Memoirs of the Scientific Sections of the Romanian Academy
Tome XXXIV, 2011

COMPUTER SCIENCE

GRAFTING A BRANCH AND BOUND METHOD ON A GENETIC ALGORITHM FOR BALANCING I / U-SHAPED ASSEMBLY LINES

OCTAV BRUDARU[1, 2] and CRISTIAN ROTARU [3]

[1] Institute of Computer Science, Romanian Academy, Iaşi Branch, Carol I, 11
[2] "Gh. Asachi" Technical University of Iaşi, D. Mangeron 53,Romania
[3] "Al. I. Cuza" University of Iaşi, Faculty of Computer Science, Gen. Berthelot 5, Romania
Corresponding author: brudaru@tuiasi.ro

This paper presents a new efficient hybrid method that combines a branch & bound technique and a genetic algorithm for solving I/U -shaped assembly lines balancing. The basic components of a branch & bound technique that operates with topological sorting of the set of tasks are described. An order-based genetic algorithm that works with embryos that are either prefixes or pairs of prefixes - suffixes of topological sortings uses the components of the branch & bound technique and a newly introduced growing operator for evolving the population towards complete solutions. Sensitive lower bounds early predicting the quality of solutions built on the current embryos are used. Two strategies based on preventive and corrective management are proposed as a tradeoff between growing and survival selection. The results of experimental investigations show that the proposed embryonic genetic algorithm dominates the pure genetic algorithm. This is due to a better trade-off between the exploration that is mainly based on the growing population of embryos and the exploitation that is transferred to the evolving of the complete solutions.

*Key words***:** assembly line balancing, genetic algorithm, branch & bound, hybridization.

1. INTRODUCTION

Genetic algorithms (GA) [Mic94] have been applied on a variety of complex combinatorial optimization problems with great success. They can easily incorporate different goals and constraints and can be combined with other heuristic methods for producing powerful hybrid algorithms [Pas09]. Although rare, combining exact methods with GAs although rare is of great interest due to its potential performance gain coming from the two component techniques.

On the other hand, as a proof of their versatility and power, GAs are applied to a large set of difficult problems in the industrial [Pas09]. In this context, GAs are very promising in providing efficient and robust solving techniques for the assembly line balancing (ALB) objectives and restrictions ([Sch98], [Tas08], [BoyFS07],).

In this paper, the design of a new embryonic genetic algorithm for solving the "I" and "U"-shaped assembly lines balancing problems with deterministic times is

described. In this new design, the grafting is done with the branch & bound method (B&B) [BruS01], which is an exact technique successfully used in finding the optimum solutions for the basic types of ALB problem [Sch98]. Previous successful attempts to graft a B&B to a GA are described for ALB with fuzzy times [BruV04], optimal routing for delivery problem [BruV07] and *K*-repairmen problem [BruMD10]. This grafting is now applied to new ALB models, whereas the design contains new and efficient components.

The design starts from a B&B method from which the estimate function and the branching rules are used. An order-based genetic algorithm working with embryos that are either prefixes or pairs of prefixes - suffixes of topological sortings (assembly flows) uses the taken components of the B&B and a newly introduced growing operator for evolving the population towards complete solutions. Lower bounds early predicting of the quality of solutions built on the current embryos are proposed to work as fitness functions both for I- and U-shaped lines. Growing and survival schemes are proposed for preventing the elimination of long embryos with more accurate but not small enough fitness values. The results of experimental investigations made with classical test data for evaluating the quality of computed solutions are presented. The comparison of the performance of the embryonic genetic algorithm with those of the classical variant indicates the better behavior of the first one.

The remaining of the paper is organized as follows. Section 2 formulates the ALB variants that can be solved by the embryonic algorithm. Section 3 presents the basic components of the B&B methods that are taken for the hybridization. The main components of hybrid GA are described in section 4. Different strategies for growing and maturation of the embryos are discussed in section 5. The results of the experimental investigation are presented in section 6. Last section summarizes the work and suggests further developments.

2. FORMULATING OF THE ALB VARIANTS

Let us denote by $V = \{1,2,\ldots n\}$ the set of tasks of the assembly process and consider the set of ordered pairs $A \subset V \times V$, where $(i, j) \in A$ means that task i must be completed before task j begins. Consider the acyclic digraph $G = (V, A)$ where V is the set of

vertices and A is the set of arcs. Let us denote by $t_i > 0$ the execution time of task i. The time C elapsed between two successive outputs from the line is the cycle time. If only serial workstations have to by used, then the condition $t_i \le C$ holds for $i = 1,\ldots,n$. Let us denote by $W = \{W_1,\ldots,W_m\}$ a partition of V into workstations that satisfies the following conditions

$$\text{if } (x,y) \in A, x \in W_r, y \in W_s \text{ then } r \le s \qquad \textbf{(1)}$$

and

$$T(W_j) = \sum_{i \in W_j} t_i \le C, j = 1,2,\ldots,m. \qquad \textbf{(2)}$$

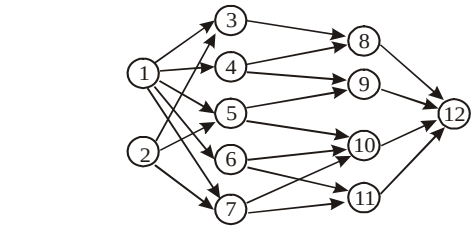
The simple ALB problem (SALB) is to find a partition W of V satisfying (1)-(2) and having a minimum number of workstations. It is NP-hard whilst its associated decision problem is NP-complete.

SALB usually concerns the I-layout of the workstations. To illustrate this, consider the assembly process whose processing times are given in Table 1.

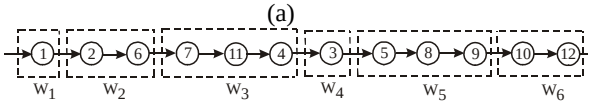
Table 1
The processing times of tasks

Task	1	2	3	4	5	6	7	8	9	10	11	12	C
Time	4	3	5	1	3	3	2	2	1	2	1	2	6

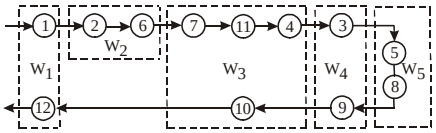
The precedence constraints shown in Figure 1(a) led to the feasible assembly flow (1,2,6,7,11,4,3,5,8,9,10,12). An example of I-shaped line is shown in Figure 1(b). In Figure 1(c), the same assembly flow is folded to form a U-shaped line.



(a)



(b)



(c)

Fig. 1. Precedence constraints (a), I-shaped line (b) and U-shaped line (c).

Formally, the U-shaped variant of SALB can be stated as follows. Consider a partition $W=\{W_1,...,W_m\}$ of V in workstations that verify the conditions:

$$W_j=\vec{W}_j\cup\overleftarrow{W}_j,\ \vec{W}_j\cap\overleftarrow{W}_j=\varnothing,\ j=1,...,m, \tag{3}$$

if $(x,y)\in A$, $x\in W_r$, $y\in W_s$ then just one of the following statements is true: (4)

(a) $x\in\vec{W}_r$ and $y\in\vec{W}_s$ implies $r\le s$,

(b) $x\in\overleftarrow{W}_r$ and $y\in\overleftarrow{W}_s$ implies $r\ge s$,

(c) $x\in\vec{W}_r$ and $y\in\overleftarrow{W}_s$,

(d) $x\in\overleftarrow{W}_r$ and $y\in\vec{W}_s$ is false.

and

$$T(W_j)=\sum_{h\in W_j}t_h\le C,\ j=1,...,m. \tag{5}$$

The U-shaped variant of SALB (shortly, USALB) requires the finding a partition that satisfies (3)-(5) and has a minimum number of workstations.

