THE ANNALS "DUNĂREA DE JOS" UNIVERSITY OF GALAȚI
FASCICLE V, TECHNOLOGIES IN MACHINE BUILDING, ISSN 1221-4566
2000

TREATING THE PRECEDENCE CONSTRAINTS IN TECHNOLOGICAL DISCRETE EVENT SYSTEMS

Dr.ing.Vasile MARINESCU, Ing. Sergiu MATEI
University "Dunarea de Jos" Galati

ABSTRACT

This paper proposes an algorithm, which is a special implementation of a stochastic descent technique, used to solve an optimum digraph-partitioning problem a general optimum criterion is considered. This kind of optimization problem has to manage with precedence and feasibility constraints. The proposed algorithm converges with probability 1 to an optimal solution.

1. INTRODUCTION

In this paper we propose a special implementation of stochastic descent technique called "Kangaroo" for solving a digraph-partitioning problem. Stochastic descent techniques are already used for solving such problems.

In order to define the "neighborhood" functions, elementary shift operators are introduced. They allow an efficient way to meet the partition and the precedence constraints. The feasibility constraint is treated implicitly, because the unfeasible solutions are penalized by the objective function. The accessibility of any optimal solution from any initial solution is proved. This is the main property, which guarantees the convergence with probability 1 of the proposed algorithm to an optimal solution.

This algorithm was already used to solve a tasks-to-workstations problem.

2. PARTITIONING PROBLEM WITH PRECEDENCE CONSTRAINTS

We consider the set Q = $\{t_1, t_2,...,t_N\}$ made up of N elements called tasks and we are looking for a partition formed by M subsets: W_i, i=1,...,M. We shall call workstation a peculiar subset.

Some precedence constraints between tasks are involved by a specific problem. So, the task set is partially ordered by a precedence relation described by a digraph G=(Q, U), $U \subset Q \times Q$. Nodes represent generic tasks and arrows represent precedence relations between tasks. An arrow from node n_1 to node n_2 indicates that task n_1 must be completed before task n_2 begins.

The time needed by the workstation W_i to perform the task t_j is τ_{ij}. It is possible that $\tau_{ij} = \infty$; that means that the workstation W_i cannot perform the task t_j.

Let P_i be the task set assigned to the workstation W_i. Obviously, the sets P_i, i=1,...,M determine a partition of the task set Q. Such a partitions is denoted by u:

$$u = \{P_1, P_2,...,P_M\}, \text{ with } P_i \subset Q, i = 1,...,M$$

A partition has to meet the precedence constraints, i.e.

$$\forall t_{j_1}, t_{j_2} \in Q, j_1 \neq j_2, \text{with } t_{j_1} \in P_{i_1}, t_{j_2} \in P_{i_2}$$

$$(t_{j_1} \text{ precedes } t_{j_2} \text{ in G}) \Rightarrow i_1 \leq i_2,$$

Obviously, an assignment has also to meet the feasibility constraint, i.e.

$$\tau_{ij} < \infty, \forall i \in \{1,...,M\}, \forall j \in P_i$$

A proper partition u meets the precedence and feasibility constraints. Generally, there are many proper partitions. One may choose the assignment that minimizes an objective function f(u) :

$$\min_u f(u) \qquad [1]$$

Two possible optimality criteria are given here after

$$f_1(u) = t_c(u) = \max_{i=1,...,M} \sum_{j \in P_i} \tau_{ij} \qquad [2]$$

$$f_2(u) = \sqrt{\sum_{i=1}^{M} (S_{max} - S_i)^2} \qquad [3]$$

where

$$S_{max} = t_C(u) \; ; \; S_i = \sum_{j \in P_i} \tau_{ij}$$

This problem is obviously NP-hard, because it is a graph partitioning which satisfies the optimality criterion [1].
One may notice that, if the processing time of a task t_j does not depend on the assigned workstation W_i, that is:

$\tau_{ij} = \tau_j$,

and $f(u)=f_2(u)$, we have the same optimality criteria as for the SMALB problem.
In order to emphasize the generality of the proposed algorithm, generally in the next sections we shall not set the objective function.

