Population Training Heuristics

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

[1] Universidade Federal do Maranhão - UFMA, Depto. de Informática, S. Luís MA, Brasil
acmo@deinf.ufma.br
[2] Instituto Nacional de Pesquisas Espaciais - INPE, Lab. Associado de Computação e Matemática Aplicada, S. José dos Campos SP, Brasil
lorena@lac.inpe.br

Abstract. This work describes a new way of employing problem-specific heuristics to improve evolutionary algorithms: the Population Training Heuristic (PTH). The PTH employs heuristics in fitness definition, guiding the population to settle down in search areas where the individuals can not be improved by such heuristics. Some new theoretical improvements not present in early algorithms are now introduced. An application for pattern sequencing problems is examined with new improved computational results. The method is also compared against other approaches, using benchmark instances taken from the literature.

Keywords: Hybrid evolutionary algorithms; population training; MOSP; GMLP.

1 Introduction

Evolutionary algorithms are efficient to explore a wide search space, converging quickly to local minima. However, their lack of exploiting local information is a well-known drawback to reach global minima. Evolutionary operators to incorporate knowledge about problem particularities have encapsulated heuristics and local search procedures. Such procedures basically consist in searching for better solutions in the set of candidate solutions (*neighborhood*) that can be obtained from a given solution by heuristic moves. An individual improved by heuristic, in general, is replaced as soon as a better individual is obtained. The more individuals are heuristically improved, the more the heuristic leads the population to incorporate the desired features.

Due to the computational cost of some heuristic procedures, a challenge in such hybrid methods is to define efficient strategies to cover all search space, applying local search only in actually promising neighborhoods. Elitism plays an important role towards achieving this goal, once the best individuals can represent such promising neighborhoods. But the elite can be sharing the same few neighborhoods and then the heuristic moves does not improve the population.

The Population Training Heuristic (PTH) proposes a way of leading the population to acquire desired characteristics. All individuals are evaluated by two

G.R. Raidl and J. Gottlieb (Eds.): EvoCOP 2005, LNCS 3448, pp. 166–176, 2005.

functions: a function computing *what the individual is* and another estimating *what it should be*. The later does not take account of a presumed potential of the individual, but its deficiency by not being what it should be.

The evolutionary process is driven by a problem-specific heuristic (called training heuristic), employed in the fitness function formulation. The *well-adapted* individuals are those who can not be improved by the training heuristic. They are *what they should be*, i.e., the best inside the neighborhood generated by the training heuristic and tend to stay in the population longer times.

The Constructive Genetic Algorithm (CGA) was proposed in [1] to Location Problems, and applied to other problems as Timetabling [2]. The CGA presents a number of new features compared to a traditional genetic algorithm, such as a dynamic-sized population composed of *schemata* (incomplete solutions) and *structures* (complete solutions). Each individual (structure or schema) has a fitness evaluation based on two functions, f and g (*fg-fitness*), that are built considering specific aspects of the problem at hand in a way that an individual with $|f - g| \approx 0$ corresponds to an optimal solution. A further optimization objective is introduced to guide the search to find structures: to maximize g. Thus, no matter the nature of the problem (minimization or maximization), the original problem is transformed in a bi-objective problem (BOP): $f - g$ minimization (optimal phase) and g maximization (constructive phase) [1].

The CGA was inspiration to PTH, especially by the double evaluation of individuals. In PTH, the *fg-fitness* leads population to subsearch spaces where no improvement can be reached by applying the training heuristic, probably optimal solutions depending on how much less *approximative* is the heuristic. To avoid costly fitness evaluation, light heuristics are used. Local search mutations are included in the evolutionary process to make a fine tuning of the well-adapted individuals.

Some early CGA applications, since they employ training heuristics, can be considered as based on PTH fundamentals [3], [4]. Such applications rank the population by a *constructive ranking* that considers simultaneously a constructive and optimal phases, as in the CGA original form. The constructive approach of PTH is still called Constructive Genetic Algorithm, but denoted by CGA^H, to avoid misunderstanding with the original CGA, which no heuristic training had been employed yet [1], [2].

