GENETIC ALGORITHM FOR ASSEMBLY LINE BALANCING WITH COMPATIBILITY CONSTRAINTS USING A CONTROL MECHANISM BASED ON INFORMATION ENERGY

Octav Brudaru
Technical University "Gh. Asachi" Iasi
Department of Production Systems Engineering and Management
Copou 22, 6600 Iasi, Romania
email: brudaru@tuiasi.ro

ABSTRACT. This paper presents a genetic algorithm for the assembly line balancing problem with an additional type of constraint, which requires that the workstations are compatible with a given cover of the assembly tasks. Specific structural genetic operators are used in order to maintain the topological order of the offsprings and no repair procedure is needed. In order to prevent the premature stagnation we propose a control technique of genetic operator frequency using the information energy within population. This technique maintains the diversity of individuals by ensuring a prescribed value of the information energy or by minimizing this energy.

KEY WORDS: genetic algorithm, information energy, population variability, line balancing, cover compatibility.

1. INTRODUCTION

The original assembly line balancing (ALB) problem concerns the clustering of a set of nondivisible tasks of an assembly process into workstations, so that the precedence constraints in task execution are respected and execution time of each workstation does not exceed the cycle time. The number of workstations has to be minimized. This optimization problem is NP-hard. Different variants of ALB problem and solving techniques can be found in [RusB90], [Tal89], [BruB96], [BruBR97], and [TsuG95].
In this paper we consider the ALB problem with compatibility constraints and present a genetic algorithm for solving this problem. The information energy of the population under evolution is used in order to estimate the diversity of individuals. A feedback mechanism is proposed in order maintain a desired level of this energy by acting on the probability of mutation and crossover events.
In Section 2, we formulate the ALB problem with compatibility constraints. Sections 3 describes the representation of problem solutions and the generation of the initial population, gives the evaluation algorithm, the replacement strategy and the rule of population sizing. Also, it presents the genetic operators and the strategy to use them, and explains the use of information energy within population for preventing premature stagnation. The results of experimental investigation of the proposed algorithm are given in Section 4. Last section summarizes the work.

2. PROBLEM FORMULATION

The assembly line balancing with compatibility constaints (ALBC) can be stated as following [Bru98]. Let $V=\{1,2,...,n\}$ be the set of tasks. The acyclic digraph $G=(V,A)$ contains the precedence constraints in task execution. If $(x,y)\in A$ then, the execution of y can begin after the ending of task x. The execution time of task i is denoted by $t(i)$ and is a positive real number. The cycle time C is the time elapsed between two successive outputs from the line and we suppose that $t(i)\le C, i=1,...,n$. Let $Q=\{Q_1,...,Q_p\}$ be a cover of V i.e. $V=Q_1\cup...\cup Q_p$. Let S be the set of all partitions $W=\{W_1,...,W_m\}$ of V that satisfy the following conditions:

if $(x,y)\in A, x\in W_r, y\in W_s \Rightarrow r\le s$; (1)

$$T(W_j) = \sum_{x \in W_j} t(x) \le C,\ j = 1,...,m; \quad (2)$$

$$(\forall)\, j \in \{1,...,m\}\ \ (\exists) k(j) \in \{1,...,p\} \text{ so that } W_j \subseteq Q_{k(j)}. \quad (3)$$

Each member of S is called a *solution* to ALBC. A solution with a minimum number m of workstations is an *optimal solution*.

We note that the original ALB problem only assumes condition (1) and (2). Condition (3) is the *compatibility constraint*. While the original is NP-hard, the ALBC remains NP-hard.

3. GENETIC APPROACH

In this section we present the components of the genetic algorithm.