The variant of SALB with the so-called *compatibility* constraints (CALB) was introduced in [Bru92] as a formalism for operating in an unitary manner with the requirement that each station contains a limited number of types of equipment, requirement of tasks to be assigned to particular types of stations, the execution of some tasks in only a left (right)-of-line station, the association of tasks according with tasks skill level and the separation of some tasks. Consider a cover $K=\{K_1,...,K_p\}$ of V. A solution to CALB is a partition $W=\{W_1,\ldots,W_m\}$ of V that satisfies (1), (2) and

$$(\forall)j\in\{1,...,m\},\ (\exists)h_j\in\{1,...,p\}\ \text{so that}\ W_j\subseteq K_{h_j}. \tag{6}$$

CALB can be reformulated for U-shaped lines as well. Remark that $m_0=ceil(T_{tot}/C)$ is a lower bound for the number m of workstations in every solution to ALB/CALB, where $ceil(x)=\min\{y/y\ge x, y\ \text{integer}\}$ and $T_{tot}=\sum_{i=1}^{n}t_i$. If $m=m_0$ holds for some solution then this solution is optimal.

3. OUTLINE OF A B&B METHOD FOR ALB

The B&B approach requires that the search space be organized as a tree-shape structure, T. In T, a solution is uniquely associated with a path from the root to a terminal node in T. It is easy to see that the ALB solutions can be built on topological sortings (that are feasible assembly flows) of the precedence graph. In turn, the topological sortings can be organized as a tree T in which a path from the root to a leaf node corresponds to a topological sorting of G. The B&B technique progressively drives the search towards the optimal solution mainly exploring the most promising zones of search space, using an estimate function that predicts the optimal value of the objective function on the subset of solutions that can be obtained from the current partial solution. In the search space tree, B&B examines nonterminal/terminal nodes in order to reach a terminal node corresponding to a complete solution with the best estimate function value among the active nodes. From this perspective, both B&B and a classical GA act in the same state space tree T, but the difference is that GA exclusively explores the terminal nodes of this tree. The main components of B&B technique for ALB, namely branching rules, estimate function and the search strategy are outlined.

Branching rules. The tree T can be constructed using the classical algorithm for topological sorting [Knu76]: instead of choosing a random candidate from the set of viable vertices, these ones are considered "children" of the current node, and the procedure is repeated for each "child". Denote by $\mathbf{x} = (x_1, x_2, ..., x_k)$ a prefix of length k of a topological sorting. This prefix corresponds to a unique path from the root ρ of T to a node $\bar{\mathbf{x}}$ on the level k in T. Let $S(\mathbf{x})$ be the set of all topological sortings having the prefix $\mathbf{x}$. Clearly, (b1) $S(\rho) = \mathbf{S}$ ($\mathbf{S}$ is the set of all topological sortings) and (b3) $\text{card}(S(\mathbf{x})) = 1$ for $k = n$. Concerning the branching rule (b2), for $k < n$, let $\{y_1, ..., y_r\} \in V - \{x_1, ..., x_k\}$ be the vertices of G whose predecessors are included in $\mathbf{x}$. Then the prefixes $\mathbf{y}_j = (x_1, ..., x_k, y_j)$, $j = 1, ..., r$ correspond to the children $\overline{\mathbf{y_1}}, ..., \overline{\mathbf{y_r}}$ of $\bar{\mathbf{x}}$. Clearly, $S(\mathbf{x}) = S(\mathbf{y}_1) \cup ... S(\mathbf{y}_k)$ and $S(\mathbf{y}_i) \cap S(\mathbf{y}_j) = \varnothing, i \neq j$. Rules (b1-b3) are illustrated in Figure 2. Rule (b2) is used in section 4.5 to define the growing operator.

Estimation function. The estimate function value $e(\bar{\mathbf{x}})$ measures of the chance to find a good solution in $S(\mathbf{x})$, where $\bar{\mathbf{x}}$ is the node in T corresponding to $\mathbf{x}$. Since ALB

is a minimization problem, if f is the goal function (i.e. the number of workstation), the function e must satisfy (e1) $e(\bar{\mathbf{x}}) \le \min\{f(\mathbf{z}) / \mathbf{z} \in S(\mathbf{x})\}$ and (e2) $e(\bar{\mathbf{x}}) = f(\mathbf{z})$ for $k = n$ and $\{\mathbf{z}\} = S(\mathbf{x})$. For ALB, $e(\bar{\mathbf{x}})$ is a lower bound of the number of workstations of the solutions built on assembly flows with the prefix $\mathbf{x} = (x_1, x_2, ..., x_k)$. It is considered that the smaller $e(\bar{\mathbf{x}})$-value is, the higher the chances for finding a better solution in $S(\mathbf{x})$ are.

Search strategy. The estimate function e induces a priority relation between the subtrees that remain to be explored. If the *least cost* search strategy is adopted, then the nonterminal node with the minimum value of the estimate function among the active node list is branched and its children replace the parent in this list. If the minimum is reached for a terminal node then an optimal solution was found.

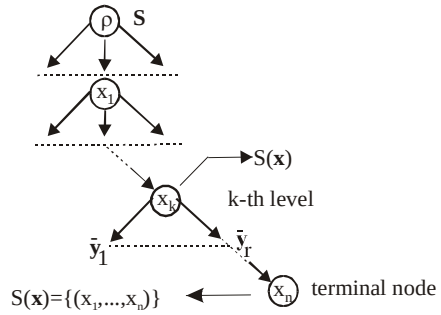


Fig. 2. Sample of the search space tree

4. BASIC COMPONENTS OF THE HYBRID GA

The main components of the hybrid algorithm resulted by grafting B&B technique on a GA (called BB_GA) and their interaction are presented below.

4.1. SOLUTION REPRESENTATION

Since the goal of ALB is to find topological sortings leading to optimal solutions, BB_GA mainly operates with *embryos*. In the case of I-shaped lines, a chromosome is a prefix $\mathbf{x} = (x_1, x_2, ..., x_k)$ of a topological sorting, where $1 \le k \le n$. If $k < n$ then x is called an embryo, whereas for $k = n$ x is an adult. For U-shaped lines, an embryo is a pair (prefix, suffix) of a topological sorting, $\mathbf{x} = (x_1, ..., x_k, *, x_h, ..., x_n)$, where $1 \le k < h \le n$, and $h - k - 1$ components are not defined. The star symbol denotes the absence of interior components. This notation can be used as well for a prefix $(x_1, ..., x_k, *)$ or a suffix $(*, x_h, ..., x_n)$. Remark that the construction of a suffix oriented state space tree is

similar to that of prefix oriented, whereas the set of solutions corresponding to $\mathbf{x} = (x_1,...,x_k,*,x_h,....,x_n)$ is given by

$$S((x_1,...,x_k,*,x_h,....,x_n)) = S((x_1,...,x_k,*)) \cap S((*,x_h,....,x_n))$$

and the branching rule means extending the specified part in $\mathbf{x}$ with exactly one position, i.e. either prefix or suffix expansion. This is illustrated in Figure 3, where the current node is branched by expanding either the prefix and or the suffix with the tasks $u_i, i = 1,..., p$ u and $v_j, j = 1,....,q$, respectively.