3. A GENERAL STOCHASTIC DESCENT TECHNIQUE

An algorithm, which guarantees a global minimum for our problem, should have an exponential computational complexity. Hence sub-optimal solutions are generally preferred to optimal solutions. That is why we propose to use an approximation technique based on stochastic descent, called "Kangaroo", inspired by the simulated annealing method, but having a quite different searching strategy.
As in the case of simulated annealing, the "Kangaroo" method is implemented by an iterative procedure, which minimizes an objective function f(u). A current solution u of the considered problem is replaced by a better one situated in its neighborhood N(u), using a random selection. If a new improvement is no longer possible, a "jump" procedure is performed, in order to escape from the attraction of a local minimum. This procedure can use a different neighborhood definition N' (u) for the random jumping.
The "Kangaroo" algorithm is described hereafter.

algorithm **Kangaroo**
begin
Initialize A;
Choose an initial solution u;
  c ← 1 ; u* ← u ;
  repeat
  if c<A *then* **descent**(u, u*, c) ;
    else **jump**(u, u*, c) ;
  until "stop criteria";
end ;
procedure **descent**(u, u*, c)
begin
Random generation of v in N(u) ;
  If f(v) ≤ f(u)
    then {*If* f(v)<f(u) *then* {c ← 0 ;
      if f(v)<f(u*) *then* u* ← v}
      u ← v ;}
  c ← c +1 ;
end **descent** ;

procedure **jump**(u, u*, c)
  Random generation of v in N'(u);
  If f(u) ≠ f(v) *then* {c ← 0;
    if f(v) < f(u*) *then* u* ← v}
  c ← c+1;
  u ← v;
end **jump**;

The parameter A is the maximum number of iterations without improvement of the current solution and u* is the best solution found until the current iteration. The variable c counts the number of iterations between two improvements of the objective function.
The stop criterion is either a maximum iteration number or a bottom bound of the objective function f(.).
If some conditions are fulfilled, then the best solution u* converges with probability 1 to a global minimum.

4. NEIGHBORHOOD'S DEFINITION AND PRECEDENCE CONSTRAINTS

The current solution u of the iterative process is a proper partition. Hence, it must be a partition of task set Q meeting the precedence constraints described by the precedence graph G.
The feasibility constraint will be treated implicitly, because a partition u, which does not meet this constraint, is penalized by the objective function.
If $\tau_{ij} = \infty$ for some pairs (i, j), we have to choose the form of objective function such that $f(u)=\infty$. Such is the case with the objective functions [2] and [3]. So, a non-feasible partition will never be assigned to u*, but it is accepted as a current solution of the iterative process, in order to assure the convergence of the algorithm.
An important advantage of the "Kangaroo" method is the fact that it works even for objective function with infinite values. This is not the case, for example, with simulated annealing method.
The procedures **descent** and **jump** use, in principle, two different definitions for the neighborhoods N(u)

and N'(u). But N(u) and N'(u) may be the same in some implementations of the "Kangaroo" algorithm. Firstly, we shall consider that :

$N(u)=N'(u)$.

Hereafter, we propose a specific definition of the neighborhoods N(u) and N'(u) for the partition problem.

When one uses a deterministic optimization algorithm (like as exhaustive search, dynamic programming, branch and bound,...), the partition constraint is the reason of the exponential complexity. Families of task set, that are not partitions, are generated as intermediary states and partition tests are necessary. The proposed algorithm generates only partitions as intermediary solutions, because N(u) contains only partitions of the task set Q. With the notation of section 2, a partition is:

$u=\{P_1, P_2,..., P_M\}$.

