><td colspan="6">Parameters</td></tr>
<tr><td>Algorithm</td><td>$\mathcal{NP}$</td><td>$Limit$</td><td>$\mathcal{MCN}$</td><td>$Error\ Rate$</td><td>$Max\ FE$</td><td>$\mathcal{PD}$</td></tr>
<tr><td>ABC</td><td rowspan="3">80</td><td rowspan="3">200</td><td rowspan="3">10000</td><td rowspan="5">0.0001</td><td rowspan="5">$2 * 10^6$</td><td rowspan="2"></td></tr>
<tr><td>HABC</td></tr>
<tr><td>ABCS</td><td>10</td></tr>
<tr><td></td><td>$PopSize$</td><td>$MaxGen$</td><td>$\mathcal{NC}$</td><td></td></tr>
<tr><td>ECS</td><td>200</td><td>8000</td><td>40</td><td>2.5</td></tr>
</table>

Table 2 shows the total benchmark test functions used in experiments, where D is the number of variables (Dimension), Min is the optimal solution and $Range$ is the known upper and lower bounds of the domain of functions's variables. HABC algorithm was not designed to detect promising areas, it only activate the Hooke & Jeeves Local Search whenever the better current solution has been improved (elitist strategy). The variables Mean, Standard Deviation (StdDev), H/A (Hits/Attempts) and FE (Function Evaluations) are necessary to analize the performance of algorithms. FE is a mean between total function calls in 30 executions for each function.

Table 2. 11 Test Functions used in the experiments

Function	Range	D	Min
Michalewicz	[0, π]	10	-9.6601
Rosenbrock	[-30,30]	30	0
Rastrigin	[-5.12,5.12]	30	0
$Schwefel_7$	[-500,500]	30	0
Langerman	[0,10]	10	-1.4999
Zakarov	[-5,10]	30	0
Ridge	[-64,64]	30	0
Colville	[-10,10]	4	0
Sphere	[-5.12,5.12]	30	0
Griewank	[-600,600]	30	0
Ackley	[-15,30]	30	0

Table 3. First set of functions

Function		ABC	HABC	ABCS	ECS
Michalewicz	Mean	-9.6600	-9.6600	-9.6600	-9.6355
	StdDev	6.8215E-005	6.9391E-005	5.8031E-005	0.0214
	H/A	30/30	30/30	30/30	6/30
	FE	35874.48	911211.31	97753.20	1745612.73
Rosenbrock	Mean	0.0343	0.0001	0.0001	6.4467E-005
	StdDev	0.0499	0.00002	2.5765E-005	4.8001E-005
	H/A	0/30	30/30	30/30	30/30
	FE	800081.10	185090.24	171625.75	508843.9
Rastrigin	Mean	0.0001	0.0001	0.0001	1.9567
	StdDev	7.1006E-005	1.1056E-005	3.0449E-005	0.8046
	H/A	30/30	30/30	30/30	2/30
	FE	70597.24	43929.13	30982.41	1928062.2
$Schwefel_7$	Mean	7.8033E-005	1295.28	0.0001	1075.82
	StdDev	6.2561E-005	252.88	6.1813E-005	416.58
	H/A	30/30	0/30	30/30	0/30
	FE	109758.65	2000000	733098	2000000
Langerman	Mean	-0.9551	-0.8450	-1.0715099	-0.8445
	StdDev	0.2802	0.1331	0.2954	0.1781
	H/A	4/30	1/30	7/30	2/30
	FE	733114.20	1952654.20	1001020.17	1918115.96

The ABCS algorithm presented better results with fewer calls for most minimized functions. HABC got bad performance at $Schwefel_7$, because this function is extremely multimodal and HABC performs many calls to Hooke & Jeeves, without information about which food sources are promissing, getting parked in local minimum points. ECS get better results to $Schwefel_7$ when the parameter $\mathcal{PD}$ is less than 2.5, 2.5 was used because is the best one to overall functions.