3.1. REPRESENTATION AND INITIAL POPULATION GENERATION

There is an intrinsic relation between the set of partition of V satisfying (1) and the set of permutations of V that sort the set of tasks in topological order [Hors82]. So, a permutation $x = (x_1,...,x_n)$ induces a topological order if $(x_r, x_s) \in A$ implies $r < s$. Actually, if the tasks in each workstation are arranged in topological order and these sequences are concatenated in the order of workstations, the resulted permutation is in topological order and is the technological assembly flow. Thus, a natural way to encode a solution is to use a permutation $x = (x_1,...,x_n)$, which sorts V in topological order. The chromosome is just such a permutation. The way to obtain the best solution corresponding to x is shown in Section 3.2.1.

A first benefit of this representation is the generation of the initial population. Let K be the number of levels of G. These levels exist because G is acyclic and can be easily obtained starting from a an arbitrary topological permutation $x = (x_1,...,x_n)$ of G and applying the steps below:

1. Set $c(x_h) = 1$, for all tasks x_h without predecessors in G;
2. Compute $c(x_i) = \max\{c(z)/(z, x_i) \in A\} + 1$, for each task x_i with predecessors.
3. Determine $K = \max_i c(x_i)$, and $L_k = \{i / c(i) = k\}$, $k = 1,...,K$.

Now, a chromosome of the initial population is obtained by concatenating the random permutations of $L_1,...,L_K$. In this way, it is possible to obtain $card(L_1) \times ... \times card(L_K)$ individuals.

3.2. EVALUATION, POPULATION SIZE AND REPLACEMENT

In this section we present a simple algorithm for computing the minimum number of workstations for a given chromosome that identifies a certain assembly sequence. Also, we give the sizing of the population at the end of each stage and the manner to select the parents and offsprings for the next iteration.

3.2.1.Evaluation

The evaluating of the chromosome $x = (x_1,...,x_n)$ returns a natural number $m(x)$ equal to the minimum number of workstations among the all solutions that have x as an assembly flow. The next greedy method constructs this solution $W = \{W_1,...,W_m\}$ with $m(x) = m$, and also returns $l_j = card(W_j)$, $j = 1,...,m$.

1. $m = 1, l_1 = 0, W_1 = \varnothing$;
2. for i=1,…,n do:
   if $T(W_m \cup \{x_i\}) \le C$ and $(\exists h \in \{1,...,p\}, W_m \cup \{x_i\} \subseteq Q_h)$
   then add x_i to W_m, $l_m = l_m + 1$,
   else $m = m + 1$, $l_m = 1$, $W_m = \{x_i\}$.
3. $m(x) = m$

Actually, we only need $m(x)$ and $l_1,...,l_{m-1}$.

3.2.2. Population size

Denote by $NP(s)$ the population size in the s-th stage of evolution, $s = 1,...,S$. We use a logarithmic dependency given by $NP(s) = [\lg(as + b)]$, where a and b are calculated from the conditions: $NP(1) = NP_{\min}$, $NP(S) = NP_{\max}$, with $NP_{\min} \le NP_{\max}$, and [.] returns the smallest integer greater than or equal to the argument. We prefer this variation because we start with small size population and terminate with a maximum value permitted by the amount of available memory. In this way we can consume a larger amount of memory for the offsprings in the earlier stages of evolution.

3.2.3. Replacement strategy

The set of offsprings generated in stage s using a population of $NP(s)$ individuals competes with the parent population in order to provide the best $NP(s+1)$ chromosomes for the stage $s+1$.

3.3. GENETIC OPERATORS

Further we present simple genetic operators that preserves the topological sorting of the chromosomes.

3.3.1. Crossover

The probability to produce a crossover is denoted by p_c. Once the crossover event is produced, the parents are obtained by generating two different uniformly distributed random numbers in the range $1,\ldots, NP(s)$. We use a structural crossover [Mic94, pp.261] that acts on the parents $x = (x_1,...,x_n)$ and $y = (y_1,...,y_n)$ and produces two offsprings $a = (a_1,...,a_n)$ and $b = (b_1,...,b_n)$ in the following way:

1. generate a cut point r in the range $1,...,n-1$;
2. take the segment $x_1,...,x_r$ and put it as prefix in a (i.e. $a_i = x_i, i = 1,...,r$);
3. starting from the beginning, the tasks in y are transferred in the same order to a for completing the locations $r+1,...,n$, omitting tasks already in a.

