Advances in Clustering Search

Tarcisio Souza Costa, Alexandre César Muniz de Oliveira,
and Luiz Antonio Nogueira Lorena

Abstract. The 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. Although, recent applications have reached success in combinatorial optimisation problems, nothing new has arisen concerning diversification issues when population metaheuristics, as evolutionary algorithms, are being employed. In this work, recent advances in the *CS are commented and new features are proposed, including, the possibility of keeping population diversified for more generations.

1 Introduction

The 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 [1, 2]. This generalized approach 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 hopefully can correspond to relevant areas of attraction in the most of search metaheuristics, including EAs.

Tarcisio Souza Costa
Universidade Federal do Maranhão, São Luís MA Brasil
e-mail: priestcp@gmail.com

Alexandre César Muniz de Oliveira
Universidade Federal do Maranhão, São Luís MA Brasil
e-mail: acmo@deinf.ufma.br

Luiz Antonio Nogueira Lorena
Instituto Nacional de Pesquisas Espaciais
e-mail: lorena@lac.inpe.br

E. Corchado et al. (Eds.): SOCO 2010, AISC 73, pp. 227–235.
springerlink.com

Relevant search areas can be treated with special interest by the algorithm as soon as they are discovered. The clusters work as sliding windows, framing the search areas and giving a reference point (center) to problem-specific local search procedures. Furthermore, the cluster center itself is always updated by incoming inner solutions. The center update is called assimilation [1, 3].

This basic idea was employed to propose the Evolutionary Clustering Search (ECS) early applied to unconstrained continuous optimisation [1]. Posteriorly, the search guided by clustering was extended to a GRASP (Greedy Randomized Adaptive Search Procedure [4]) with VNS (Variable Neighborhood Search [5]), and applied to Prize Collecting Traveling Salesman Problem (PCTSP) [3]. Despite the expected and desirable population convergence, it must occur in a controlled way, avoiding diversification loss and, in consequence, a poor overall performance. *CS has an inherent potential of controlling the convergence not explored yet: search areas may be monitored not allowing oversearch in a promising area.

This paper is devoted to discuss the recent *CS applications in optimisation problems, focusing how its components were implemented for each application. Other issue treated in this work is concerning population diversification. New features to improve *CS are proposed at last. The remainder of this paper is organized as follows. Section 2 describes the concepts behind *CS's components. Recent advances in each component are described in sections 2.1, 2.2, 2.3 and 2.4, including the now proposed diversification mechanism adequate for the approach improvement. The findings and conclusions are summarized in section 3.

2 Clustering Search Foundations

The *CS attempts to locate promising search areas by framing them by clusters. A cluster can be defined as a tuple $\mathscr{G} = \{c, r, s\}$, where c and r are the *center* and the *radius* of the area, respectively. The radius of a search area is the distance from its center to the edge. There could exist different *search strategies* s associated to the clusters. Initially, the center c is obtained randomly and progressively it tends to slip along really promising sampled points in the close subspace. The total cluster volume is defined by the radius r and can be calculated, considering the problem nature. It is important that r must define a search subspace suitable to be exploited by the search strategies s associated to the cluster.

For example, in unconstrained continuous optimisation, it is possible to define r in a way that all Euclidean search space is covered depending on the maximum number of clusters [1]. In combinatorial optimisation, r can be defined as the number of movements needed to change a solution into another. The neighborhood is function of some distance metric related to the search strategy s [1]. *CS can be splitted off in four independent parts: a search metaheuristic (SM); an iterative clustering (IC) component; an analyzer module (AM); and a local searcher (LS).

2.1 Search Metaheuristic

SM component works as a full-time solution generator, according to its specific search strategy, *a priori*, performing independently of the remaining parts, and manipulating a set of $|P|$ solutions. For example, early implementations of Evolutionary Clustering Search (ECS) employ a real-coded steady-state genetic algorithm as SM component in which a sampling mechanism in selection [1] or updating [1] is used to fed the clustering process. Non-population approaches, as Variable Neighborhood Search (VNS) [6], Simulated Annealing (SA) [7] and Greedy Randomized Adaptive Search Procedure [8] have been employed as SM component to solve combinatorial optimisation as well.