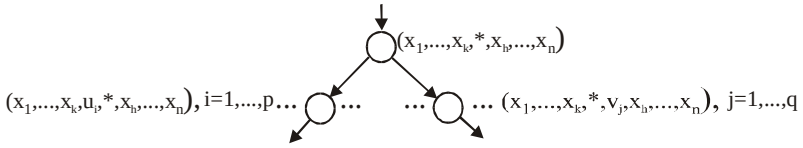


Fig. 3. Branching a node for an U-embryo.

Further, $(x_1,...,x_k,*)$ is called I-embryo whereas $(x_1,...,x_k,*,x_h,....,x_n)$ designates an U-embryo. A chromosome coded in this manner covers a set of solutions to the problem, consisting in all the topological sortings that, for I-shaped lines, begin with the prefix $(x_1,x_2,...,x_k,*)$, whereas for U-shaped they begin and end (with the prefix $(x_1,x_2,...,x_k,*)$ and the suffix $(*,x_h,....,x_n)$), respectively. In this way, BB_GA operates with sets of solutions instead of individual solutions. An early orientation of the search to the promising zones of the search space is expected.

4.2. INITIAL POPULATION

The initial population is randomly generated in the limit of fixed population size ps. Complete topological sortings are no longer generated. For I-embryos, the generated prefixes of the topological sortings have the lengths between two bounds, $h_{\min} = ceil(ph_1 \cdot n)$ and $h_{\max} = ceil(ph_2 \cdot n)$, where $0 < h_{\min} \le h_{\max} < n$, $ph_1 = 0.05$ and $ph_2 = 0.2$. The number of defined tasks is equally distributed between the prefix and suffix.

A prefix of a topological sorting is generated by cutting the activity of the classical algorithm [Knu76] after the desired size is reached. For generating U-embryos, a slightly modified version of the classical topological sorting algorithm was created in which two disjoint sets of available vertices are maintained: one containing vertices with none/already assigned predecessors and one with vertices with none/already assigned successors (a mirrored-like version). These two sets are the source of nodes assigned to the U-embryo.

4.3. FITNESS FUNCTION VIA ESTIMATE FUNCTION

Fitness function fit is applied to a chromosome $\mathbf{x}$ and returns a score $fit(\mathbf{x})$ indicating its the performance. It contains two terms m_c and m_e, where m_c is the number of workstations constructed on the partial assembly flow(s) contained in $\mathbf{x}$ and m_e, whereas m_e is the minimum number of workstations that must be allocated in order to accommodate the tasks non included in the evaluated I/U-embryo. As defined before, T_{tot} is the total execution time. Two cases are considered.

Fitness calculation for I-embryo
input: $\mathbf{x} = (x_1, x_2, ..., x_k, *)$ with $k < n$;
output: $fit(\mathbf{x})$;
// computing the term m_c
1. $m_c = 1$, $W_1 = \varnothing$
2. for $i = 1,...,k$ do:
if $T(W_{m_c} \cup \{x_i\}) \le C$
then $W_{m_c} = W_{m_c} \cup \{x_i\}$
else $m_c = m_c + 1$, $W_{m_c} = \{x_i\}$
// computing the term m_e
3. $\tau_r = T_{tot} - \sum_{j=1}^{m_c} T(W_j)$; // execution time of the tasks not in $\mathbf{x}$
4. $\tau_l = C - T(W_{m_c})$ // idle time in last workstation;
5. $m_e = (\tau_r - \tau_l)/C$;
6. $fit(\mathbf{x}) = m_c + m_e$;

Fitness calculation for U-embryo
input: $\mathbf{x} = (x_1, ..., x_k, *, x_h, ..., x_n)$, $k + (n - h + 1) < n$;
output: $fit(\mathbf{x})$;
1. A greedy method is used to serially create the workstations $W_1, ..., W_{m_c}$ with the tasks in $\mathbf{x} = (x_1, ..., x_k, *, x_h, ..., x_n)$. The prefix is consumed from left to right, whereas the suffix vice versa. One starts with $m_c = 1$ and $W_1 = 1$. The current step is an attempt to complete the current workstation. If exactly one task fits into the idle time of the current station, it is assigned to this station. If one task from the prefix and one from the suffix can fit, then the maximum time task is assigned. If no task fits then $m_c = m_c + 1$, $W_{m_c} = \varnothing$. This step is repeated until the prefix and suffix are consumed.
2. The term m_e is computed the same way as for I-embryo.
3. Finally, $fit(\mathbf{x}) = m_c + m_e$.

For CALB problem, the m_c value is computed by imposing condition (6) whenever an attempt is made for extending the current workstation with a new task. It is obvious that for $k = n$ it results $m_e = 0$ and $fit(\mathbf{x}) = m_c$ is the value of the fitness function for an

adult chromosome. In order to obtain a lower bound of the number of workstations of the solutions in $S((x_1,...,x_k,*))$, the estimation for the unknown part of the solution is calculated by assuming that the remaining tasks can be ideally distributed, that is W_{m_c} and all the next workstations have no idle time. Moreover, m_e is a real number for obtaining a more discriminating fitness.

4.4. GENETIC OPERATORS

The common attribute of genetic operators used for BB_GA is the topological sorting preserving property. Moreover, a new variant of the growing operator first described in [BruV04] is proposed. It has the role of increasing the size of embryos in order to obtain adult chromosomes and it corresponds to the branching operation specific to the B&B method.

Mutation. The mutation is applied to the entire population. Each individual can suffer a mutation with the probability p_m. Apropriate values for p_m are between 0.1 and 0.2. This operator is a dynamic variant of that described in [BruV04]. Let $\mathbf{x} = (x_1,...,x_k,*,x_h,.....,x_n)$ be the chromosome under mutation. By convention, it can be viewed as an I-embryo if $h = n+1$. A vertex x_j is randomly selected from the set $\{1,...,k-1,k,h,h+1,...,n\}$ and, in the same part of the embryo containing x_j (prefix /suffix), one computes the positions l_{left} and l_{right} of the rightmost predecessor and leftmost successor of x_h. The basic step of mutation is successful either $l_{left} < h-1$ or $l_{right} > h+1$. If so, x_h is inserted in a random position between l_{left} and l_{right} and the sequence between the old and new position of x_h shifts in opposite direction. Since this basic step does not produce enough change especially in long embryos or adults, it is recursively applied r times. For allowing a fine tuning toward the end of the evolution and for adapting the mutation to the length of embryo, r is given by $r = (a\,t + b)(k + n - h + 1)/n$, where $a = (1 - n/8)/(0.9\,t_{max} - 1)$, $b = n/8 - (1 - n/8)/(0.9\,t_{max} - 1)$, t is the current iteration number and t_{max} is maximum number of generations. The factor $(k + n - h + 1)/n$ ensures that the changes produced by mutation are proportional with the relative length of the embryo.

Crossover operator. The matting pool represents the best 40% -65% of the population. Crossover is applied with probability p_c. Each prefix (or prefix and suffix) of the two parents is randomly cut into a number of parts $np = ceil(\gamma[nc \cdot (1-\sigma) + m_0\sigma]/2)$, where γ is a random number in the range $[0.95,1.1]$, $nc = \max\{1, \lfloor \min\{k_j, n - h_j + 1 / k_j \neq 0, h_j \neq n+1, j = 1,2\} / n_{ws} \rfloor\}$, with $n_{ws} = C / \max_i t_i$ is a lower bound of number of tasks per workstation and $m_0 = \lceil T_{tot} / C \rceil$. The parameter σ varies from 0 to 1 as the number of the current generation goes from 1 to $t_{\max}+1$, where $t_{\max}$ is the maximum number of generations. The variant $\sigma(t) = (t-1)/(1.01t_{\max} - 1)$ is simple and works well. This variation of np is adopted for protecting the constructive blocks whose lengths increase during the evolution. One operates separately on each prefix/suffix. The offspring takes the first part from one parent, and then the parts of both parents are alternated so that each task appears just once. This crossover operator is suitable for ALB since it ensures the feasibility of the offspring. For each offspring, the prefix/suffix is the reunion of the prefixes/suffixes of the parents, thus crossover also acts as a growing operator.

