RAIRO. RECHERCHE OPÉRATIONNELLE

R. TADEI
F. DELLA CROCE
G. MENGA

Advanced search techniques for the job shop problem : a comparison

RAIRO. Recherche opérationnelle, tome 29, n° 2 (1995), p. 179-194

<http://www.numdam.org/item?id=RO_1995__29_2_179_0>

Recherche opérationnelle/Operations Research
(vol. 29, n° 2, 1995, p. 179 à 194)

ADVANCED SEARCH TECHNIQUES FOR THE JOB SHOP PROBLEM: A COMPARISON (*)

by R. TADEI ([1]), F. DELLA CROCE ([1]) and G. MENGA ([1])

Communicated by Franco GIANNESSI

Abstract. – *Recently several new heuristic search techniques have been used in machine scheduling. In this paper we make some methodological and computational comparisons between such techniques as bottleneck-based algorithms (Shifting Bottleneck), deterministic neighborhood search procedures (Tabu Search), random oriented local search techniques (Simulated Annealing and Genetic Algorithms) and Lagrangian relaxation approaches.*

Keywords: Shifting Bottleneck, Tabu Search, Simulated Annealing, Genetic Algorithms, Lagrangian Relaxation.

Résumé. – *Récemment, des nouvelles méthodes de recherche heuristique ont été appliquées dans le domaine de l'ordonnancement des machines. Dans cet article on présente des comparaisons théoriques et de calculabilité entre les méthodes suivantes : des algorithmes fondés sur l'approche « goulot d'étranglement » (Shifting Bottleneck), des procédures pour la recherche locale déterministe (Recherche Tabu) et stochastique (Recuit Simulé et Algorithmes Génétiques), et enfin l'approche de la relaxation Lagrangienne.*

Mots clés : *Shifting Bottleneck,* Recherche Tabu, Recuit Simulé, Algorithmes Génétiques, Relaxation Lagrangienne.

1. INTRODUCTION

This paper deals with the job shop scheduling problem where n jobs are to be processed on m machines with the objective of minimizing a function of the completion times. Let us define $p_{i,j}$ as the processing time of job i on machine j (*i.e.* the processing time of operation $o_{i,j}$), $c_{i,j}$, as the completion time of $o_{i,j}$, d_i as the due date (if present) of job i and r_i as the release time (if present) of job i. The constraints are such that each machine can process only one job at a time, no two jobs can work on the same machine contemporaneously and no job can be interrupted while being

(*) Received March 1994.

([1]) D.A.I., Politecnico di Torino, C. Duca degli Abruzzi 24, 10129 Torino (Italy), e-mail tadei@polito.it

Recherche opérationnelle/Operations Research, 0399-0559/95/02/$ 4.00

processed (preemption is not allowed). The most commonly used objective function is the makespan, *i.e.* the maximum completion time. It is useful to represent this problem on a disjunctive graph (*see* Roy and Sussmann [23]) $G = (N, A, E)$ where each node $n \in N$ represents an operation, each arc $a \in A$ represents a precedence constraint between two operations belonging to the same job and each edge $e \in E$ corresponds to a disjunctive constraint of the two operations belonging to the same machine. With the makespan as objective function, the problem is strongly NP-hard, therefore different heuristics have been proposed in recent years. Panwalker and Iskander [21] mention in their survey most of the practical priority rules on which many of *List Scheduler* algorithms are based. Adams, Balas and Zawack [1] introduced a sophisticated heuristic, the *Shifting Bottleneck Procedure* (**SBP**), which minimizes the makespan in a job shop by the iterative choice of the bottleneck machine and solution of the related subproblem. Also based on the bottleneck philosophy is the *Bottleneck Dynamics* approach (*see* [15, 17, 18]) in which the operations are given priorities and the resulting schedule acts as a feedback for updating those priorities in order to iteratively recompute improved schedules. Neighborhood techniques have been proposed by Dell'Amico, Trubian [6] and Nowicki, Smutnicki [20] in their papers based on *Tabu Search* (**TS**) and by Van Laarhoven, Aarts and Lenstra [24] who applied the *Simulated Annealing* technique (**SA**). *Genetic algorithms* were proposed by Della Croce, Tadei and Volta [5] (**GA**) and by Nakano and Yamada [19]. Another attemp has been the application of a Lagrangian relaxation approach to the job shop problem. The *Cellular Control* approach (**CC**) proposed by Della Croce *et al.* [4] relaxed the temporal consistency constraints of materials between modules, *i.e.* the precedence constraints among the operations belonging to the same job by using augmented Lagrangian multipliers. Hoitomt *et al.* [13] also used augmented Lagrangian multipliers relaxing two sets of constraints: the precedence constraints among operations belonging to the same job and the disjunctive constraints among different operations operating on the same machine. Many of these works (SBP [1], CC [4], GA [5], TS-1 [6], TS-2 [20] and SA [24]) used the same input data (*see* Lawrence [14]) for the job shop problem and applied it to the same objective function, *i.e.* the makespan. The aim of this paper, however, is not to survey the search techniques for the job shop problem and their applications but to carry out a methodological and computational comparison of these methods, pointing out both the differences and the similarities and extending this analysis to other objective functions, e.g. total completion time, total tardiness etc.

