Pattern Sequencing Problems by Clustering Search

Alexandre C. M. Oliveira[1] and Luiz A. N. Lorena[2]

[1] Universidade Federal do Maranhão - UFMA, Departamento de Informática, São Luís MA, Brasil
acmo@deinf.ufma.br

[2] Instituto Nacional de Pesquisas Espaciais - INPE, Laboratório Associado de Computação e Matemática Aplicada, São José dos Campos SP, Brasil.
lorena@lac.inpe.br

Abstract. Modern search methods for optimization consider hybrid search metaheuristics those employing general optimizers working together with a problem-specific local search procedure. The hybridism comes from the balancing of global and local search procedures. A challenge in such algorithms is to discover efficient strategies to cover all the search space, applying local search only in actually promising search areas. This paper proposes the Clustering Search (*CS): a generic way of combining search metaheuristics with clustering to detect promising search areas before applying local search procedures. The clustering process aims to gather similar *information* about the problem at hand into groups, maintaining a representative solution associated to this information. Two applications to combinatorial optimization are examined, showing the flexibility and competitiveness of the method.
Keywords: Hybrid search metaheuristic; pattern sequencing problem; Clustering search.

1 Introduction

Local search methods have been combined with search metaheuristics in different ways to solve particular problems more efficiently. Hill-climbing procedures are largely employed in the so called memetic algorithms (MA) as a Lamarckian learning process [1]. For example, a simple crossover can work as a local search around the parents, hill-climbing by repeatedly generating some number of offspring and replacing the worst parent [2].

The main challenge in such hybrid methods is to define efficient strategies to cover all search space, applying local search only in actually promising areas. Elitism plays an important role towards achieving this goal, once the best solutions represent promising neighborhood. However, such well-evaluated solutions can be concentrated in few areas and thus the exploitation moves are not rationally applied.

An approach attempting to find out relevant areas for continuous optimization is a parallel hill-climber, called Universal Evolutionary Global Optimizer

(UEGO) by its authors [3]. The separated hill-climbers work in restricted search regions (or clusters) of the search space. The volume of the clusters decreases as the search proceeds, resulting in a cooling effect similar to simulated annealing. Each cluster center represents diversity and quality, since it is result of hill-climbing procedures [3].

The scatter search (SS), proposed in [4], by another way, separates diversified and improved solutions in two sets: the reference set, containing the best solutions found so far and the diversity set, containing the solutions most distant from the solutions of the reference set. The solutions in these two sets are improved by local search. Thus, SS employs systematic exploration/exploitation moves, combining quality and representative solutions [4].

Clusters of mutually close solutions hopefully can correspond to relevant areas of attraction in the most of search metaheuristics, such as Genetic Algorithms (GA) [5] and Greedy Randomized Adaptive Search Procedure (GRASP) [6]. Relevant search areas can be treated with special interest by the algorithm as soon as they are discovered. This basic idea was first employed to propose the Evolutionary Clustering Search (ECS), applied to unconstrained continuous optimization [7]. Posteriorly, the search guided by clustering was extended to a GRASP with VNS (Variable Neighborhood Search[8]), and applied to Prize Collecting Traveling Salesman Problem (PCTSP) [9].

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 a permanent interaction with inner solutions, called assimilation [7, 9].

This paper proposes the Clustering Search (*CS) as a generalized way of detecting promising search areas by clusters of solutions, suitable to be employed together with any metaheuristic and applicable to combinatorial and continuous optimization problems. To consolidate this approach as a flexible method, an ECS and a GRACS(Greedy Randomized Adaptive Clustering Search), both based on *CS, are proposed for pattern sequencing problems.

The remainder of this paper is organized as follows. In Section 2, the basic ideas and conceptual components of *CS are described. Theoretical issues of the sequencing problems are presented in Section 3. In section 4, the GRACS and the ECS are proposed for pattern sequencing problems. The computational results are examined in Section 5 and conclusions are summarized in Section 6.