The VNS with Clustering Search (VNS-CS) always executes VND local search, differently of the Greedy Randomized Adaptive Clustering Search GRACS, so named in allusion to a special form to gather *CS with GRASP [8]. In the GRACS the full-time solution generator is a greedy constructive phase, controlled by the parameter α that indicates how greedy a given solution must be built. The improvement phase of a typical GRASP may be computational costly. A supposed advantage of the GRACS is rationally apply costly local search algorithms only to promising areas, represented by cluster centers [8]. Figure 1 illustrates the structural difference between GRASP and GRACS. One can observe that the intrinsic improvement mechanism of GRASP is called according the *CS's criteria.

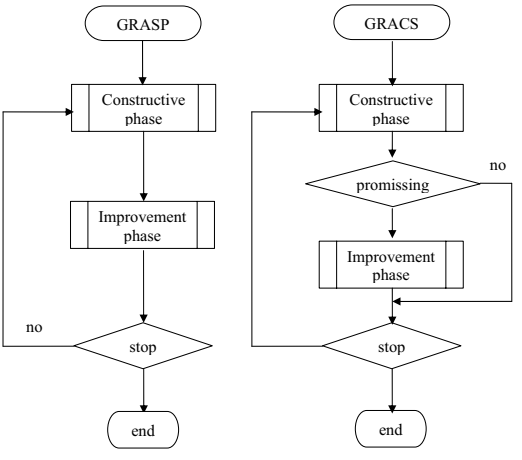

Fig. 1 Difference between GRASP and GRACS

2.2 Iterative Clustering

IC component aims to gather similar solutions into groups, maintaining a representative cluster center for them. To avoid extra computational effort, IC is fed, progressively, by solutions generated in each regular iteration of SM. A maximum

number of clusters $\mathscr{NC}$ is an upper bound value that prevents an unlimited cluster creation, initially. A *distance metric*, $\wp$, must be defined, allowing a similarity measure for the clustering process. Solutions s_k generated by SM are passed to IC that attempts to group as known information, according to $\wp$. If the information is considered sufficiently new, it is kept as a center in a new cluster, c_{new}. Otherwise, redundant information activates the closest center c_i (cluster center that minimizes $\wp(s_k, c_{j=1,2,\cdots,})$), causing an *assimilation* of the information represented by s_k. Considering $\mathscr{G}_{j\ (j=1,2,\cdots)}$ as all current detected clusters:

$$c_{new} = s_k \text{ if } \wp(s_k, c_j) > r_j, \forall \mathscr{G}_j, \text{ or} \tag{1}$$

$$c_i' = c_i \oplus \beta(s_k \ominus c_i), \text{ otherwise.} \tag{2}$$

where $\oplus$ and $\ominus$ are abstract operations over c_i and s_k meaning, respectively, addition and subtraction of solutions. The operation $(s_k \ominus c_i)$ means the difference between the solutions s_k and c_i, considering the distance metric. A certain percentage, β, of this difference is used to update c_i, giving c_i'. According to β, the assimilation can assume different forms: simple, crossover or path assimilations [2]. A maximum number of clusters, $\mathscr{NC}$, is defined *a priori*. To cover all the search space, a common radius r_t must be defined, working as a threshold of similarity. For the Euclidean space, for instance, r_t has been defined as [1]:

$$r_t = \frac{x_{sup} - x_{inf}}{2 \cdot \sqrt[n]{|C_t|}} \tag{3}$$

where $|C_t|$ is the current number of clusters (initially, $|C_t| = \mathscr{NC}$, in general, about 20-30 clusters), x_{sup} and x_{inf} are, respectively, the known upper and lower bounds of the domain of variable x, considering that all variables x_i have the same domain.

For combinatorial optimisation, the threshold of similarity concerns the number of moves need to change a solution in another considering a specific neighborhood relationship. For the simple 2-swap neighborhood, commonly used in permutation search spaces, r_t has been defined as [8]:

$$r_t = \lceil \lambda \cdot N \rceil \tag{4}$$

where N is the permutation size and λ is the percentage of dissimilarity, in general, about 0.9 has been set.