The offspring b is obtained in the same way interchanging the roles of x and y. An example is given in Table 1, for $n = 7$ and $r = 3$.

Table 1. Example of applying the crossover operator.

Parents	x							y						
	1	2	4	3	5	6	7	1	4	6	7	2	3	5
Children	a							b						
	1	2	4	6	7	3	5	1	4	6	2	3	5	

It is easy to prove that if x and y are in topological order then so are a and b.

3.3.2. Mutation

Denote by p_m the probability of mutation. If a mutation event appears then generate a random number in the range $1,...,NP(s)$ for finding a chromosome $x = (x_1,...,x_n)$ which is altered by the mutation operator to obtain a mutant $a = (a_1,...,a_n)$. The mutation operator is called *last-first mutual change* (LMC), and is suggested by the rule applied by the human operator to produce good balanced workstations. Suppose that evaluation algorithm described in Section 3.2.1 produced the number of workstations $m(x)$ and the number of tasks l_j in the workstation $W_j = \{a(j,1),...,a(j,l_j)\}$, $j = 1,...,m(x)$. The simplest form of LMC is to generate a random number between 1 and $m(x)-1$, say h, and tray to make a mutual change between $x_r = a(h,l_h)$ (the last task in W_h) and

$x_{r+1} = a(h+1,1)$ (the fist task in W_{h+1}). Since this change could lead to precedence constraint violation, the attempt is successful only if $(x_{r+1}, x_r) \notin A$. If the last task in W_h precedes the fist task in W_{h+1} then a different h is generated. In this way the mutant $a = (x_1,...,x_{r-1}, x_{r+1}, x_r, x_{r+2},..., x_n)$ satisfies the precedence constraints. A more general LMC can be obtained taking many interchange points. On the other hand, it seems that a mutual change between tasks that are not consecutive is computational intensive. So, it is better to use many times the simplest form of LMC operator.

3.4. INFORMATION ENERGY AND CONTROL OF GENETIC OPERATORS FREQUENCY

One of the major problems in classical GAs is the premature stagnation due to the loss of diversity in the population. As is shown in [Rya95] and [VemC95], niches and species or the isolation by distance models are able to maintain diversity in a population. Another remedy is to use a so-called statistical population, which has a considerably larger size than the GA working population [NegA95]. The price of these techniques is a large amount of computing time and memory space. One possible way to introduce diversity in a convergent population is to increase the mutation rate [Das95]. Also, the decrease of diversity could be the effect of a too high crossover probability. Thus, we have at least two simple tools to modify the population diversity. On the other hand, a simple way to measure the population diversity, especially with regard to the variability of the fitness function values, is to compute the information energy [OniS79] within current population. In case of our problem, consider the set $M(s) = \{m(x) / x \in POP(s)\}$, where $POP(s)$ is the population that enters in the $s+1$ evolution stage and is the result of mutation, crossover and replacement from stage s. Denote by d the number of elements in M and by $q_1,...,q_d$ their relative frequencies. Following [Onis78] the information energy of $M(s)$ is

$$E(s) = \sum_{j=1}^{d} q_j^2 \text{, where } \sum_{j=1}^{d} q_j = 1 \qquad (4)$$

and

$$1/d \le E(s) \le 1. \qquad (5)$$

Also, $E(s) = 1/m$ when $q_j = 1/d$, $j = 1,...,d$ and $E(s) = 1$, when for some j it holds $q_j = 1$ and $q_h = 0, h \neq j$. Another important property of information energy is: *the energy decreases whenever the difference between* q_i *and* q_j *decreases even if the inequality* $q_i > q_j$ *is preserved* [OniS78]. Therefore, the information energy is smaller as the variability in $E(s)$ is higher.