2 Clustering Search foundations

The *CS employs clustering for detecting promising areas of the search space. It is particularly interesting to find out such areas as soon as possible to change the search strategy over them. An area can be seen as a search subspace defined by a neighborhood relationship in metaheuristic coding space.

A cluster can be defined as a tuple $\mathcal{G} = \{c, r, s\}$, 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 also 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 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 strategy s associated to the cluster.

For example, in unconstrained continuous optimization, it is possible to define r in a way that all search space is covered depending on the maximum number of clusters [7]. In combinatorial optimization, r can be defined as the number of movements needed to change a solution into another. In both case, the neighborhood is function of some distance metric related with the search strategy s, i.e., a problem-specific local search to be employed into the cluster.

2.1 Components

*CS can be splitted off in 4 conceptually independent parts: (a) a search metaheuristic (SM); (b) an iterative clustering (IC) component; (c) an analyzer module (AM); and (d) a local searcher (LS). Fig. 1 brings its conceptual design.

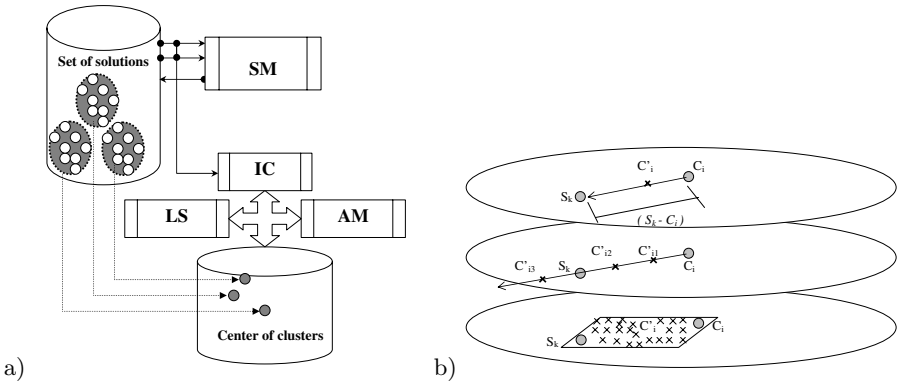

Fig. 1. a) *CS components; b) Simple, path and crossover assimilations, respectively.

The SM component works as a full-time solution generator, according to its specific search strategy, performing independently of the remaining parts, and manipulating a set of $|P|$ solutions ($|P| > 1$ for evolutionary algorithms - EA). In an EA fashion, for example, individuals are selected, crossed over, and updated for the next generations. This entire process works like an infinite loop, in which solutions are generated along the iterations.

IC component aims to gather similar solutions into groups, maintaining a representative cluster center for them. To avoid extra computational effort, IC is designed as an online process, in which the clustering is progressively fed by solutions generated in each regular iteration of SM. A maximum number of clusters $\mathcal{NC}$ is a bound value that prevents a unlimited cluster creation. For

a $n-$dimensional problem, the IC complexity is, at most, $O(\mathcal{NC} \cdot n)$ when all cluster centers are allocated. A *distance metric*, $\wp$, 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*, 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 solutions, in the EA case[7]). 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. Clusters with lower δ_i are eliminated, as part of a mechanism that will allow creating other centers of information, keeping framed the most active of them. The cluster elimination does not affect the set of $|P|$ solutions in SM. Only the center of information is considered irrelevant for the process.

At last, the LS component is an internal searcher module that provides the exploitation of a supposed promising search area, framed by cluster. This process can happen after AM having discovered a target cluster or it can be a continuous process, inherent to IC, being performed whenever a new point is grouped. LS can be considered as the particular search strategy s associated with the cluster.

2.2 The assimilation process

Solutions 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. Otherwise, redundant information activates a cluster, causing some kind of perturbation on it. This perturbation means an *assimilation process*, in which the previously learned knowledge (center of the cluster) is updated by the received information. More precisely, the assimilation process is applied over the closest center c_i, considering the new generated solution s_k. The general assimilation form is:

$$c_i' = c_i \oplus \beta(s_k \ominus c_i) \tag{1}$$

where $\oplus$ e $\ominus$ 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 vector of differences between each one of the n variables compounding the solutions s_k and c_i, considering $\wp$. A certain percentage β of the vector is the update step for c_i, giving c_i'. According to β, the assimilation can assume different forms: simple, path and crossover assimilations, represented in Fig. 1b.