Let us consider a shift-left operator:

$$(u, t_j) \xrightarrow{shl} u', \qquad [4]$$

which generates a partition u' when it is applied to a partition u for a task t_j. This operator moves the task $t_j \in P_{io}$, $i_o>1$ from the set P_{io} to the set P_{io}-1. If we denote $u'=\{P'_1, P'_2,...,P'_M\}$ it holds:

$P'_{io-1}=P_{io-1} \cup \{t_j\}$; $P'_{io}=P_{io} - \{t_j\}$

$P'_i = P_i$, *for* $i \neq i_o-1$ *and* $i \neq i_o$;

If a partition u meets the precedence constraints and if all the direct predecessors of the task t_j belong to sets P_i, $i<i_o$, then u' will also meet the precedence constraints. Moreover, u' is obviously a partition of Q. If this condition is not met, i.e. if there is a direct predecessor of the task t_j belonging to P_{io}, we consider that the shift-left operator is not well defined

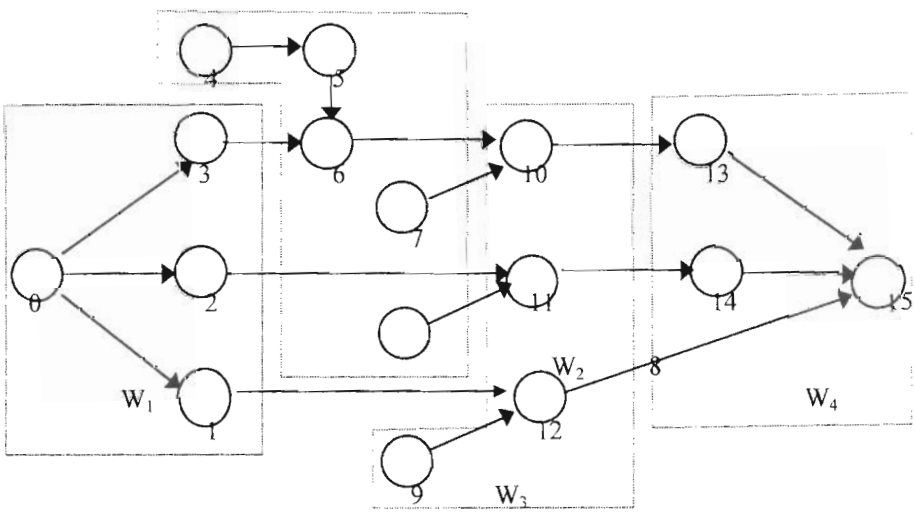

Figure 1. Example of precedence graph and partition

For example, an assembly line with 4 workstations has to process the tasks described by the superimposed graph presented in fig.1. Let us consider the assignment corresponding to the partition

$u=\{P_1, P_2, P_3, P_4\}$

with

$P_1=\{0, 1, 2, 3\}$, $P_2=\{4, 5, 6, 7, 8\}$,

$P_3=\{9, 10, 11, 12\}$, $P_4=\{13, 14, 15\}$.

The task 10 can be shifted left from workstation 3 to workstation 2, because this moving meets the precedence constraints, i.e. tasks 6 and 7 belong to workstation 2. The resulting partition is:

$u'=\{\{0,1,2,3\}$,

$\{4,5,6,7,8,10\},\{9,11,12\},\{13,14,15\}\}$,

$u' \in N(u)$.

For the task 6 the shift left operator is not well defined.

In the same way, a shift-right operator,

$$(u, t_j) \xrightarrow{shr} u', \qquad [5]$$

can be defined as below:

$t_j \in P_{io}$, $i_o<M$; $P'_{io+1}=P_{io+1} \cup \{t_j\}$; $P'_{io}=P_{io} - \{t_j\}$

$P'_i = P_i$, *for* $i \neq i_o+1$ *and* $i \neq i_o$;

If partition u meets the precedence constraints and if all the direct successors of the task t_j belong to sets P_i, $i>i_o$, then u' will also meet the precedence constraints. Moreover, u' is obviously a partition of Q. If there is a direct successor of the task t_j belonging to P_{io}, we consider that the shift-right operator is not well defined.