5. STRATEGIES FOR MATURATION OF THE EMBRYOS

Different ways to combine the transforming the embryos into complete solutions with survival selection and the changing the evolution goals are discussed in this section.

5.1. GROWING OPERATOR

The growing operator transforms the embryos in adult chromosomes. Many growing strategies are possible and different mechanisms for creating a growing pressure can be used. For ALB problem, some growing steps on a given embryo can lead to visible effect on the fitness value because the number of workstations only increases with one if enough tasks were added to the last workstation. It was experimentally found that the most appropriate strategy consists in the introducing of ascending thresholds for the minimum accepted size of the embryos involved in the current evolution stage. This strategy can be given as a staircase function of the form $\lambda(t) = \min(\text{ceil}(\alpha t + \beta), n)$, where the slope $\alpha = \Delta m / \Delta t$ represents the increase of the number

of workstations for each successive Δt evolution stages. Before the ending of the evolution not later than $t_{\max}$, the BB_GA should have an interval $[t_a, t_{\max}]$ in which all the chromosomes are adult. Therefore, $\Delta m / \Delta t = (n - h_1)/(t_a - 1)$, where $h_1 = (h_{\min} + h_{\max})/2$ and one parameter can be expressed as a function of the remaining ones. Figure 4 illustrates this growing strategy with $t_a = 280$, $t_{\max} = 420$, $n = 40$ and $h_1 = 20$.

Further, the length of an embryo is the number of specified tasks in it. During stage t, the growing operator is applied to the individuals whose length is less that $\lambda(t)$. The result of the operator replaces the parent. Often, the child has a weaker performance that its parent, because the shorter representations have a better fitness but this fitness is rather a more optimistic and less accurate estimation than that given by a longer embryo.

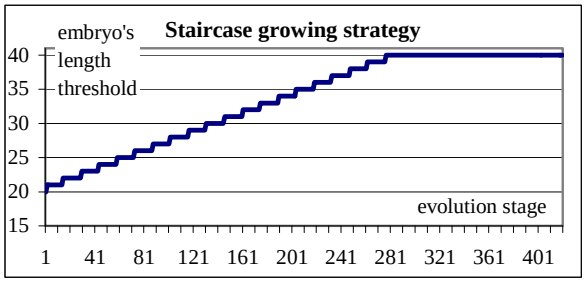


Fig. 4. Growing strategy

The required increase of an embryo from the initial to its final length could be achieved in several ways. The embryo can increase its size no matter how much, but it is not indicated to achieve a larger increase because the resulted chromosome is disadvantaged during the survival selection by the shorter embryos. The definition of the *basic growing* is:

1. Determine the set of candidate tasks as in the branching rules for I/U-embryos.
2. Randomly apply one rule from the variants:

   g1. Randomly select a task from all candidates and assign it to the embryos.

   g2. Randomly select a task from a limited subset of candidate tasks having the largest execution times and assign it to the embryo.

   g3. Randomly select a task from a limited subset of candidate tasks with the most successors for growing a prefix (or predecessors if a suffix is under growing) to favor the occurrence of more candidates during the following growing and assign it to the embryo.

   Rules g, g2, g3 are selected with prescribed probabilities $\gamma_i, i = 1,2,3$, respectively.

Now, the following growing operators are applied with prescribed probabilities $\Gamma_i, i = 1,2$:

G1. Recursively apply the basic growing until the current workstation is filled up i.e. one more growing leads to a new workstation.
G2. Recursively apply the basic growing until the current workstation and the next one are filled up.

Appropriate distributions for growing are $\gamma_1 = 0.25$, $\gamma_2 = 0.5$, $\gamma_3 = 0.25$ and $\Gamma_1 = 0.9, \Gamma_2 = 0.1$. These distributions are experimentally obtained.

Mention that for U-embryos, the growing is applied on either prefix or prefix. In the spirit of the B&B technique, growing the prefix and then the suffix of the same parent can generate two children. Growing both prefix and suffix could lead to a higher premature increase of the chromosome so the child has little chance to survive if no protectionist tools are used. Strategies to correlate growing and survival selection are described in section 4.7.

5.2. AN EXAMPLE

As an example of applying the genetic operators, consider the acyclic digraph in Figure 5. The processing times are given in Table 1.

For illustrating the basic mutation step, consider the embryo

position	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8…*
x = (	1,	3,	2,	5,	7,	6,	4,	8,	*)

where the positions of genes are in italic. One selects task 5 on the position 3. Task 3 is its predecessor, thus $l_{left} = 1$. The unique successor of task 5 is task 10 that does not appear in x and in the most unfavorable case it can occur on position 8, thus $l_{right} = 8$. So, task 5 can be moved anywhere between positions 2-7. If one chose 6 is its final rank, then the sequence of tasks 7,6,4 moved leftward and $x' = (1,3,2,7,6,4,5,8,*)$.

position	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8…*
x′ = (	1,	3,	2,	7,	6,	4,	5,	8,	*)

Now, apply a mutation to the U-embryo

position	*0*	*1*	*2*	*3*	*4*	*…*	*9*	*10*	*11*	
y = (	1,	4,	9,	2,	7,	*,	12,	11,	10	)

and select task 11 from the suffix. The predecessor of 11 is task 7 that is in the prefix, thus $l_{left} = 5$. Task 11 has no successor, therefore $l_{right} = 11$. Moving task 11 as left as possible, it results

position	*0*	*1*	*2*	*3*	*4*	*5*	*…*	*10*	*11*	
y' = (	1,	4,	9,	2,	7,	11	*,	12,	10	)

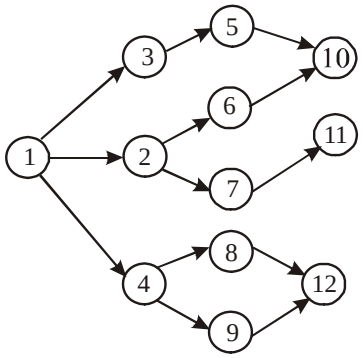


Fig. 5. Acyclic digraph for illustrating mutation, crossover and growing.

Now, for illustrating the crossover, consider that $t = 10$ and the embryos $\mathbf{x}$ and $\mathbf{y}$ above are the parents. For the processing times in Table 1, one obtains $m_0 = [29/6] = 5$, $n_{ws} = 1.2$, the lengths of prefixes/suffixes are $k_1 = 8$, $h_1 = n + 1 = 13$, $k_2 = 5$, $h_2 = 3$ and $nc = 3$. For $\gamma = 1.1$ it results $\sigma(10) = 0.018$ and the number of cutting points is $np = ceil\left(0.9 \cdot \left[3 \cdot (1 - 0.018) + 5 \cdot 0.018\right]/2\right) = ceil(1.36) = 2$. One generates two cutting points in each part of x and y, namely x=(1,3,2,|5,7,6,4,8,*) , y=(1,|4,9,2,7,*,12,11,|10). The first child is $\mathbf{c}_1$=(1,3,2,|5,7,6,4,8,| 9,*, 12,11,10). By interchanging the roles of parents, one obtains the second child $\mathbf{c}_2$=(1,|3,2,| 4,9,7,| 5,6,8, *, 12,11,10).