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 is applicable. Its specific form is:

$$c_i' = c_i + \beta(s_k - c_i) \tag{2}$$

Although the name, crossover assimilation is not necessarily associated with an evolutionary operator. In a general way, it means any random operation between two candidate solutions, giving other ones, 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 e 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). The $\overrightarrow{\beta}$ parameter is resulting from the type of crossover employed, not the crossover parameter itself. The crossover assimilation can be written by:

$$c_i' = c_i + \overrightarrow{\beta} \cdot (s_k - c_i) \tag{3}$$

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. It seems to be advantageous, but clearly costly. These exploratory moves are commonly referred in path relinking theory [10]. In path 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. The new center c_i' is given by:

$$\begin{array}{rl} c_i' = c_V',\, f(c_V') = \min\left\{f(c_1'), f(c_2'), \cdots, f(c_\eta')\right\} \\ c_j' \quad\;\; = c_i + \beta_j(s_k - c_i) \\ \beta_j \quad\;\; \in \{\beta_1, \beta_2, \cdots, \beta_\eta\} \end{array} \tag{4}$$

where $\beta_j \in \{]0,1[\bigcup]1,\infty]\}$, $f(c_V')$ is the objective function of the best evaluated solution sampled in the path and min is concerned to minimization problems.

With respect to the infinite interval in (4), it means the external points can be sampled indefinitely while there are well-succeeded points beyond s_k. A well-succeeded point has an objective function value better than the previous point sampled, in a way that a worse point stops the sampling. In the Fig. 1b, the point c_{i3} is evaluated after s_k. Such extrapolation move is suitable for path relinking [10] and it can intentionally shift the center cluster beyond the cluster edge.

3 Theoretical issues of the pattern sequencing problem

Pattern sequencing problems may be stated by a matrix with integer elements where the objective is to find a permutation (or sequencing) of rows or patterns (client orders, or gates in a VLSI circuit, or cutting patterns) minimizing some objective function [11]. Objective functions considered here differ from traveling salesman-like problems because the evaluation of a permutation can not be computed by using values that only depend on adjacent patterns. There are

two similar pattern sequencing problems found in the literature: Minimization of Open Stacks Problem (MOSP) and Gate Matrix Layout Problem (GMLP) [12]. The difference between them resides only in their enunciation. This work is considering only the largest GMLP instances found in the literature.

GMLPs are related to one-dimensional logic arrays and programmable logic arrays folding [13]. The data for a GMLP are given by an $I \times J$ binary matrix $\mathbf{P}$, representing gates (rows) and nets (columns), where $\mathbf{P}_{ij} = 1$, if gate i belongs to net j, and $\mathbf{P}_{ij} = 0$ otherwise. Non-overlapping nets can be placed at the same connection track, minimizing the layout area (cost). A GMLP consists of determining a sequence of gate that minimizes the maximum of tracks (MOT) in the circuit. Another binary matrix, here called matrix $\mathbf{Q}$, can be used to calculate the MOT for a certain gate permutation. It is derived from the input matrix $\mathbf{P}$, by the following rules:

- $\mathbf{Q}_{ij} = 1$ if there exists x and $y | \pi(x) \leq i \leq \pi(y)$ and $\mathbf{P}_{xj} = \mathbf{P}_{yj} = 1$;
- $\mathbf{Q}_{ij} = 0$, otherwise;

where $\pi(b)$ is the position of gate b in the permutation. The $\mathbf{Q}$ shows the consecutive-ones property [14] applied to $\mathbf{P}$: in each column, "0" 's between "1"'s are replaced by "1" 's. The sum of "1" 's, by row, computes the number of open stacks when each pattern is processed. Fig. 2 shows an example of matrix $\mathbf{P}$, its corresponding matrix $\mathbf{Q}$, and the number of tracks in the circuit. At most, 7 tracks ($MOT = \max\{3, 3, 3, 5, 6, 7, 7, 5, 3\} = 7$) are needed to manufacture a circuit with permutation $\pi_0 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