From now on, a shift operator is either a shift-left, or a shift-right operator. To apply such an operator to a partition u for a task t_j, one has to verify the constraints concerning the direct predecessors or the direct successors of t_j.

The main idea of the proposed algorithm is that the next solution is obtained from the current solution by a well-defined shift operator. *If the initial partition of the iterative process meets the precedence constraints, then the algorithm will generate only partitions meeting these constraints, without additional tests.* Hence, this algorithm will have a weak computational complexity.

In the current partition u, there are several tasks which can be shifted. The proposed algorithm forms a list L_u composed by these tasks. A task may be present two times in L_u, when it can be shifted left and right (the both operators being well defined). The algorithm constructs also the list D_u, which contains the index of the corresponding destination workstation.

For example, if u is the partition illustrated in fig. 1, it holds

$$L_u = [1,2,3,\ 4,\ 7,\ 8,\ 6,\ 7,8,9,10,\ 11,10,11,12,13,14]$$
$$D_u = [2,\ 2,\ 2,\ 1,\ 1,\ 1,\ 3,\ 3,\ 3,\ 2,\ 2,\ 2,\ \ 4,\ 4,\ 4,\ 3,\ 3]$$

So, the task $L_u(k)$ can be shifted from its workstation-to-workstation $D_u(k)$, $1 \le k \le \text{card}(L_u)$. The neighbourhood N'(u) is the set of partitions u' obtained by shifting all tasks belonging to L_u :

$$N'(u) = \left\{ u' \,\middle|\, (u,\ L_u(k)) \to u',\ 1 \le k \le \text{card}(L_u) \right\}$$

$$N(u)=N'(u) \qquad [6]$$

So, to generate a partition $u' \in N(u)$ is equivalent to choose a task in the list L_u. The random selection of the task that will be shifted is done in accordance with a uniform distribution law.

With the elements described above, the **descent** procedure for our assignment problem becomes:

procedure **descent_TWA**(u, u*, c)
begin

- Construct the list L_u of tasks which can be shift and the corresponding D_u list ;
//possible integration of heuristic elements in order to reduce the length of L_u ;
- Random selection of a task t_j in L_u;
- Construction of the partition v by shifting t_j : $(u, t_j) \to v$

If f(v)≤f(u)
then {*If* f(v)<f(u) *then* {c←0 ;
if f(v)<f(u*) *then* u* ← v }
u ← v }
c← c+1 ;
end **descent_TWA**;

5. CONVERGENCE OF THE ALGORITHM

The series $\{u_n\}$ of the current solutions in the iterative process, realized by a particular execution of the presented algorithm, is a realization of a Markov chain. At iteration n, u^*_n is the best-encountered solution. One may consider the series $\{u^*_n\}$ of the best solutions of the iterative process; that is also a realization of Markov chain.

In this section, we shall prove that the stochastic process (u^*_n) converges with probability 1 to a global minimum. The main point is the accessibility property of the neighborhood function N'(u).

Let X be the partition set meeting the precedence constraint (i.e. the proper assignment set) and let X_{min} be the optimal partition set (i.e. the set of proper assignments which minimize the objective function). Obviously, X and X_{min} are finite sets.

Proposition 1 (accessibility): If u_i, $u_f \in X$ there is a finite sequence of tasks t_1, t_2,..., t_k such that

$$u_i \xrightarrow{sh(u_i, t_1)} u_1 \xrightarrow{sh(u_1, t_2)} u_2 \cdots \cdots u_{p-1} \xrightarrow{sh(u_{p-1}, t_p)} u_p = u_f ,$$

where sh(.) is either shl(.) or shr(.) operator.

Notice that the probability of the transfer mentioned in proposition 1 is not equal to zero, because the probability of a single transition is determined by a uniform random selection over a finite number of possibilities.

Sometimes, the partitions obtained in the iterative process are not feasible assignments. We must consider such partitions, in order to assure the accessibility property of the neighborhood function N'(u).

Proposition 2: The TWA algorithm with the neighborhood functions N'(.) and N(.) defined by equation [6] is convergent with probability 1 to an optimal solution.