On the other hand, the *non-constructive ranking*, proposed in this work, only focuses the optimal phase, concerning to the adaptation of individuals to training heuristic. The main reason for this alternative form of ranking is to aggregate more flexibility to the approach, not necessarily coding individuals as incomplete solutions. The non-constructive approach is called Population Training Algorithm (PTA) and it was firstly applied to numerical optimization [5].

This work introduces theoretical improvements for PTH and also presents new results for pattern sequencing problems not found in early works [3], [4], [5]. The remainder of this paper is organized as follows. Section 2 presents the general guidelines of PTH, consolidating the formulation needed to future applications. In section 3, an application in pattern sequencing problems is modeled. The conclusions are summarized in the section 4.

2 General Guidelines of PTH

The PTH can be defined by the tuple $\{P, \Theta, f, H, \wp, \delta\}$, where P is a population sampled from the coded search space S, hence individuals $s_k \in P \subset \mathsf{S}$. Individuals are generated by a set of evolutionary specific operators Θ and evaluated by an objective function f that maps S in $\Re$. The training heuristic H is defined by the pair $\{\varphi^H, g\}$, where g heuristically evaluates each solution generated by the neighborhood relationship φ^H. The neighborhood relationship φ^H can be understood as set of solutions which can be obtained from an original one, s_k, by heuristic moves (H moves):

$$\varphi^H(s_k) = \{s_k, s_{v1}, s_{v2}, \ldots, s_{vl}\} \tag{1}$$

where $l + 1$ is the length of the neighborhood of s_k, including itself.

The heuristic knowledge about a problem is then used to define g. A typical g, adopted in this work, is the objective function f calculated over $\varphi^H(s_k)$. Thus, for minimization problems:

$$g(s_k) = f(s_{vb}) = \min\{f(s_k), f(s_{v1}), f(s_{v2}), \ldots, f(s_{vl})\} \tag{2}$$

The best neighbor of s_k is denoted by s_{vb}. The concept of proximity, $\wp$, is concerned with the effort necessary to reach s_{vb} from s_k by H moves. More proximity means more adaptation of s_k to the heuristic that generated s_{vb}. Depending on the coding being employed in the application, some distance metrics may be used, such as hamming for binary-coded, euclidean for real-coded and heuristic distance [6]. In this application, the fitness distance between s_k and s_{vb} is adopted:

$$\wp(s_k, s_{vb}) = |f(s_k) - f(s_{vb})| \tag{3}$$

Independently of the nature of the problem (minimization or maximization), the $\wp(s_k, s_{vb})$, adopted in this work, always reflects how much s_{vb} is better than s_k. However, if another distance metric was used, $\wp(s_k, s_{vb})$ would mean just the adaptation to the training heuristic. Finally, the entire population is ranked by a function δ that considers the overall individual adaptation. The constructive and non-constructive rankings are, respectively:

$$\delta_{cons}(s_k) = \frac{d \cdot G_{max} - |f(s_k) - g(s_k)|}{d \cdot [G_{max} - g(s_k)]} \tag{4}$$

$$\delta_{ncons}(s_k) = d \cdot [G_{max} - g(s_k)] - |f(s_k) - g(s_k)| \tag{5}$$

Considering G_{max} an estimate of the upper bound for all possible values of the function g (even function f), the interval $G_{max} - g(s_k)$, gives the fitness distance between individual s_k and the upper bound. This distance is used in two distinct ways. In the constructive ranking, to estimate the completeness of the individuals, penalizing the schemata. In the non-constructive ranking, such

interval just considers the objective function evaluation, once in minimization problems, the greater is $G_{max} - g(s_k)$, the better is the individual.

The constant d is used to equilibrate both ranking equations (generally, about $1/G_{max}$). In the beginning of the evolution, the upper bound G_{max} can be analytically calculated, considering the problem instance, or estimated by sampling. For maximization problems, a lower bound G_{min} is introduced in the constructive ranking:

$$\delta^{max}_{ncons}(s_k) = d \cdot [g(s_k) - G_{min}] - |f(s_k) - g(s_k)| \tag{6}$$

The constructive ranking considers simultaneously a constructive and optimal phases, as in the CGA original form. The non-constructive ranking, on the other hand, only focuses the optimal phase. The main reason for these two forms of ranking is to aggregate more flexibility to the approach, as the possibility of employing distinct heuristics, evolutionary operators and, especially, other solution codings: incomplete solutions (schemata) not always may be naturally incorporated by the evolutionary process. A good example of such applications is the numerical optimization coded in real numbers [5].