The assimilation process is applied over the closest center c_i, considering the new generated solution s_k. According to β in Eq. 2, the assimilation can assume different forms. The three types of assimilation, proposed in [1], are: simple, crossover and path relinking. In simple assimilation, $\beta \in [0,1]$ is a constant parameter, meaning a deterministic move of c_i in the direction of s_k. Only one internal point is generated more or less closer to c_i, depending on β, to be evaluated afterwards. The greater β, the less conservative the move is. This type of assimilation can be employed only with real-coded variables, where percentage of intervals can be applied

to. Crossover assimilation means any random operation between two candidate solutions (parents), giving other ones (offsprings), similarly as a crossover operation in EAs. In this assimilation, β is an $n-$dimensional random vector and c_i' can assume a random point inside the hyper plane containing s_k and c_i. Since the whole operation is a crossover or other binary operator between s_k and c_i, it can be applied to any type of coding or even problem (combinatorial or continuous one).

Simple and crossover assimilations generate only one internal point to be evaluated afterwards. Path assimilation, instead, can generate several internal points or even external ones, holding the best evaluated one to be the new center. These exploratory moves are commonly referred in path relinking theory [9]. In this assimilation, β is a $\eta-$dimensional vector of constant and evenly spaced parameters, used to generate η samples taken in the path connecting c_i and s_k. Since each sample is evaluated by the objective function, the path assimilation itself is an intensification mechanism inside the clusters, allowing yet sampling of well-succeeded external points, extrapolating the s_k-c_i interval. For combinatorial optimisation, path assimilation has been chosen [6, 7, 8, 10], becoming the most popular of them. The more distance $\wp(c_i, s_k)$, the more potential solutions exist between c_i and s_k. The sampling process, depending on the number of instance variables, can be costly, since each solution must be evaluated by objective function.

For continuous optimisation, path relinking might be replaced by some kind of direct search movement, as Hooke-Jeeves algorithm [11], in which, the best solution found in the activated cluster is set as new center. In this case, assimilation must consider other inner points, besides c_i and s_k to proceed with the direct search algorithm.

2.3 Analyzer Module

AM component examines each cluster, indicating a probable promising cluster or a cluster to be eliminated. A *cluster density*, δ_i, is a measure that indicates the activity level inside the cluster *i*. For simplicity, δ_i counts the number of solutions generated by SM (selected or updated solutions in the EA case) [1, 10]. Whenever δ_i reaches a certain *threshold*, meaning that some information template becomes predominantly generated by SM, such information cluster must be better investigated to accelerate the convergence process on it. In the EA case, the promising threshold has been set according to the number of cluster, $\mathscr{NC}$, and the number of generated individuals, $|P_{new}|$, at each SM iteration. Considering that each cluster should receive the same number of individuals, a cluster *i* is promising whether:

$$\delta_i > \frac{\mathscr{NC}}{|P_{new}|} \tag{5}$$

On the other hand, clusters with lower δ_i are eliminated, as part of a mechanism that allows creating other centers of information, keeping framed the most active of them. At the end of each generation, the AM performs the *cooling* of all clusters,

resetting the accounting of δ_i. A cluster is considered inactive when no activity has occurred in the last generation.

In EA fashion, premature convergence is an undesirable behaviour in which a population converges too early, resulting in a suboptimal solution. In this context, the individuals are concentrated in few search areas, not being able to generate offsprings that are superior to their parents. Premature convergence may happen in case of loss of diversification, i.e., every individual in the population become identical. In early ECS applications, the clustering process did not affect the set of $|P|$ solutions in SM. In this work, a control of diversification is being proposed in order to avoid premature or even undesirable convergence. By this new feature, the AM module plays another important role in the ECS. It is responsible by detecting promising areas, eliminating inactive clusters and *avoiding overcrowded areas*. Considering the same basic idea of Eq. 5, a cluster i is overcrowded whether:

$$\delta_i > \kappa \frac{\mathscr{NC}}{|P_{new}|} \tag{6}$$

where κ must be defined *a priori*, but satisfactory behavior can be reached setting it to 0.5. The population cannot be updated with a new individual grouped in a overcrowded cluster, interfering in the population diversification.