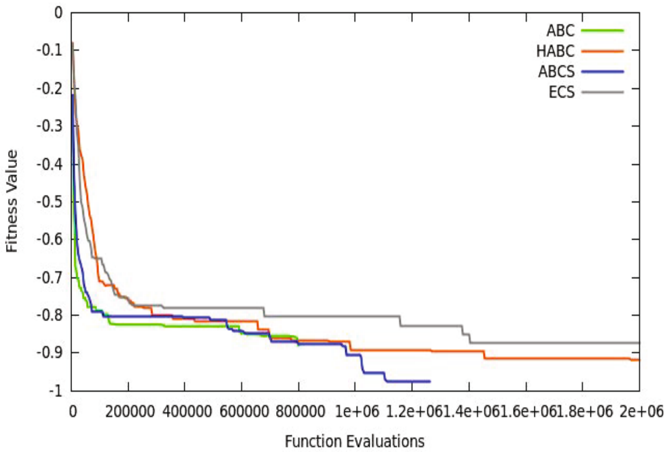

Fig. 1. Convergence history comparison of all algorithms at Langerman

Table 4. Second set of functions

Zakarov	Mean	136.45	0.023	0.014	2.5389
	StdDev	23.76	0.025	0.0172	1.555
	H/A	0/30	0/30	0/30	0/30
	FE	800040.58	2000000	2000000	2000000
Ridge	Mean	417.33	0.0003	0.0002	0.766
	StdDev	170.63	0.00007	1.98E-005	0.538
	H/A	0/30	2/30	25/30	0/30
	FE	800040.10	1938631.93	1447802.55	2000000
Colville	Mean	0.17965	0.0001	0.0001	0.00006
	StdDev	0.13612	2.86E-005	2.96E-005	5.5197E-005
	H/A	0/30	30/30	30/30	30/30
	FE	800218.517	22002.65	15069.03	3123.6
Sphere	Mean	100.000	100.000	100.000	100.000
	StdDev	3.67E-005	8.66E-006	3.08E-005	2.2968E-006
	H/A	30/30	30/30	30/30	30/30
	FE	30326.89	57343.65	11508.96	3274.2
Griewank	Mean	0.0001	0.0001	0.0001	0.0001
	StdDev	4.82E-005	9.07E-006	0.00002	3.574E-005
	H/A	30/30	30/30	30/30	30/30
	FE	55186.20	39759	11105.58	4943
Ackley	Mean	0.0001	0.0001	0.0001	8.646E-005
	StdDev	3.33E-005	1.46E-005	2.38E-005	7.1095E-005
	H/A	30/30	30/30	30/30	30/30
	FE	61055.17	40015.55	32731.55	366911.866

Figure 1 illustrates the progress of all algorithms minimizing Langerman function. The mean of objective function (known as fitness value in the chart) was calculated between 30 trials at intervals of 5000 function evaluations.

5 Conclusion

In this work a new approach is proposed combining Clustering Search with Artificial Bee Colony. The proposal is called Artificial Bee Clustering Search (ABCS) and two elements of *CS are added to ABCS aiming to improve inherent characteristics of detecting promissing food sources by ABC: Quantity of activations of each food source and the Local Search component. Some unconstrained continuous optimisation functions are minimized and ABCS presented competitive results compared to other algorithms: Pure ABC, HABC and ECS.

Future work consist of testing some *CS components as the *distance metric* and different types of bee assimilation by food sources.

References

1. Karaboga, D.: An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
2. Karaboga, D., Akay, B.: A comparative study of Artificial Bee Colony Algorithm. Applied Mathematics and Computation 214, 108–132 (2009)
3. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. J. Assoc. Comput. Mach. 8, 212–229 (1961)
4. Oliveira, A.C.M., Lorena, L.A.N.: Detecting promising areas by evolutionary clustering search. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 385–394. Springer, Heidelberg (2004)
5. Chaves, A.A., Correa, F.A., Lorena, L.A.N.: Clustering Search Heuristic for the Capacitated p-median Problem. Springer Advances in Software Computing Series 44, 136–143 (2007)
6. Costa, T.S., Oliveira, A.C.M., Lorena, L.A.N.: Advances in Clustering Search. Advances in Soft Computing 73, 227–235 (2010)