3 Applications in Pattern Sequencing Problems

The problems treated in this section can be classified as pattern sequencing problems. Pattern sequencing problems may be stated by a matrix with integer elements where the objective is to find a permutation of rows or patterns (client orders, or gates in a VLSI circuit, or cutting patterns) minimizing some objective function [7]. 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.

The PTA is modeled for two similar pattern sequencing problems found in the literature: Minimization of Open Stacks Problem (MOSP) and Gate Matrix Layout Problem (GMLP). Theoretical aspects are basically the same for both problems. The difference between them resides only in their enunciation. A more detailed description of the MOSP is emphasized in next sections. The particularities of the GMLP are occasionally presented, when needed.

3.1 Theoretical Issues of the MOSP

The MOSP appears in a variety of industrial sequencing settings, where distinct patterns need to be cut and each one may contain a combination of piece types. For example, consider an industry of woodcut where pieces of different sizes are cut of big foils. Pieces of equal sizes are heaped in a single stack that stays *opened* until the last piece of the same size is cut. A stack is considered *opened* while there exist pieces of the same size to be cut. A MOSP consists of determining a sequence of cut patterns that minimizes the maximum of open stacks (MOS)

during the cutting process. Typically, this problem is due to limitations of physical space, so that the accumulation of stacks can cause the temporary need of removal of one or other stack, delaying the whole process.

The data for a MOSP are given by an $I \times J$ binary matrix P, representing patterns (rows) and pieces (columns), where $P_{ij} = 1$, if pattern i contains piece j, and $P_{ij} = 0$ otherwise. Patterns are processed sequentially, opening stacks whenever new piece types are processed and closing stacks of pieces that do not have any items else to be cut. The sequence of patterns being processed determines the number of stacks that stays open at same time. Another binary matrix, here called the open stack matrix Q, can be used to calculate the MOS for a certain pattern permutation. It is derived from the input matrix P, by the following rules:

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

where $\pi(b)$ is the position of pattern b in the permutation.

The Q shows the consecutive-ones property [8] applied to 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. Figure 1 shows an example of matrix P, its corresponding matrix Q, and the number of open stacks to each pattern processed. When pattern 1 is cut, 3 stacks are opened. No stack else is opened for patterns 2 and 3, but pattern 4 requires 5 open stacks. At most, 7 stacks ($MOS = \max\{3, 3, 3, 5, 6, 7, 7, 5, 3\} = 7$) are needed to process the permutation $\pi_0 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

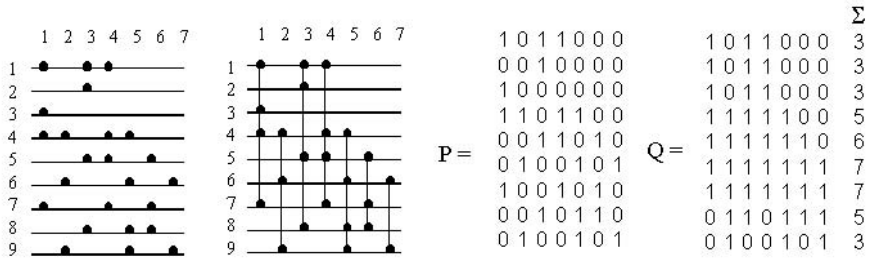

Fig. 1. MOSP (or GLMP) instance: original and corresponding matrix

Recently, several aspects of the MOSP and other related problems, as the GMLP, have been presented, including the NP-hardness of them [9], [10], [11]. The GMLP goal is to arrange a set of gates (horizontal wires), which are interconnected by nets (vertical wires), such that the number of tracks is minimized. This can be achieved by placing non-overlapping nets in the same track. The same example of Figure 1 can be seen as a GMLP instance. The number of open stack is equivalent to the number of overlapping nets.

3.2 PTA Modeling

A very simple representation is implemented for the MOSP and GMLP: a direct alphabet of symbols (natural numbers) represents the pattern (or gate) permutation. Each label is associated to a row of binary numbers, representing the piece type presence in each pattern. A permutation of rows is called structure and consists of a candidate solution for an instance.