In the first experiment, an ECS, a steady-state real-coded genetic algorithm, similar to [1], was run during 10 generations (about 200 updatings), for some test functions, commonly found in literature as optimisation benchmark, as *Rastrigin* and *Langerman* [12]. Figure 2 shows the first experiment with this two test functions.

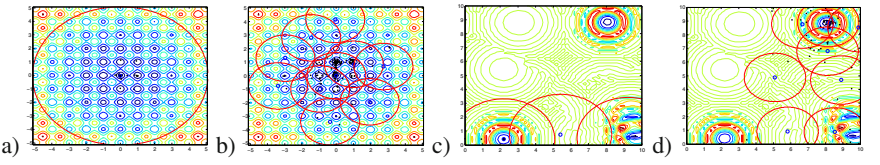

Fig. 2 Test functions after 10 generations: a) *Rastrigin* without and b) with diversification control; c) *Langerman* without and d) with diversification control

The circles show active clusters framing search areas. The dots mean individuals scattered along the promising areas. One can observe that mechanism to prevent overcrowded clusters is effective against premature convergence since ECS keeps population diversified. For example, in Figure 2a, one can note that with the same number of generations, just one cluster is sufficient to cover all population without diversification control, while several clusters are necessary to cover the population with diversification control (Figure 2b).

The second experiment consists of applying the ECS to other test-functions found in literature to analyze if the diversification control contributes to the algorithm performance improvement. The test-functions used in this experiment are showed in Table 1. A study about all of these functions can be found at [13].

Table 1 Test-functions

function	var	opt	$x_i^{low}; x_i^{up}$	function	var	opt	$x_i^{low}; x_i^{up}$
Ackley	n	0	-15 ; 30	Goldstein	2	3	-2 ; 2
Griewank	n	0	-600; 600	Easom	2	-1	-100;100
Schwefel	n	0	-500;500	Shekel 10	4	-10,536	-10 ; 10
Rosenbrock	n	0	-5,12;5,12	Langerman	5	-1,4	0;10
Rastrigin	n	0	-5,12;5,12	Langerman	10	-1,4	0;10
Sphere	n	0	-5,12;5,12	Michalewicz	5	-4,687	0; π
Zakharov	n	0	-5 ; 10	Michalewicz	10	-9,66	0; π
				Hartman	6	-3,322	0 ; 1

The results in Figure3 were obtained, in 20 trials, allowing ECS to perform up to $500,000$ objective function calls at each trial. For each test-function, the average of well-succeeded trials (hits), the average of gap (difference between the best found solution and optimal one) and the average of objective function calls were considered to verify the algorithm performance. The parameter κ was set to $\{0.1, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0\}$, respectively, where values above 1.0 allow overcrowded cluster. It is expected that very low values (e.g 0.1 and 0.3) does not allow overcrowded cluster, but the desirable convergence also is affected, causing a poor performance of the ECS.

In Figure3, one can observe the behavior of the algorithm along the experiment, varying κ. The best results concerning hits and gap were obtained for $0.5 \leq \kappa \leq 1.0$, while the best result regarding the number of function calls was obtained about $\kappa = 1.0$. One can conclude that, as expected, the diversification control can improve the ECS performance for continuous optimisation, depending on the equilibrium

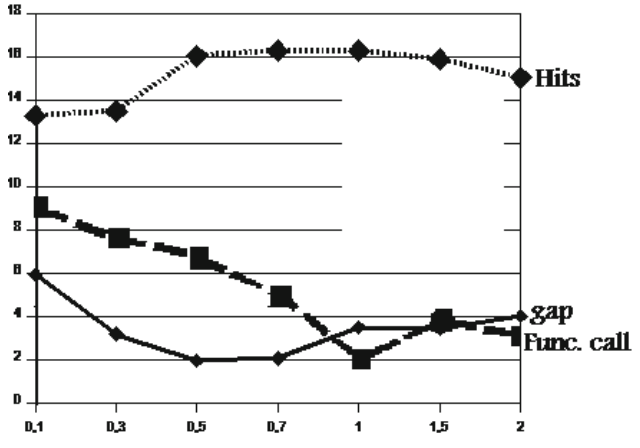

Fig. 3 Performance results: hits, function calls and GAP for test-functions

obtained during the evolutionary process. The lack of convergence prejudices the algorithm behavior.