Now, consider $\mathbf{x}$ and $\mathbf{y}$ for basic growing. The set of candidates for completing $\mathbf{x}$ is $\{9,10,11,12\}$. If rule (g2) is applied then largest execution time is for task 10 or 12 and task 10 is selected. Thus $\mathbf{x}$ grows to $\mathbf{x}^{(1)}$=(1,3,2,5,7,6,4,8,10,*). For $\mathbf{y}$, the set of candidates for prefix (1,4,9,2,7) is {3,6,8} and if (g3) is applied (each candidate has one successor) and task 6 is selected then $\mathbf{y}$ grows to $\mathbf{y}^{(1)}$=(1,4,9,2,7,6,*,12,11,10). On the other hand, the set of candidates for the suffix of $\mathbf{y}$ is {5,6,7,8,9} and task 8 is randomly selected to precede the suffix. Thus, another possibility is to grow y to $\mathbf{y}^{(2)}$=(1,4,9,2,7,*,8,12,11,10). Notice that tasks 7,6,9 have the successors in the suffix but they are not candidates because they already are in the prefix of $\mathbf{y}$.

Finally, the computing of the fitness is illustrated for embryo $\mathbf{y}^{(2)}$=(1,4,9,2,7,*,8,12,11,10) and $C = 6$ with the processing times in Table 1. The procedure presented in section 4.3 returns the partial solution described in Table 2 and the term m_c=3. The layout of this solution is shown in Figure 6. The predicted

value of the number of workstations to accommodate the unassigned tasks 3,5,6 is $m_e = [t_3 + t_5 + t_6 - (C - T(W_3))]/C = 1.83$. Therefore, $fit(\mathbf{y}^{(2)}) = 4.83$.

Table 2
The workstations built on U-embryo (1,4,9,2,7,*,8,12,11,10)

i	W_i	$T(W_i)$
1	{1,10}	6
2	{4,11,12,8}	6
3	{9,2,7}	6

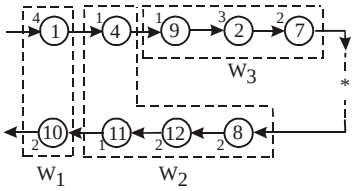


Fig. 6**.** Layout of the partial solution given by (1,4,92,7,*,8,12,11,10).

5.3. SELECTION FOR SURVIVAL

No matter what the selection criterion is, survival selection is deterministic and elitist in the limit of the population size. Further, some issues that are specific to the embryonic approach are discussed.

Accuracy versus survival chance. As a common and general fact with B&B methods for minimization problems, the value of the estimate function increases when it is applied from the parent to the children, i.e. the individual size increases. This is a normal situation because the unknown part becomes less and less, and the estimation gets closer to reality. Consequently, during the survival selection, individuals that are closer to their final length lose in favour to some individuals with small dimensions, but which have small estimations. The risk, experimentally observed, is that, for certain instances, BB_GA cannot deliver complete solutions. This phenomenon is observed for other problems as well and is related to the so-called *phase transition* that means that the changing of the state in the solving of the problem instance is accompanied by a large variation of the objective function. For ALB, this occurs whenever one passes from the completion of the current workstation (while the number of workstation is constant) to the initialization of the next workstation moment at which the number of workstation increases with 1. This is the reason for which the fitness function defined here is a real value one, although the objective is minimizing the number of workstations. Similar phenomena appear for routing problems when passing from a

cluster of locations to another group, because the transition between clusters leads to a high increment in the total cost, so the extension of the embryos is penalized [BruV07]. Two strategies are presented as a remedy.

Strategies for balancing predicted and real performance. The first strategy concerns the preventive management and it forces the embryos to synchronously grow so that the population of embryos does not have high variations in lengths. This is done using the staircase growing strategy for applying the growing operators. The growing operators are applied to those embryos whose length is less that the target length $\lambda(t)$ at stage t and the offsprings are evaluated. Then, elitist determinist selection is made in the limit of the population size. Long prefix/suffix embryos resulted from crossover, can be frozen for some evolution stages and compete with the current population as soon as the own length is not too different from the average of the population.

The second strategy is based on corrective management. In this case, the growing operator is applied to the entire population with probability p_g. Let us denote by $len(\mathbf{x})$ the number of known genes in the chromosome $\mathbf{x}$. Suppose that the numbers of individuals before survival selection is pp, with $pp > ps$. Some chromosomes must be removed from the population so that (i) the final average of the chromosomes lengths approaches $\lambda(t)$ and (ii) the sum of the fitness value of the remaining individuals is as small as possible. The following greedy method combines these objectives in one optimization measure by reordering the embryos $\mathbf{x}_1,....,\mathbf{x}_{pp}$ so that $fit(\mathbf{x}_i)/len(\mathbf{x}_i) \le fit(\mathbf{x}_{i+1})/len(\mathbf{x}_{i+1})$, $i = 1,..., pp-1$ and keep the individuals $\mathbf{x}_1,...,\mathbf{x}_{ps}$ for the next generation. The fitness/length measure ensures that for the same length the smallest fitness is preferred, whereas at the same fitness value the embryo that has most information is favored. This is equivalent with the temporary replacement of the current fitness $fit(\mathbf{x})$ with $fit(\mathbf{x})/len(\mathbf{x})$.

5.4. POPULATION MANAGEMENT

One of the objectives of this hybridization is to give a chance for each method to capitalize its major strong points. For this reason, some arrangements are done in the evolution process.

Tuning the growing. Experimental investigation shows that, in general, it is better to the increase the embryos from time to time, not during every generation. That means to apply the growing more rare but to more embryos. Equally, one can establish large stairs in the allure of function λ by keeping the slope of the oblique side and to test the length at every R stages. This allows that the evolution mechanism acts more time on successive level of state space tree. It is convenient to take $R \approx \Delta t$.

Evolution epochs. The hybrid algorithm has two main epochs in its action. The first is the maturation era that starts with embryos at time t_1 and ends with adult chromosomes at evolution stage t_a. The duration of the maturation depends on the initial lengths of embryos, the length of the adults and the desired growing rate. An average growing rate has good effects on the exploration quality. A short one expands the computing time and could lead to less promising zones of the search space because the driving is based on prediction instead of an exact performance score. Too large growing rate do not allow profiting of the possibility to make good selections in the early stages of solution constructing. From the moment t_a, it begins the exploitation in which the first part is dedicated to minimizing the number of workstations. Its second part begins at the moment $t_s > t_a$ detected by the condition described below and from t_s until $t_{\max}$, the algorithm acts for maintaining the best-found number of workstations and to equalize the loading of the stations. Computational experience suggests that $t_a = pf \cdot t_{\max}$ with $pf = 0.6 - 0.7$ and $t_s = t_a / 4 + 3t_{\max} / 4$.