A second objective for the MOSP have been used by early works: to close the stacks as soon as possible, allowing that the customer's requests be available with minimum delay [11], [3]. The second objective is to minimize the time that the stacks stay open (TOS) and it can be calculated by the sum of all "1" 's in Q. The TOS is particularly useful for increasing the fitness distinction among individuals. The function f reflects the total cost of a given permutation and considers the primary (MOS) and secondary (TOS) objectives:

$$f(s_k) = I \cdot J \cdot \max_{i \in I} \left\{ \sum_{j \in J} Q_{ij} \right\} + \sum_{i \in I} \sum_{j \in J} Q_{ij} \tag{7}$$

where the product $I \cdot J$ is a weight to reinforce MOS part of the cost.

A dynamic-sized population was implemented and controlled by an adaptive rejection threshold that eliminates the ill-adapted individuals, i.e., structures such that $\alpha \geq \delta(s_k)$. The adaptive rejection threshold, α, is initialized at the beginning of the process with the rank of the worst individual in population. During the evolutionary process, α is updated with adaptive increments, considering the current range of the rank values in the population, the population size, and the remaining number of generations. The adaptive increment of α is:

$$\alpha = \alpha + Step \cdot |P| \cdot \frac{(\delta_{bst} - \delta_{wst})}{RG} \tag{8}$$

where δ_{bst} and δ_{wst} are, respectively, the best and the worst rank of structures in current population, $|P|$ is the current population size, RG is the remaining number of generations, and $Step$ is an adjustment parameter, used to give more or less speed to the evolutionary process.

At the beginning, the population tends to grow up, generally, accepting all new individuals. After some generations, α determines the adaptation values that can be kept in population and the ill-adapted individuals are eliminated. Whenever no improvement is obtained, the population eventually can collapse, becoming empty. Therefore, the correct adjustment of $Step$ (generally, a value about 0.001 is used) is needed to avoid premature emptying of the population.

3.3 The Training Heuristics

The 2-Opt is a well-known improvement heuristic based on k-changes over a complete initial solution. Typically, a 2-change of a permutation consists of deleting 2 edges and replacing them by 2 other edges to form a new permutation. It can be obtained by breaking the permutation in 2 reference points and inverting the

order in the middle subpermutation. For example, $\{1-2-3-4-5-6-7-8\}$, breaking in 3 and 6 becomes $\{1-2-3-6-5-4-7-8\}$.

The 2-Opt-like heuristic is employed in function g for training the population. Each one of the 2-changes generates a neighbor structure that is evaluated, looking for the best objective function value. At the end, up to $0.5 \cdot (I^2 - I)$ neighbor structures are evaluated .

Another heuristic used in this work, called the Faggioli-Bentivoglio's heuristic, is based on the constructive heuristic described in [12]. The basic idea of this heuristic is to build a complete solution, minimizing the differences among the patterns. An initial group of patterns (in this work, the first $N/2$ patterns), in a given structure, is accepted as start patterns. The neighborhood is defined as all structures that begin with the start group of patterns and minimize the difference to the subsequent patterns, according to a three stage criterion.

At the first stage, the patterns that open as few new stacks as possible are chosen. A stack is opened when the new sequenced pattern contains a piece type that is not yet stacked, i.e., the i_{th} item presents a $0-1$ transition, from previous pattern to next. At the second stage, the pattern that removes the greatest number of stack is chosen among the patterns previously selected. A stack is removed when the new sequenced pattern ends a piece type that is being stacked, i.e., the i_{th} item presents a $1-0$ transition, from previous pattern to next. At the last stage, the pattern that continues the production of the greatest number of stacked pieces is chosen among the patterns previously selected. The production continues when the new sequenced pattern contains a piece type that is already stacked, i.e., the i_{th} item presents a $1-1$ transition, from previous pattern to next. If these three rules lead to more than one pattern to be inserted in sequence, one of them is selected at random.

3.4 Evolutionary Operators

The structures in the population are kept in descending order, according to the ranking in equation 5. Thus, well-adapted individuals appear in first places on the population, being privileged for evolutionary operations.

Two structures are selected for recombination. The first is called the base (s_{base}) and it is randomly selected out from the first positions in the population (generally, about 20% from the population). The second structure is called the guide (s_{guide}) and is randomly selected out of the entire population. They are recombined by a variant of the Order Crossover (OX) called Block Order Crossover (BOX) [13]. 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 (Figure 2a).