2.4 Local Searcher

At last, the LS component is an internal searcher module that provides the exploitation of a supposed promising search area, framed by a cluster. This process can happen after AM having discovered a target cluster. The component LS has been implemented by a Hooke-Jeeves direct search in continuous optimisation [1]. For combinatorial optimisation, costly algorithms as VND [6] and 2-Opt heuristic, which explores several 2-Opt neighborhoods [8], have been employed. Given the pressure of density, $\mathscr{PD}$, responsable for controlling the sensibility of the component AM, the component LS has been activated, at once, if

$$\delta_i \geq \mathscr{PD} \cdot \frac{\mathscr{NS}}{|C_t|} \tag{7}$$

3 Conclusion

This paper is devoted to discuss the recent applications of Clustering Search (*CS) in combinatorial and continuous optimisation, as well as a new diversification control feature here proposed and validated. In a general way, *CS attempts to locate promising search areas by cluster of solutions. In this work, the concepts behind *CS's components are presented, taking a special care with new ways of implementing them. A new type of assimilation based on direct search algorithms is suggested and a diversification mechanism is effectively proposed and validated by performance analysis. The diversification control allowed to keep controlled the convergence pressure in the cluster without damaging the overall performance. For further work, it is intended to proposed a adaptive mechanism of control of the diversification and to apply new features for combinatorial optimisation.

Acknowledgements. The authors acknowledge CNPq for the support received in the research project " Logística e Planejamento de Operações em Terminais Portuários".

References

1. 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)
2. Oliveira, A.C.M., Lorena, L.A.N.: Hybrid evolutionary algorithms and clustering search. In: Grosan, C., Abraham, A., Ishibuchi, H. (eds.) Hybrid Evolutionary Systems. SCI, vol. 75, pp. 81–102 (2007)

3. Chaves, A.A., Lorena, L.A.N.: Hybrid algorithms with detection of promising areas for the prize collecting travelling salesman problem. In: HIS 2005: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, pp. 49–54. IEEE Computer Society, Washington (2005)
4. Resende, M.G.C.: Greedy randomized adaptive search procedures (grasp). Journal of Global Optimization 6, 109–133 (1999)
5. Hansen, P., Mladenovic, N.: Variable neighborhood search. Computers and Operations Research 24, 1097–1100 (1997)
6. Biajoli, F.L., Lorena, L.A.N.: Clustering Search Approach for the Traveling Tournament Problem. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 83–93. Springer, Heidelberg (2007)
7. 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)
8. Oliveira, A.C.M., Lorena, L.A.N.: Pattern Sequencing Problems by Clustering Search. In: Sichman, J.S., Coelho, H., Rezende, S.O. (eds.) IBERAMIA 2006 and SBIA 2006. LNCS (LNAI), vol. 4140, pp. 218–227. Springer, Heidelberg (2006)
9. Glover, F., Laguna, M.: Fundamentals of scatter search and path relinking. Control and Cybernetics 29(3), 653–684 (2000)
10. Filho, G.R., Nagano, M.S., Lorena, L.A.N.: Evolutionary clustering search for flowtime minimization in permutation flow shop. In: Hybrid Metaheuristics, pp. 69–81 (2007)
11. Hooke, R., Jeeves, T.A.: "Direct search" solution of numerical and statistical problems. Journal of the ACM 8(2), 212–229 (1961)
12. Digalakis, J., Margaritis, K.: An experimental study of benchmarking functions for Genetic Algorithms. IEEE Systems Transactions, 3810–3815 (2000)
13. Oliveira, A.: Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuos e discretos. PhD Thesis. INPE (2004)