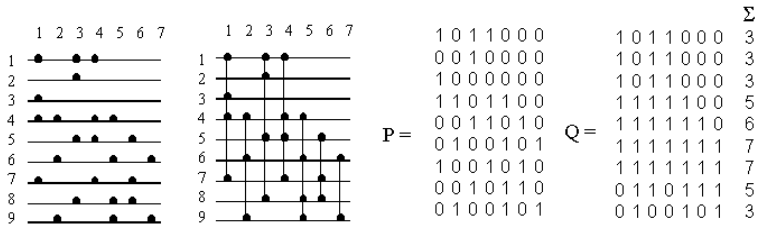

Fig. 2. GMLP (or MOSP) instance: matrix $\mathbf{P}$ and corresponding $\mathbf{Q}$.

4 GRACS and ECS Implementations

Two *CS metaheuristics are now described: a GRACS (Greedy Randomized Adaptive Clustering Search) and an Evolutionary Clustering Search ECS. In the latter, the component SM is a steady-state GA employing well-known genetic operators as roulette wheel selection [5], block-order crossover (BOX) [15], and 2-swap mutation [16]. In BOX, the parent(A) and parent(B) are mixed into

only one offspring, by copying blocks of both parents, at random. Pieces copied from a parent are not copied from other, keeping the offspring feasible (Fig. 3a).

In general, GRASP consists of a greedy construction phase and a subsequent one which iterative local search improvements are made in the previously obtained greedy solution. In GRACS, the component SM is a modified GRASP, removing the native local search procedure from it and adding the IC-AM) components, which are responsible for intermediating the local search calls. Instead of always to perform the local search, the improvements are made according to the IC-AM criterion for promising search areas.

The constructive greedy procedure chosen for pattern sequencing problems is based on the rule for filling schemata [17], which says that the new pattern to be included in greedy solution shall minimize a bit-to-bit *xor* difference with respect to the previous included. In other words, examining all the J columns (nets) of the candidate list, matching bit-to-bit and summing all the xor results ($xor(1,0) = xor(0,1) = 1$ and $xor(0,0) = xor(1,1) = 0$), the patterns are included in a sequence that improves the similarity between adjacent ones. A typical GRASP parameter, α, that balances the greedy/random behaviour of the constructive phase, was set to 0.10, meaning that only 10% of the candidate list is considered to choose the next included pattern in solution.

The component LS in both *CS algorithms was implemented by a 2-Opt hill-climbing procedure which is applied to the center of promising cluster. The hill-climbing explores a search tree, considering several 2-Opt neighborhoods (Fig. 3b). The best neighbor from a level (bold circle) is taken as starting point to the next, respecting a maximum width l (maximum number of swaps at each level) and height m (maximum number of levels). For GRACS, m was set unlimited, i.e., while better solutions were being found. Rather, for ECS, $m \leq 40$. For both, ECS and GRACS, l was set to 0.70, meaning that 70% of the patterns can be exchanged in each level during the local search procedure.

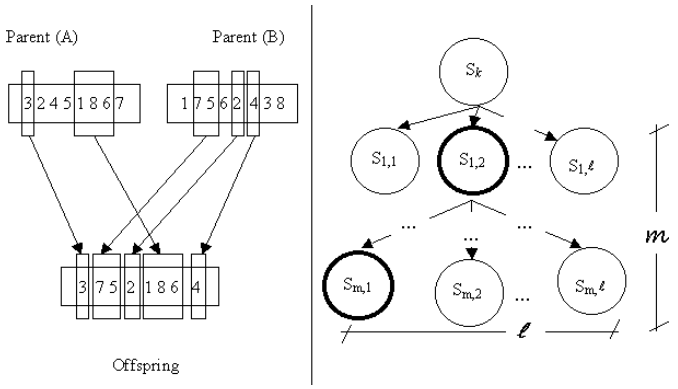

Fig. 3. (a) Block order crossover and (b) 2-Opt hill-climbing tree.