A local search mutation is applied to each new structure generated with a certain probability (generally, about 20%). This procedure is very important to get intensification moves around a solution. The local search mutation explores a search tree, considering several 2-Opt neighborhoods, not only one as fitness function g (Figure 2b).

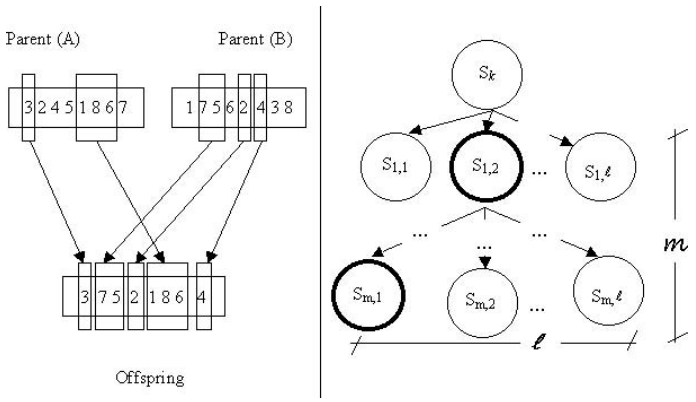

Fig. 2. (a) Block order crossover and (b) local search mutation tree

Some neighbors (generally, $l = 20$ neighbors) are evaluated in each tree level and the best one is held on to be used as starting point to next tree level. Successive neighborhoods are generated until a pre-defined maximum number of neighborhoods (generally, $m = 20$ neighborhoods).

The PTA pseudo-code, shown as follows, is based on traditional genetic algorithms. The stop criteria is either when the best-known solution is found, or after a certain number of objective function calls, to be set depending on the number of patterns (length) of the instance at hand. Once the stop criteria is reached the searching process has failed to find the best-known (optimal) solution.

```
{PTA pseudo-code}
RandomlyInitialize (P_α);
for all s_k ∈ P_α do
  Compute f(s_k), g(s_k), δ(s_k); {equations 7, 2, 5}
end for
α := δ_wst;
while not stopCriteria do
  while numberOfCrossovers do
    SelectionBaseGuide (s_base, s_guide);
    CrossoverBOX (s_base, s_guide) giving s_new;
    if mutationCondition then
      LocalSearchMutation (s_new);
    end if
    Compute f(s_new), g(s_new), δ(s_new);
    Update (s_new) in P_α;
  end while
  α := AdaptiveIncrement (α); {equation 8}
  for all s_k ∈ P_α e δ(s_k) < α do
    Delete (s_k) from P_α;
  end for
end while
```

3.5 Computational Tests

A pool of 300 MOSP instances and one GMLP instance were chosen for tests, taken from [11], [12]. The MOSP instances have different number of patterns ($I \in \{10, 15, 20, 25, 30, 40\}$), each one of them with different number of piece types ($J \in \{10, 20, 30, 40, 50\}$). The GMLP instance, called *w4*, has 141 gates (patterns) and 202 nets (piece types) and it is the largest instance of pattern sequencing found in literature [11].

The PTA population size was 50 for the MOSP instances and 100 for GMLP instance. Two versions of PTA (PTA^{2opt} and PTA^{fag}), respectively using the 2-Opt-like and Faggioli-Bentivoglio's heuristics for training, were built to evaluate how different heuristics interfere in the algorithm performance.

The best two approaches presented in [12]: a) a tabu search method (TS) based on an optimized move selection process; and b) a generalized local search method (GLS) that employs the Faggioli-Bentivoglio's procedure in a simplified tabu search that only accepts improving moves [12]. Besides, another two methods are included in this comparison: c) the Constructive Genetic Algorithm (CGA^{2opt}); and d) the Collective method (COL). The CGA^{2opt} employs 2-Opt-like heuristic as training heuristic [3], [4]. The COL method explores distance measures among permutations and employs 2-exchange local search to drive the search of a simulated annealing like algorithm [11].

Table 1 shows the best MOS averages obtained by the methods in each instance group. Each pair (I, J) is an instance group with ten instances and different solutions. A comparison is made putting together the results shown in [11], [12] and the new results found by the PTH approaches in this work.