Changing of evolution goal and stopping. Denote by $\varphi(t)$ the average of the fitness value of the population in the stage t of the evolution. BB_GA contains a simple optimality test that can be performed on the best found solution, as it is mentioned in the final of first section. The algorithm changes its fitness function when the best fitness in the population equals the lower bound m_0, or $\varphi(t)$ stagnates for a given number of successive stages. This moment is just t_s. Previous computational experience with many ALB instances indicate that after minimizing the number of workstations the evolution can continue with the improving the balance of workstations by keeping the best found number of workstations. This can be done by changing the definition of the current fitness to

$$\overline{fit}(\mathbf{x}) = w_1 \cdot (mC - T_{tot}) + w_2 \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(C - T\left(W_j\right)\right)^2} \ ,$$

where the first term represents the total idle time of workstations, and the second is the smoothness of the workstations. The positive weights w_1 and w_2, verify $w_1 + w_2 = 1$ and w_1 varies from 0.9 to 0.2 when the iteration number goes from t_a to $t_{\max}$. So, between t_a and $t_{\max}$, the evolution is focused on the reducing of the imbalance between workstations. The condition to stop the algorithm is the stagnation of the average $\overline{\varphi}(t)$ of new fitness for some successive stages in the limit of $t_{\max}$, or $t > t_{\max}$.

As a conclusion of this design specification, the grafting of the B&B method on GA and the strategies used to unify their behaviors, leads to good balance between exploration and exploitation. More specifically, during the first stages of the evolution, it is desired to preserve within the population some short-medium embryos that should act as representatives of large zones of the search space in order to improve the exploration. As long as the evolution advances, it has in view the concentration on smaller zones (longer embryos), which are supposed to be of good quality and this is the beginning of exploitation. Finally, the most promising zones from the search space are intensively exploited and complete solutions to the problem are provided.

6. EXPERIMENTAL EVALUATION

Numerical experiments have been done using the test instances described in MERTENS.IN2 (7 tasks), HESKIA.IN2 (28 tasks), LUTZ1.IN2 (32 tasks), GUNTHER.IN2 (35 tasks), ARC83.IN2 (83 tasks), ARC111.IN2 (111 tasks), and SCHOLL.IN2 (297 tasks). These instances can be found in [*SCHs09] for I-lines, while the test instances for U-shaped lines can be found in [*SCHu09]. The instances for CALB are obtained from Lutz instance with 32 tasks and can be found in [*CALB09].

The values of the parameters that were used in experiments are: ps=100, p_c=0.5, p_m=0.05, mating pool consists of 50% of the best individuals, the length limits for the initial population $ph_1 = 0.05$, $ph_2 = 0.2$, $h_1 = (h_{\min} + h_{\max})/2$, $t_{\max} = 1000$, $pf = 0.65$, $p_g = (n - h_1)/((t_a - 1) \cdot ps)$. Let m* be the number of workstations of the best solution found

by BB_GA, while m_{opt} denotes the minimum number of workstations for the current instance. As before, m_0 is the lower bound defined in section 2.

The experiment involves the solving of groups of ALB instances, each group having the same precedence digraph and different C-values. For each C-value, 24 runs were executed.

The first goal of the experiment was to compare BB_GA with the pure GA. The test included the intances shown in Table 3 and different values of the cycle times. The obtained approximating distribution of m^*-m_0 is shown Table 3.

Table 3

I-shaped lines: distribution of m^*-m_0, for BB_GA and pure GA.

	Instance	C	m*-m0			m_{BB_GA}	m_{GA}	I_q
			0	1	2			
	I-shaped							
1	ARC111.IN2	5755	0.84	0.16	0	0.16	1	6.25
2	ARC111.IN2	5785	0.97	0.03	0	0.03	1	33.3
	average		***0.90***	***0.10***	***0.00***	***0.10***		***19.77***
3	SCHOLL.IN2	1394	0	0	1	2.00	3.37	1.68
4	SCHOLL.IN2	1422	0.10	0.90	0	0.90	2.37	2.63
	average		***0.05***	***0.45***	***0.50***	***1.45***		***2.16***
	U-shaped							
5	SCHOLL.IN2	1394	0	0.8	0.2	1.20	2.82	2.35
6	SCHOLL.IN2	1422	0.08	0.9	0.02	0.94	2.42	2.57
	average		***0.04***	***0.85***	***0.11***	***1.07***		***2.46***
	CALB-I-shaped							
7	LUTZ1_coverS.xls	1583	0.81	0.09	0.1	0.29	0.81	3.103
8	LUTZ1_coverS.xls	2035	0.82	0.18	0	0.18	0.42	4.556
9	LUTZ1_coverS.xls	3560	0.84	0.14	0.02	0.18	0.14	0.778
	average		***0.82***	***0.14***	***0.04***	***0.22***		***1.97***

The mean of the approximating error distribution achived by BB_GA is shown in the column labeled with m_{BB_GA}, whereas in column m_{GA} the average of the error achieved by the pure GA is given. The quality index is defined as $I_q = m_{GA} / m_{BB\_GA}$, for $m_{BB\_GA} \neq 0$ and $I_q = 1$ if $m_{BB\_GA} = m_{GA} = 0$. The I_q-value is a measure of how much better BB_GA works in comparison to pure GA. Both techniques always found the optimal solutions for LUTZ1.IN2. The most important gain was obtained for ARC111.IN2 instances, for which the hybrid ensures an error (in average) 19.77 times smaller than the pure GA (lines 1-2), whereas for SCHOLL.IN2 instance (297 tasks) the average gain is 2.16 (lines 3-4). This gain is about 2.5 for the group of instances of U-shaped lines (line 5-6). For CALB with I-shaped lines, the hybrid BB_GA is 1.97 times better

than classical GA. Similar results are obtained for CALB with U-shaped lines. The conclusion is that BB_GA is better than its pure counterpart. Because $m_{opt} \geq m_0$ it results that $m^* - m_{opt} \leq m^* - m_0$, thus the real performance is better than the estimations above. It reaches many times the optimum even in difficult cases.

The second goal of the experiment was to measure the performance of BB_GA with regard to the deviation of the best computed solution from the optimal one. The results of evaluating the performance of BB_GA for a larger data set is shown for I-lines and U-lines in Table 4 and Table 5, respectively. For each instance, the performance is estimated by approximating the distribution of $m^* - m_{opt}$ (columns labeled with *rel. freq.*). The average of this distribution is shown in column entitled *abs. dev. from opt.* The last column shows the value of $(m^* - m_{opt}) / m_{opt}$ (*rel. dev. from opt.*). Last line in each group corresponding to the same digraph contains the average values aggregated for all instances generated by different C-values. The results in Table 4 and Table 5 are self-explanatory.

Table 4
Error distribution, average and relative deviation from optimum for I-lines.

I-shaped			**m*-m$_{opt}$**			**rel. dev.**
			0	**1**	**av. err.**	**from**
Digraph	**C**	**m$_{opt}$**	**rel. freq.**			**opt.**
Mertens	6	6	1.00	0.00	0.00	0.000
7	7	5	1.00	0.00	0.00	0.000
tasks	8	5	1.00	0.00	0.00	0.000
	10	3	1.00	0.00	0.00	0.000
	15	2	1.00	0.00	0.00	0.000
	18	2	1.00	0.00	0.00	0.000
average			***1.00***	***0.00***	***0.00***	***0.000***
Heskia	138	8	0.83	0.	0.17	0.02
28	205	5	0.67	0.33	0.33	0.067
tasks	216	5	0.92	0.08	0.08	0.017
	256	4	0.75	0.25	0.25	0.063
	324	4	0.92	0.08	0.08	0.021
	342	3	0.75	0.25	0.25	0.083
average			***0.81***	***0.19***	***0.19***	***0.045***
Lutz1	1414	11	1.00	0.00	0.00	0.000
32	1572	10	1.00	0.00	0.00	0.000
tasks	1768	9	1.00	0.00	0.00	0.000
	2020	8	1.00	0.00	0.00	0.000
	2357	7	1.00	0.00	0.00	0.000
	2828	6	1.00	0.00	0.00	0.000
average			***1.00***	***0.00***	***0.00***	***0.000***
Gunther	41	14	0.67	0.05	0.33	0.024
35	44	12	1.00	0.00	0.00	0.000
tasks	49	11	1.00	0.00	0.00	0.000
	54	9	1.00	0.00	0.00	0.000