A very important aspect is the fact that the convergence with probability 1 requires accessibility in the sense of neighborhood N'(.), which is used by jump procedure. Hence, deterministic heuristics may be integrated in descent procedure in order to guide the search of an optimum solution. This is equivalent to modifying the definition of neighborhood N(u) such that

$N(u) \subset N'(u)$.

If the property $\forall u \in X$, $u \in N(u)$ holds, the algorithm not only keeps its convergence property but the convergence speed is improved.

6. COMPUTATIONAL ASPECTS

The iterative procedure begins with any initial partition. An implementation of our algorithm was realized in accordance with the elements presented before. It is a general variant of the algorithm, because any objective function can be used.
We can improve the convergence speed of the partition algorithm by adding some deterministic heuristics in descent procedure. So, the definition of the neighborhood N(u) will be modified in order to guide the search of an optimum solution. Obviously, the heuristics added in descent procedure depend on the objective function.
For example, if the objective function is given by equation [2] or [3], the list L_u may contain only the tasks belonging to the maximum work content workstations. Intuitively, to decrease the line's cycle time, one have to shift a task belonging to such a workstation. In a second variant of the proposed algorithm, we have implemented this heuristic. In the same time, the complexity of each iteration has been improved by reducing the number of tasks of the list L_u. The two variants of assignment algorithm have been coded in C++ and executed on a PC workstation.
An example of the objective function's evolution during the iterative process is presented in Figure 2.

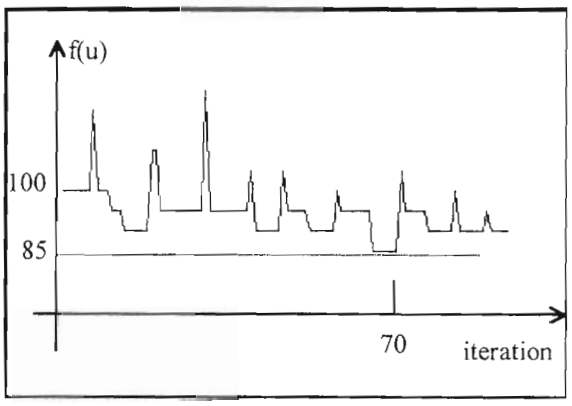

Figure 2. Evolution of the cycle time in 100 iterations

We have considered the objective function given by equation [3]. For the same value of parameter A, the second variant of the proposed algorithm is obviously faster then the first one. That means that a solution with the same value of the objective function is found, generally, in a smaller number of iterations. The results presented hereafter are obtained with this variant of the proposed algorithm.
Table I provides some computational results obtained for a set of problems whose optimal solutions are known. When the algorithm finds a good solution u_o instead of the optimal one u*, the approximation error is

$$\frac{f(u_O) - f(u^*)}{f(u^*)}.$$

Concerning the choice of the number of iterations without improvement of the objective function, the computational experience proved that a good value for A is:

$$A = a.\left[\frac{N}{M}\right], a \in \{1, 2\}$$ ([x] is the greater integer less or equal to x)

N	M	K	iteration	approximation error
32	8	3	9-17	5.8%
64	16	3	11-34	5.8%
128	16	6	9-164	5.8%
128	16	6	13-304	2.9%

Table I Computational results

For all the problems, the second variant of the algorithm has found the optimal solutions in an acceptable iteration number. For example, the optimal solution of a large size problem with 128 tasks, 16 workstations were found in 16394 iterations. But, after up to 164 iterations, the best solution is at 5.8% from an optimal one.
The tests have proved a good behavior of the proposed assignment algorithm for real problems concerning large assembly lines.