Now, suppose we had as objective for the stage s of the evolution a prescribed energy level $E_{ob}(s)$ of $M(s)$. We use the difference $\Delta E(s) = E_{ob}(s) - E(s)$ for modifying the mutation and crossover probabilities as it follows:

$$p_m = p_m - a \times \Delta E(s), \quad p_c = p_c + b \times \Delta E(s), \qquad (6)$$

where a and b are parameters in range $[0,1]$ and tune the action of ΔE on the probabilities. Whenever a new probability value goes out of the margins of $[0,1]$, it is set to some value closed to that margin but strictly inside the interval $[0,1]$. These new values p_m and p_c are used in stage $s+1$.

3.5. OUTLINE OF THE PROPOSED GENETIC ALGORITHM

Using the notation above, define

$$MIN(s) = \min\{m(x) / x \in POP(s)\},$$

$$MED(s) = \frac{1}{NP(s)} \sum_{x \in POP(s)} m(x)$$

and

$RD(s) = (MED(s) - MIN(s)) / MIN(s)$, representing the relative difference between the median fitness and the performance of the best individual. Denote by $S_{\max}$ the maximum number of evolution stages.

The structure of GA for ALBC is:

- Set $s = 1$ and generate the initial population $POP(1)$.
- While ($RD(s) > tol$ and $s < S_{\max}$) do:
  - $s = s + 1$;
  - compute $E(s-1)$ (the energy of $POP(s-1)$) and $E_{ob}(s-1)$ (target energy for $POP(s-1)$);
  - compute the probabilities p_m and p_c to use in the current stage;
  - generate offsprings by mutation and crossover;
  - evaluate offsprings;
  - determine the resulted population $POP(s)$.

We denote by $tol \in (0,1)$ a prescribed error threshold.

4. EXPERIMENTAL RESULTS

An investigation was carried out using 20 instances of the ALBC problem with up to 40 tasks. This set was split in 4 subgroups of 5 instances having the same digraph and execution times but different covers. The experiments included different structure of initial population, absence/presence of information energy control and different energy targets. A maximum number of 40 evolution stages were used. The population size varies from $SP(1) = 10$ to $SP(40) = 50$.

If the information energy control mechanism is not active then:

(a) the final selection set is dominated by the copies of chromosomes within a subset representing 5-8% from the entire set;
(b) the relative difference stopping condition becomes true after 15-20 iterations;
(c) for the same problem instance, the ratio between the mean value of m and the minimum m-value is in the range $(1.3, 1.5)$;
(d) no significant improvement of the results is obtained if the number of evolution stages is forced to be higher.

The activation of the information energy control leads to:

- a significant higher diversity in the final population expressed by a maximum value of relative frequency of the same m-value less than $1/10$;
- a higher number of stages needed to make true the relative difference stop condition, but in this case $SP(s)$ can be reduced with $20\% - 30\%$ without bad effects in quality of solutions;
- the ratio defined in (c) becomes $(1.1, 1.2)$;
- absolute better solutions in 80% of cases;

We taken $a > b$ (e.g. $a = 1/2$ and $b = 1/4$), in order to make the mutation more sensitive at ΔE-value, while $tol = 0.05 \div 0.1$. In all experiments we started with $p_m = p_c = 0.5$. The population energy oscillates around a mean value. The distance between this mean value and the target energy is less than 10% of the target value. No monotonic variation of the energy was observed even if sometimes the probabilities monotonically vary.

Experiments show that no gain is obtained if we take zero as target value for the energy (hopping in a minimization of the information energy). In this case, the control mechanism keeps the maximal and minimal value for p_m and p_c and the process becomes a random search. This aspect can be prevented if we consider (5) and use $E_{ob}(s) = 1/NP(s)$. This objective is more realistic than $E_{ob}(s) = 0$. On the other hand, since $NP(s)$ is an increasing sequence, the energy target is a decreasing one.