	61	9	1.00	0.00	0.00	0.000
	69	8	1.00	0.00	0.00	0.000
	81	7	1.00	0.00	0.00	0.000
average			***0.95***	***0.05***	***0.05***	***0.003***
Arcus	3786	21	1.00	0.00	0.00	0.000
83	3985	20	0.67	0.33	0.33	0.017
taks	4206	19	0.92	0.08	0.08	0.004
	4454	18	0.67	0.33	0.33	0.019
	4732	17	0.75	0.25	0.25	0.015
	5048	16	0.83	0.17	0.17	0.010
	5408	15	0.92	0.08	0.08	0.006
	5824	14	1.00	0.00	0.00	0.000
average			***0.84***	***0.16***	***0.16***	***0.009***

The conclusion of the experiments is that the grafting of the B&B method on a GA for solving ALB problem certainly improves the performance of the GA and BB_GA is able to find the optimal solutions of very difficult ALB instances with a high frequency, even if it not an exact method. Moreover, it requires modest amounts memory and computing time.

Table 5

Error distribution, average and relative deviation from optimum for U-lines.

U-shaped			**m^*-m_{opt}**				**rel.**
			0	**1**	**2**	**av.**	**dev.**
Digraph	**C**	**m_{opt}**	**rel. freq.**			**err.**	**from opt.**
Mertens	6	6	1.00	0.00	0.00	0.00	0.000
7	7	5	0.83	0.17	0.00	0.17	0.033
tasks	8	5	1.00	0.00	0.00	0.00	0.000
	10	3	0.92	0.08	0.00	0.08	0.028
	15	2	1.00	0.00	0.00	0.00	0.000
	18	2	1.00	0.00	0.00	0.00	0.000
average			***0.96***	***0.04***	***0.00***	***0.04***	***0.01***
Heskia	138	8	0.08	0.42	0.50	1.42	0.177
28	205	5	0.00	1.00	0.00	1.00	0.200
tasks	216	5	0.75	0.25	0.00	0.25	0.050
	256	4	0.08	0.92	0.00	0.92	0.229
	324	4	1.00	0.00	0.00	0.00	0.000
	342	3	0.25	0.75	0.00	0.75	0.250
average			***0.36***	***0.56***	***0.08***	***0.72***	***0.151***
Lutz1	1414	11	0.67	0.33	0.00	0.33	0.030
32	1572	10	0.00	1.00	0.00	1.00	0.100
tasks	1768	9	1.00	0.00	0.00	0.00	0.000
	2020	8	1.00	0.00	0.00	0.00	0.000
	2357	7	1.00	0.00	0.00	0.00	0.000
	2828	6	1.00	0.00	0.00	0.00	0.000
average			***0.78***	***0.22***	***0.00***	***0.22***	***0.022***
Gunther	41	14	0.33	0.50	0.17	0.83	0.060
35	44	12	0.33	0.67	0.00	0.67	0.056
tasks	49	11	0.42	0.58	0.00	0.58	0.053
	54	9	0.50	0.50	0.00	0.50	0.056
	61	9	0.67	0.33	0.00	0.33	0.037
	69	8	0.75	0.08	0.17	0.42	0.052
	81	7	0.83	0.17	0.00	0.17	0.024

average			***0.55***	***0.40***	***0.05***	***0.50***	***0.048***
ARC83	3786	21	0.58	0.33	0.08	0.50	0.024
83	3985	20	0.33	0.58	0.08	0.75	0.038
tasks	4206	19	0.67	0.25	0.08	0.42	0.022
	4454	18	0.50	0.42	0.08	0.58	0.032
average			***0.52***	***0.40***	***0.08***	***0.56***	***0.029***

Some qualitative aspects concerning the behavior of BB_GA are illustrated further.

Experiments show that, in general, the preventive management strategy is better that corrective management. The corrective strategy offered better results in situations when the processing times are not very small as compared to the cycle time. It is worth mentioning that the pure GA cannot match the performance of BB_GA even if the maximum number of allowed iterations is doubled and the mutation rate is artificially increased in order to avoid stagnation and forcing the search.

The graph in Figure 7 represents the variation of the best individual fitness along the evolution for instance ARC111.IN2 with $C = 5755$.

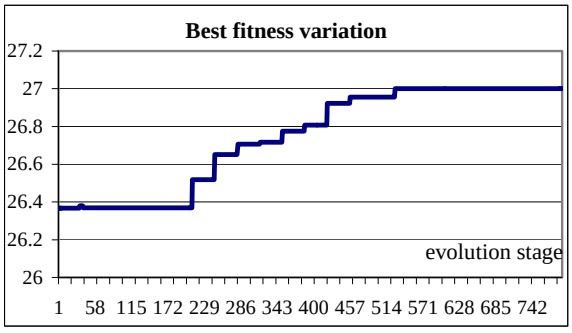


Fig. 7. Variation of the best fitness value vs. number of iterations

The graph indicates an increasing trend, whereas a pure GA has a decreasing allure. This difference appears because the fitness function is a lower bound of the minimum number of workstations and it becomes closer to the minimum value of the number of workstations of the subset of solutions corresponding to the embryo. At $t = 33$, an increase of the best fitness that is followed by a decrease occurs. This non-monotonic variation is due to the increasing of the length and fitness of embryos but after 6 evolution stages better embryos having the same length are found and this leads to a fitness decreasing. After generation 550, only complete representations are evolved, the current fitness value stagnates at the optimal value 27 and the improving of the smoothness begins.

The variation of the average of the fitness during evolution is represented in Figure 8. The pseudo-periodical peaks correspond to the moments when the length of all the individuals is brought to the minimum threshold. This causes a jump in the graph because the precision of the estimation increases. Few generations later, the evolution of embryo reestablishes the fitness to some better values (sudden decreases that succeed to sudden increases). Toward the end of the evolution, all the adults have values gathered around the best value, due to the elitist selection.

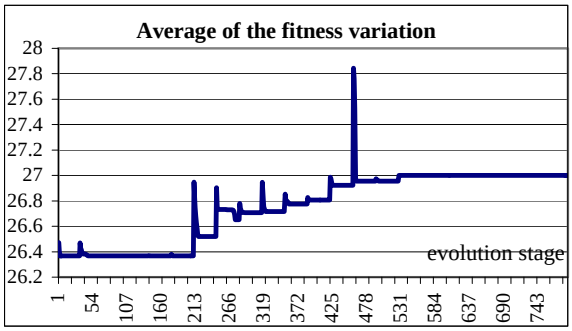


Fig. 8. Variation of the medium fitness vs. number of iterations

In Figure 9, the variation of the length of the best fitness chromosome is depicted. The almost periodical increases correspond to bringing the individuals to the minimum imposed size, and the plateaus correspond to the iterations in which the individuals maintain the imposed size, until to the next increase one produces.