Both versions PTA^{fag} and PTA^{2opt} found the same results and were referred as PTA. The TOS is not considered in the other works and was excluded from the comparison. For this test, CGA^{2opt} and PTA were run 10 times. The PTA have found the best overall average of solutions for the instance groups. The CGA^{2opt} appears with the second best performance, failing in achieving the best average in 2 instance groups (20×40 and 25×40). The hardest instances to find the best-known solution were *p2040n6*, *p2540n3*.

Other performance aspects are focused in Table 2: average of the MOS found (AS), the number of times that the best solution was found (NS), the average of

Table 1. Performance comparison with another approaches per instance groups

I	J	COL	TS	GLS	CGA	PTA	I	J	COL	TS	GLS	CGA	PTA	I	J	COL	TS	GLS	CGA	PTA
10	10	5.5	5.5	5.5	5.5	5.5	20	10	7.5	7.7	7.5	7.5	7.5	30	10	7.8	7.8	7.8	7.8	7.8
	20	6.2	6.2	6.2	6.2	6.2		20	8.5	8.7	8.6	8.5	8.5		20	11.2	11.2	11.2	11.1	11.1
	30	6.1	6.1	6.2	6.1	6.1		30	9.0	9.2	8.9	8.8	8.8		30	12.2	12.6	12.2	12.2	12.2
	40	7.7	7.7	7.7	7.7	7.7		40	8.6	8.6	8.7	8.6	**8.5**		40	12.1	12.6	12.4	12.1	12.1
	50	8.2	8.2	8.2	8.2	8.2		50	7.9	8.0	8.2	7.9	7.9		50	11.2	12.0	11.8	11.2	11.2
15	10	6.6	6.6	6.6	6.6	6.6	25	10	8.0	8.0	8.0	8.0	8.0	40	10	8.4	8.4	8.4	8.4	8.4
	20	7.2	7.2	7.5	7.2	7.2		20	9.8	9.8	9.9	9.8	9.8		20	13.0	13.1	13.1	13.0	13.0
	30	7.3	7.4	7.6	7.3	7.3		30	10.6	10.7	10.6	10.5	10.5		30	14.5	14.7	14.6	14.5	14.5
	40	7.2	7.3	7.4	7.2	7.2		40	10.4	10.7	10.6	10.4	**10.3**		40	15.0	15.3	15.3	14.9	14.9
	50	7.4	7.6	7.6	7.4	7.4		50	10.0	10.1	10.2	10.0	10.0		50	14.6	15.3	14.9	14.6	14.6

objective function calls (FC). A parallel memetic algorithm (PMA) taken from [14] was included in this comparison. The PMA presents a new 2-exchange local search with a reduction scheme, which discards useless swaps, avoiding unnecessary objective function calls. Table 2 shows the comparison among PTA^{fag}, PTA^{2opt}, CGA^{2opt} and PMA in 10 trials for GMLP instance *w4*.

Table 2. Comparison between PTA and CGA^{2opt} and PMA for instance *w4*

	AS	NS	FC		AS	NS	FC
PTA^{2opt}	28.6	2	8,488,438	CGA^{2opt}	28.0	3	6,537,706
PTA^{fag}	28.3	2	9,330,802	PMA	29.4	2	9,428,591

Observing Table 2, CGA^{2opt} has obtained the best AS and NS for *w4*. PTA^{fag} and PTA^{2opt} are slightly similar in AS, but the later seems to perform less function calls. All approaches based on population training were better than PMA. Despite the Faggioli-Bentivoglio's procedure seemingly should perform less function calls than 2-Opt-like heuristic, this can not be observed in FC. Indeed, it was expected a superior FC for versions employing 2-Opt. This fact can be explained perhaps by the mutation procedure: the mutation would dominate the number of function calls and the training heuristic was not relevant for FC. Another possibility is that 2-Opt-like training heuristic would improve the algorithm performance so that it could compensate its computational cost.

The comparison among the methods here presented are based only in the average performance because the other works found in literature does not mention nothing about the variability of their models. Table 3 shows the average and standard desviation in 20 trials for the hard MOSP instances *p2040n6*, *p2540n3*.