7. CONCLUSION

We have proposed an algorithm for solving a partitioning problem, whatever the optimal criteria are. It is a particular implementation of a stochastic descent technique.
The definition of the "neighborhood" function based on elementary shift operations allowed a very efficient way for treating the partition and precedence constraints. If the initial solution is a proper partition, the test of partition and precedence constraints is no longer necessary. Because the algorithm manages with objective functions having infinite values, the feasibility constraint is treated implicitly. Hence, the algorithm's complexity is very good.
The solution space has the property of accessibility in the sense of "neighborhood" function. The algorithm converges with probability 1 to a global minimum. This is not the case with other heuristic search methods. Its convergence speed can be improved by adding some deterministic heuristics in the descent procedure.

The proposed algorithm was tested for solving assignment problems for large assembly systems. We have adopted two objective functions used also by ALB problem. The results have that this approach is realistic and efficient.

References

[1] **Falkenauer E.,** *«A Hybrid Grouping Genetic Algorithms for Bin Packing»*, Journal of Heuristics, 2 : 5 - 30 Kluwer Academic Publisher, 1996

[2] **Fleury G.** *Méthodes stochastiques et déterministes pour les problèmes NP - difficiles*, PhD Thesis , Université Blaise Pascale , Clermont - Ferrand, 1993

[3] **Fleury G.**,*«Applications des méthodes stochastiques inspirées du recuit simulé à des problèmes d'ordonnancement»*, Automatique Productique Informatique Industrielle, Vol. 29 - no 4 -5/1995, p445-470.

[4]**S. Ghosh, R.J. Gagnon** : *«A comprehensive literature review analysis of the design, balancing and scheduling of assembly systems»*, International Journal of Production and Research, Vol.27, no. 4, 1989, p637-670.

[5] **Gutjahr A. and Nemhauser G.**: *«An Algorithm for the Line Balancing Problem»*, Management Science , 11(2), pp 308-315, 1964

[6] **Holmes C.A.**, *Equipment Selection and Task Assignment for Multi-product Assembly System Design*, Master thesis, M.I.T, USA, 1987

[7] **Mînzu V., Henrioud J. M** : *"Tasks -to -workstations assignment in assembly system using a stochastic algorithm"*, IFAC Conference "Control of Industrial Systems", 20-22 May 1997, Belfort FRANCE

[8] **Mînzu V., Henrioud J. M** :*"Assignment Stochastic Algorithm in Multi-Product Assembly Lines"*, IEEE International Symposium on Assembly and Task Planning, August 7-9, 1997, Marina de Rey, California, USA.

[9] **Mînzu V., Henrioud J. M** : *"A general and convergent stochastic descent algorithm for tasks to workstations assignment"* , INES'98, IEEE International Conference on Intelligent Engineering Systems, september 17-19, 1998 , Vienna, AUSTRIA

[10] **Rekiek B., Falkenauer E. and Delchambre A** ,*«Multi-Product Resource Planning »*, Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning, USA, Marina del Ray, August 7-9, 1997, p115-121

[11] **Suresh G., Vinod V. V. and Sahu S.,** *«A genetic algorithm for assembly line balancing»*, Production Planning & Control, vol.7, no.1, 1996, p38-46

TRATAREA CONSTRÂNGERILOR DE PRECEDENȚĂ ÎN SISTEME TEHNOLOGICE CU ENEVIMENTE DISCRETE

(Rezumat)

Această lucrare propune un algoritm cu ajutorul căruia este implementată o tehnică stohastică descendenta utilizată în atingerea grafului optim în problema de partiționare cu criteriu de optim general considerat. Acest tip de problema de optimizare a fost generată de constângerile de precedența și fezabilitate. Convergenta algoritmului propus este cu probabilitate 1 pentru solutia optimală.

TRAITEMENT DU PRECEDENCE CONSTRAINTS DANS LES SYSTÈMES DISCRETS TECHNOLOGIQUES D'ÉVÉNEMENT

(Résume)

Cet article propose un algorithme qui est une mise en place spéciale d'une technique stochastique de descente, employé pour résoudre un problème de division A de digraphe optimum le critère optimum que général est considéré. Ce genre de problème d'optimisation doit contrôler avec des contraintes de priorité et de praticabilité. L'algorithme proposé converge avec la probabilité 1 à une solution optimale.