Concerning to component IC, for both algorithms, the 2-swap distance metric is employed, i.e., the number of 2-swap needed to move a solution, along the search space, to another. Identical solutions need no changes to turn one into other. By the other side, completely distinct solutions may need about $I - 1$ 2-swap moves to lead a point to another. The radius of a cluster is given by:

$$r_t = \lceil 0,9I \rceil \tag{5}$$

i.e., a relatively greater radius, because it requires only 10% of labels matching for a given pattern sequencing to be considered close enough to the center of a cluster. Whenever a selected individual s_k is *far away* from all centers (a distance above r_t), then a new cluster must be created.

In this application, the path assimilation was chosen. 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. In Table 1, a completely 2-swap path between two solutions, c_i and s_k, can be seen.

Table 1. Example of full path between center c_i and new point s_k.

c_i=	1 2 3 4 5 6 7 8 9	comparison	swap	evaluation
1)	4 2 3 1 5 6 7 8 9	1	1	1
2)	4 8 3 1 5 6 7 2 9	1	1	1
3)	4 8 5 1 3 6 7 2 9	1	1	1
4)	4 8 5 9 3 6 7 2 1	1	1	1
5)	4 8 5 9 1 6 7 2 3	1	1	1
6)	4 8 5 9 1 7 6 2 3	1	1	1
7)	4 8 5 9 1 7 6 2 3	1		
8)	4 8 5 9 1 7 6 2 3	1		
s_k=	4 8 5 9 1 7 6 2 3	8	6	6

Each comparison means one iteration in the assimilation algorithm which also can occur one swap/evaluation of the intermediary points. At last, the center will be shiffted to the best point evaluated in this path. Actually, there have been occurred 6 pattern swaps and, consequently, 6 objective function calls. The distance $\wp(c_i, s_k)$ is not necessarily 6 because other paths with distance less than 6 could be found. However, **CS* applications require computing such distance to associate the point to a particular center during the clustering process. Therefore, $\wp(c_i, s_k)$ is estimate considering the number of pattern in different positions in each permutation (variables that do not match). This value is still decremented by one, because even all I patterns were in different positions in each permutation, it would be generated at most $I - 1$ intermediary solutions.

5 Computational results

ECS, GRACS and GRASP were coded in ANSI C and were run on Intel AMD (1.33 GHz) platform. The most important performance parameters were set as follows: for ECS, $20 \leq \mathcal{NC} \leq 30$ and $300 \leq |P| \leq 500$; for GRACS and GRASP, $\alpha = 0.10$ and $l = 0.7$. For each instance, were performed 20 trials, allowing the approaches to perform a maximum number of objective function calls. These parameter values try to make the tuning for the algorithm speed-accuracy trade-off and they were chosen through the authors' expertise.

ECS, GRACS and GRASP are now compared against the Parallel Memetic Algorithm (PMA)[16]. Besides a parallel algorithm, employing a suitable migration policy, PMA presents a new 2-swap local search with a reduction scheme, which discards useless swaps, avoiding unnecessary objective function calls. Its results were considered so far the best ones obtained in the literature, specifically with large GMLP instances [16].

The Table 2 shows the comparison between all *CS approaches and PMA. For the latter, the results were obtained in 10 trials[16]. GRASP was included for verifying a probable improvement by the clustering process, since GRACS is a modified GRASP. The success rate (SR) to reach the best known solution as well as the average of the number of objective function calls (FC) were considered for measuring the algorithm performances. For each tested instance, the result in bold shows the winning approach. In 3 of 5 instances, at least one of the *CS approaches was better than PMA. But in the particular contest between GRACS and GRASP, both performances were very similar. In $w4$ instance, GRASP has obtained better SR, but requiring more FCs. This fact evidences the need of better tuning GRACS, since GRASP/GRACS appear to be very promising approaches, with SR comparable with the best results found in the literature. For instance $w4$, the largest one found in literature, ECS has reached meaningful best results.