Table 3. Average and standard desviation for MOSP instances *p2040n6* and *p2540n3*

Instances (solution)	$\mathcal{PTA}^{2opt}$	$\mathcal{PTA}^{fag}$	$\mathcal{CGA}^{2opt}$	$\mathcal{CGA}^{fag}$
p2040n6 (8, 0)	$8,7 \pm 0,5$	$8,7 \pm 0,5$	$8,9 \pm 0,3$	$8,9 \pm 0,3$
p2540n3 (10, 0)	$10,7 \pm 0,5$	$10,8 \pm 0,4$	$10,7 \pm 0,5$	$10,9 \pm 0,3$

Statistical tests showed that the differences in averages are not statisticaly significant for MOSP instances *p2040n6* and *p2540n3*. For GMLP instance *w4*, averages obtained by CGA^{2opt} was significantly better than those obtained by PTA^{fag} and PTA^{2opt}.

4 Conclusion

In the Population Training Heuristic (PTH), proposed in this paper, the evolutionary process is driven by a training heuristic, employed in the fitness definition. The population is led to settle down in search areas where the individuals can not be improved by such heuristic. In this work, the general guidelines for

PTH are introduced and new versions employing a non-constructive ranking are presented.

The algorithms based on PTH showed the best performance when compared against other approaches found in literature. Both constructive and non-constructive approaches were able to reach the known optimal solutions. The 2-Opt-like training heuristic has presented better results concerning the computational cost. For further work, it is intended to implement a multi-heuristic version with subpopulations trained by different heuristics, evolving in parallel, for multi-objective problems.

References

1. Lorena, L.A.N., Furtado, J.C.: Constructive genetic algorithm for clustering problems. *Evolutionary Computation.* (2001) 9(3): 309-327.
2. Ribeiro Filho, G., Lorena, L.A.N.: A constructive evolutionary approach to school timetabling, In: Applications of Evolutionary Computing, Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H., (Eds.) - *Springer LNCS 2037*(2001). 130-139.
3. Oliveira, A.C.M., Lorena, L.A.N.: A constructive genetic algorithm for gate matrix layout problems. *IEEE Trans. on Computer-Aided Designed of Integrated Circuits and Systems*(2002) 21(8): 969-974.
4. Oliveira, A.C.M., Lorena, L.A.N.: 2-Opt population training for minimization of open stack problem. Advances in Artificial Intelligence - XVI Brazilian Symposium on Artificial Intelligence. Guilherme Bittencourt e Geber L. Ramalho (Eds). Springer LNAI 2507. (2002) 313-323.
5. Oliveira, A.C.M., Lorena, L.A.N.: Real-coded evolutionary approaches to unconstrained numerical optimization. Advances in Logic, Artificial Intelligence and Robotics. Jair Minoro Abe and João I. da Silva Filho (Eds). (2002) 10-15.
6. Reeves, C.R. Landscapes, operators and heuristic search. Annals of Operations Research, v. 86, p. 473490, 1999.
7. Fink, A., Voss, S.: Applications of modern heuristic search methods to pattern sequencing problems, Computers and Operations Research, (1999) 26(1): 17-34.
8. Golumbic, M.: Algorithmic graph theory and perfect graphs. *Academic Press*, New York (1980).
9. Möhring, R.: Graph problems related to gate matrix layout and PLA folding, *Computing* (1990) 7: 17-51.
10. Kashiwabara, T., Fujisawa, T.: NP-Completeness of the problem of finding a minimum clique number interval graph containing a given graph as a subgraph, *In Proc. Symposium of Circuits and Systems*(1979).
11. Linhares, A.: Industrial pattern sequencing problems: some complexity results and new local search models. Doctoral Thesis, INPE, S. José dos Campos, Brazil (2002).
12. Faggioli, E., Bentivoglio, C.A.: Heuristic and exact methods for the cutting sequencing problem, *European Journal of Operational Research*(1998) 110: 564-575.
13. Syswerda, G.: Schedule optimization using genetic algorithms. *Handbook of Genetic Algorithms*,Van Nostrand Reinhold, New York (1991) 332-349.
14. Mendes, A., Linhares, A.: A multiple population evolutionary approach to gate matrix layout, Int. Journal of Systems Science, Taylor & Francis Eds, (2004), 35(1): 13-23.