The paper is organized as follows. In section 2 we briefly outline and comment the above mentioned heuristics. In section 3 a computational comparison is introduced in terms of quality of the solution and computing times and a methodological comparison is examined more fully pointing out similarities, differences and links among the various methods. Section 4 concludes the paper with final remarks.

2. SKETCHING THE VARIOUS HEURISTICS

2.1. Shifting Bottleneck Procedure

The main concepts of the SBP are the following: first, the machines are sequenced consecutively one at a time and, to do this, a one-machine scheduling problem, which is a relaxation of the original problem, is solved to optimality for each machine not sequenced yet. Then each solution is compared with the others and all the machines are ranked on the basis of their solution. The bottleneck machine is the one corresponding to the worst solution. Each time the bottleneck machine is sequenced, every previously sequenced machine susceptible to improvement is reoptimized by solving a one-machine problem again (this is the so-called SBI part of the procedure). The special feature of this method is that the iteratively computed one-machine problem is the maximum lateness problem subject to different release dates $(1/r_j/\max L_j)$ to which, though strongly NP-hard [9], Carlier's method [2] provides an exact and rapid solution. As mentioned by the authors, while solving the one-machine problem the selection associated with the optimal solution may occasionally create a cycle in the resulting disjunctive graph. To avoid this, it is sufficient to take into account precedence relations between the operations when solving the one-machine problem. A partial enumeration extension (SBII), which is a beam search applied to the original SBI procedure, is then developed: each time the bottleneck is searched for, the top few candidates are saved and each defines a branch on the tree.

2.2. Simulated Annealing

Simulated Annealing is a random-oriented local search technique that was introduced as an analogue from statistical physics of the computer simulation of the annealing process of a hot material until the minimum energy state is reached. In the field of *machine sequencing* it can be considered a generalization of a classic neighborhood search such as pairwise interchanges. Given an objective function and the neighborhood structure, the

classic search moves from one solution to its neighbor only if the objective function is improved. One a local minimum is reached, the search ends. SA can be viewed as an attempt to allow the system to reach local minima but to permit it to escape from them by accepting moves which worsen the objective function. Van Laarhoven, Aarts and Lenstra [24] applied SA to the job shop problem in the following way:

• The neighborhood chosen is the set S of all sequences obtained by reversing one of the arcs on a critical path in the corresponding disjunctive graph. This is due to the following properties of the problem:

– if $s \in S$ is the current solution, then reversing one of the arcs on a critical path of s can never lead to an unfeasible solution;

– if the reversal of an arc of the current solution s, that does not belong to a critical path, leads to a feasible solution s^*, then a critical path in s^* cannot be shorther then a critical path in s.

• The probability of accepting the transition from solution s to solution s^* is given by

$$P = \min (1,\ e^{c^{-1}\,[-C(s^*)-C(s)]})$$

where $C(s)$ is the objective function of solution s and c is the control parameter which starts from a given initial value and progressively decreases until the system reaches its stability in a local minimum.

The slope of the decrease of c is proportional to the inverse of the *cooling* parameter which belongs to the range]0, 1]. The smaller this parameter is, the more slowly the system reaches a stable state.