5. CONCLUSIVE REMARKS

We presented a GA for assembly line balancing with compatibility constraints. The genetic operators have good performances, a low computational complexity, and need no repair procedure. We included in the GA a control mechanism based on the information energy of the population. It is quite simple and significantly improves the variability of population. This mechanism offers better solutions and avoids premature stagnation of evolution.

Further direction of investigation are related to the following aspects:

- initial population should be generated so that condition (3) is favored (i.e. as many as possible successive tasks should be in the same Q -set);
- design of a local optimization operator oriented to the cover compatibility constraints;
- design of an information energy based control capable to discriminate the states in population set using both chromosome structure and objective function values.

REFERENCES

[Bru98]Brudaru, O., Fuzzy Compatibility Constraints in Assembly Line Balancing, Proceedings of 6th European Congress on Intelligent Techniques & Soft Computing, Aachen, Germany, September. 7-10, 1998, H.-J. Zimmermann (ed.), vol.3, Verlag Mainz, Aachen, pp. 1651-1655.

[BruB96]Brudaru, O., Belous, V., *Assembly Line Balancing with Fuzzy Execution Times*, Proceedings of EUFIT'96 - Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September 2-5, 1996, vol. 3, Verlag Mainz, 1996, pp. 1966-1971.

[BruBR97]Brudaru, O., Belous, V., Rusu, C., *Assembly Line Balancing with Fuzzy Execution Times and Mixed Models*, Proceedings of ISFL'97 - Second International Symposium on Fuzzy Logic and Application, Zurich, February 12-14,1997, pp. 158-164.

[Das95]Dasgupta, D., Incorporating Redundancy and Gene Activation Mechanism in Genetic Search for adapting to Non-Stationary Environments, Practical Handbook of Genetic Algorithms: New Frontiers, L. Chambers (ed.), CRC Press, New York, 1995.

[HorS82]Horowitz, E., Sahni, S, *Fundamental of Data Structures*, Computer Science Press,Inc. 1982.

[Mic94]Michalewiz, Z., Genetic Algorithms + Data Structures = Evolution Program, 2nd ed., Artificial Intelligence, Springer Verlag, Berlin, 1994.

[NegA95]Negoita, M. G., Agapie, A. High Performance GA based on Statistical Macroevolution, Proceedings of 3rd European Congress on Intelligent Techniques & Soft Computing, Aachen, Germany, August 28-31, 1995, H.-J. Zimmermann (ed.), vol.1, Verlag Mainz, Aachen, pp. 442-446.

[OniS79]Elements of Informational Statistics with Applications, Technical Editing House, Bucharest, 1979 (in Romanian).

[RusB90] Rusu, C., Brudaru, O., *Computer Assisted Design of Balanced Assembly Lines*, The Technical Publishing House, Bucharest, 1990.

[Rya95]Ryan, C., Niche and Species Formation in Genetic Algorithms, Practical Handbook of Genetic Algorithms: Applications, L. Chambers (ed.), CRC Press, New York, 1995.

[Tal89]Talbot,F.B., *A Comparative Evaluation of Heuristic Line Balancing Technique*, Management Science, vol. 32, no. 4, 1986, pp. 430-454.

[TsuG95]Tsumura, Y., En, M., Lie, Y., Cabot, E., *An Efficient Method for Solving Fuzzy Assembly Line Balancing Problem Using Genetic Algorithm*, Proceedings of the Third European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, August 28-31, 1995, vol. 1, Verlag Mainz, 1995, pp. 406-415.

[VemC95]Vemuri, V., Cedeno, W., Multi-Niche Crowding for Multi-Modal Search, Practical Handbook of Genetic Algorithms: New Frontiers, L. Chambers (ed.), CRC Press, New York, 1995.