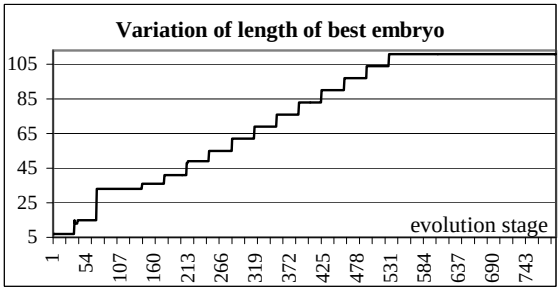


Fig. 9. Variation of the length individual with the best fitness vs. number of iterations

Figure 10 shows the variation of the average of the lengths of the individuals along evolution. A short spike appears at the beginning of each plateau period. Such a spike is generated by the massive applying of the growing operator to some embryos coming from crossover and having lengths that exceed the average of the population. After few evolution stages, these longer embryos improve the performance or lose the competition and the spike disappears.

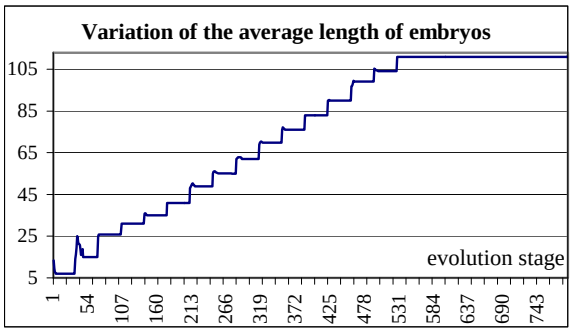


Fig. 10**.** Variation of the medium length population vs. number of iterations.

From these graphs, it results that the "steps" within the evolution of lengths are more than the "steps" appearing in the evolution of the fitness values. This thing indicates that the fitness value does not increase as fast as the length of embryo does. So, the first large plateau (stages 1-200) in the graph in Figure 8 expresses the poor discriminating power of the fitness of short embryos that is natural due to the lack of information in them. This situation shows that even if one works with large embryos, the generations succeed to maintain a better value of the fitness function although they advanced in length. In other words, the individuals are concentrated on subtrees containing high quality solution, which is a specific effect of the B&B type grafting.

As a conclusion, some of the major advantages of the genetic algorithm hybridized with B&B over the pure GA are: (i) it requires fewer generations to obtain similar results due to the rapid sorting of the low quality zones of the search space, (ii) it is faster because the most part of time it works with shorter representations and (iii) if the number of generations is the same for both hybrid and pure GA, the first one offers in all cases better results, due to a more intelligent guiding through the search space.

7. CONCLUSIONS AND FURTHER RESEARCH DIRECTIONS

A new and efficient hybrid method resulted from the grafting of a branch & bound technique on a genetic algorithm for solving the "I"-shaped and "U"-shaped ALB problems was presented. Specific genetic operators including a new type of growing operator acting on partial representations of the solutions are proposed. Fitness function acting like an estimate function specific to B&B methods was used. Growing operator, survival strategies, and appropriate population management rules have been designed.

The evolving mechanism operates with subsets of solutions and has very good capabilities to locate the best potential zones of the search space early and thus better exploit them. Conceptually, both B&B and pure GA are particular cases of the proposed hybrid algorithm: the first when only the best embryo is branched and no survival selection is made, the second when one operates exclusively with adult chromosomes. The efficiency of grafting the B&B method on a genetic algorithm led to an efficient tool due to an appropriate tradeoff between exploration and exploitation. Experimental investigation has shown that the performances of this new hybrid method are better than those of the basic genetic algorithms. The impact of these results is larger because these design elements can be used to combine other variants of such algorithms within ALB paradigm as well as for other optimization problems. This type of hybrid GA can also be used as the "working horse" in more sophisticated metaheuristics, like cellular or segregative GA. This is the subject of some further research directions. More accurate fitness evaluation of embryos in the case of U-shaped lines and mixed models are of real interest. Extending this type of grafting to multiple models assembly lines seems to be a challenge too.

Acknowledgement. The second author acknowledges that this work was partially supported by the European Social Fund in Romania, under the responsibility of the Managing Authority for the Sectorial Operational Programme for Human Resources Development 2007-2013, Grant POSDRU/88/1.5/S/47646.

Authors' contributions. O. B. developed the concepts and the methodology for combining B&B with GAs, adapted them to ALB paradigm, designed the components of the hybrid GA, designed the experiments, interpreted the results and wrote the paper. C.R. wrote the code, assembled input data, run the model, experimentally evaluated and redesigned different growing strategies, performed all experiments, analyzed and interpreted output data, and edited the manuscript.

REFERENCES

1. [BoyFS07] BOYSEN N., FLIEDNER M., SCHOLL A., *A classification of assembly line balancing problems,* European Journal of Operational Research, 2007,**183**, 674-693.
2. [Bru99] BRUDARU O., *A genetic algorithm for assembly line balancing with compatibility constraints using a control mechanism based on information energy,* EUFIT'99 – 7th European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, Sept. 13-16, 1999.
3. [BruMD10] BRUDARU O., MORARU E., DAMASCHIN A., *Embryonic hybrid genetic algorithm for k-repairmen problem,* MOPGP'10 – The 9th International

Conference on Multiple Objective Programming and Goal Programming, May 24-26, 2010, Sousse, Tunisia (submitted for publication).

4. [BruS01] BRUDARU O., SOFRONIE V., *Optimal solution of assembly line balancing with fuzzy times using a branch and bound technique,* ICPR-16, 16th International Conference on Production Research, July 30- August 3, 2001, Prague.
5. [BruV04] BRUDARU O., VALMAR B., *Genetic algorithm with embryonic chromosomes for assembly line balancing with fuzzy processing times*, The 8th International Research/Expert Conference Trends in the Development of Machinery and Associated Technology, TMT , 15-19 sept. 2004, Neum, Bosnia and Herzegovina, pp. 875-878.
6. [BruV07] BRUDARU O., VALMAR B., *Hybrid genetic-algorithm / branch & bound technique to solve a time-dependent transportation problem*, Proc.of 6th Eurosim Congress on Modelling and Simulation (B. Zupancic, R. Karba, S. Blazic, eds.), Ljubljana, Slovenia, Sept. 9-13, 2007 , vol. **2**, pp. 1-7.
7. [Gen08] GEN M., CHENG R., LIN L., *Assembly Line Balancing Models*, Network Models and Optimization, Springer, London, 2008, 477-550 .
8. [Knu76] KNUTH D. E., *The Art of Computer Programming*, Technical Publishing House, Bucharest, 1976 (in Romanian).
9. [Mic94] MICHALEWICZ Z., *Genetic Algorithms + Data structures = Evolution Programs*, Springer Verlag, Berlin, 1994.
10. [Pas09] PASZKOWICZ W., *Genetic Algorithms, a Nature-Inspired Tool: Survey of Applications in Materials Science and Related Field*s, Materials and Manufacturing Processes, 2009, **24,** 174 – 197.
11. [Sch98] SCHOLL A., *Balancing and sequencing of assembly lines*, 2nd edition, Physica-Verlag, Heidelberg, 1998.
12. [TasT08] TASAN S., TUNALI S., *A review of the current applications of genetic algorithms in assembly line balancing*, Journal of Intelligent Manufacturing, February 2008, **19,** *1,* 49-69.
13. [*SCHs09] http://www.assembly-line-balancing.de/files/uploads/SALBP data sets.zip, accessed June 2009.

14. [*SCHu09] http://www.assembly-line-balancing.de/files/uploads/UALBP-1_data sets.zip, accessed June 2009.
15. [*CALB09] http://www.misp.tuiasi.ro/obrudaru/line_balancing/CALBs.rar, posted October 2009.

Received April 14, 2010