Table 2. Results of ECS, GRACS, GRASP and PMA for GMLP instances.

Inst.(IxJ)	ECS		GRACS		GRASP		PMA	
	SR(%)	FC	SR(%)	FC	SR(%)	FC	SR(%)	FC
x0 (48x40)	100	119296.0	100	39187.8	**100**	**24662.6**	100	43033.0
v4470 (47x37)	60	169136.0	**100**	**90459.8**	100	98081.0	60	176631.0
w2 (33x48)	100	26185.0	100	10002.6	100	12580.7	**100**	**3523.0**
w3 (70x84)	50	540893.0	90	372021.0	90	360225.4	**90**	**203892.0**
w4 (141x202)	**55**	**1695924.0**	20	2357496.0	30	3998868.0	20	9428591.0

6 Conclusion

This paper proposes a new way of detecting promising search areas based on clustering: the Clustering Search (*CS). Together with other search metaheuristics, working as full-time solution generators, *CS attempts to locate promising

search areas by solution clustering. The clusters work as sliding windows, framing the search areas and giving a reference point to problem-specific local search procedures, besides an iterative process, called assimilation.

Two metaheuristics based on *CS were also proposed for large scale GMLP instances: a GRACS (Greedy Randomized Adaptive Clustering Search) and an Evolutionary Clustering Search ECS. In comparison against the best results found in the literature, *CS approaches have achieved similar and sometimes superior performance. However, in the particular contest between GRACS and GRASP, both performances were very similar, evidencing the need of further tuning GRACS. Besides, to combine clustering with other metaheuristics as Immune Systems and Evolution Strategies will be considered in further research.

References

1. Moscato, P.: Memetic algorithms: a short introduction. In: Corne, D.; Dorigo, M.; Glover, F. (eds) New Ideas in Optimization. London: McGraw-Hill (1999) 219-234
2. Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded memetic algorithms with crossover hill-climbing, Evol. Computation (2004) 12(3):273-302
3. Jelasity, M., Ortigosa, P., García, I.: UEGO, an Abstract Clustering Technique for Multimodal Global Optimization, Journal of Heuristics (2001) 7(3):215-233
4. Glover, F.: A template for scatter search and path relinking. Selected Papers from the Third European Conference on Artificial Evolution. Springer-Verlag (1998) 3-54
5. Goldberg, D.E.: Genetic algorithms in search, optimisation and machine learning. Addison-Wesley (1989)
6. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization (1995) 6:109-133
7. Oliveira, A.C.M, Lorena, L.A.N.: Detecting promising areas by evolutionary clustering search. In: SBIA 2004, 17, (LNAI) v. 3171 (2004) 385-394
8. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research(1997) 24:1097-1100
9. Chaves, A.A., Lorena, L.A.N.: Hybrid algorithms with detection of promising areas for the prize collecting travelling salesman problem. In: HIS 2005, 5 (2005) 49-54
10. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics(2000) 39:653-684
11. Fink, A., Voss, S.: Applications of modern heuristic search methods to pattern sequencing problems, Computers and Operations Research (1999) 26(1):17-34
12. Linhares, A.: Industrial pattern sequencing problems: some complexity results and new local search models. Doctoral Thesis, INPE, S. José dos Campos, Brazil (2002)
13. Möhring, R.: Graph problems related to gate matrix layout and PLA folding, *Computing* (1990) 7:17-51
14. Golumbic, M.: Algorithmic graph theory and perfect graphs. *Academic Press*, New York (1980)
15. Syswerda, G.: Schedule optimization using genetic algorithms. *Handbook of Genetic Algorithms*,Van Nostrand Reinhold, New York (1991) 332-349
16. Mendes, A., Linhares, A.: A multiple population evolutionary approach to gate matrix layout, Systems Science, Taylor & Francis (2004) 35(1):13-23
17. Oliveira A.C.M., Lorena, L.A.N.: A Constructive Genetic Algorithm for Gate Matrix Layout Problems. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (2002) 21(8):969-974