2.3. Tabu Search

Tabu search is a neighborhood search technique originally introduced by Glover [11, 12]. Starting from a feasible schedule, the search moves from one solution to another choosing the best not forbidden element (tabu-move) in its neighborhood.

Moves with certain attributes may be forbidden in order to prevent cycling and guide the search towards unexplored regions. The need for such restrictions derives, from the fact that, without them, the method could choose the "best" move away from a local optimum and then fall back at the following step into the previous local optimum. Dell'Amico and Trubian [6] applied to job shop scheduling a Tabu Search procedure (TS-1) working in the following way.

• A greedy (so-called "bi-directional") algorithm is applied to find a starting feasible solution.

• The neighborhood structure is strictly related to the above observation on the disjunctive graph related to the job shop problem in which the makespan is the objective function. Only moves that involve reversing at least one of the arcs on any critical path of the current solution are therefore considered.

• In order to establish whether a candidate move is allowed, all its single swaps are considered. Swapping arc (i, j) means to reverse the direction of the arc, *i.e.* choose the arc (j, i). If at least one component swap is currently tabu, the move is forbidden.

• Every time a move is accepted, all its component swaps are included in the tabu list.

• The tabu list has a finite dimension and is handled through a *First In First Out* policy, *i.e.* at each iteration the algorithm memorizes a new set of swaps and forgets the oldest ones.

Nowicki and Smutnicki [20] reached better results than [6] by applying a Tabu Search technique (TS-2) whose main differences from [6] are the following:

• The starting solution is obtained with a fast heuristic based on an insertion technique with complexity $O(n^3m^2)$ where n and m are the number of jobs and machines, respectively (*see* Werner and Winkler [25]).

• The neighborhood is much smaller: just one critical path is considered.

2.4. Genetic Algorithms

Genetic Algorithms are an optimization technique for functions defined over finite domains. Their name refers to their imitation of natural evolutionary processes. The application of a GA in optimization problems is performed in the following way.

• The search space of all possible solutions is mapped onto a set of finite strings (chromosomes): the alphabet of symbols is of finite dimension.

• For sequencing problems the chromosomes are usually the permutation of a basic string: if there are k jobs on a machine, a permutation of the numbers from 1 to k is a chromosome which codifies a possible solution.

• A set of solutions is selected (usually at random) and this constitutes the initial *population*, *i.e.* the first generation.

• A *fitness* is computed for each of the individuals. *Fitness* is a parameter which reflects the quality of the solution represented by the individual.

• A set of individuals is selected for reproduction. The selection takes place at random but with a probability proportional to fitness.

• From the selected set some progeny is generated by applying different genetic operators.

• Some of the older individuals are eliminated from the population to allow the entrance of the new ones.

• The new individuals' fitness is computed and those individuals are included in the population. This step marks the end of a generation.

• If the iteration threshold is reached, THEN the process is finished and the best chromosome is returned, else the system performs a new reproduction.

The specific features of the genetic algorithm (GA) proposed by Della Croce, Tadei and Volta [5] for the job shop problem are the following:

• Two genetic operators are applied: Crossover and Mutation. The Crossover is a variant called LOX (Linear Ordered Crossover) introduced by Falkenauer and Bouffoix [7] for machine scheduling problems. The Mutation is defined as follows: two elements of the string are chosen at random and their positions are exchanged.

• A chromosome is formed of several subchromosomes (strings permutations as above), one for each machine. The actual schedule is deduced from the chromosome through a simulation. When a number of different operations is available for processing on the same machine the conflict is solved using a *preference list*, but the machine lies idle if another operation with greater priority is due to become available in a short time.

• Given that the chromosome does not describe an actual schedule and that different chromosomes can produce equal schedules, a partition of the chromosomes space C into classes of equivalence is introduced. Let S be the space of schedules and $F : C \rightarrow S$ be the function which generates a schedule starting from a chromosome. It is shown that in each equivalence class there is one chromosome which is an "eigenvector" of F with "eigenvalue" equal to 1. By assuming that the eigenvector is the chromosome which best transmits the specific characters of the class to the descendants, the algorithm is consequently modified in such a way as to substitute each new chromosome with the eigenvector of its own equivalence class (*updating step*).

2.5. The Lagrangian relaxation approach

The Cellular Control algorithm (CC) proposed by Della Croce *et al.* [4] uses a Lagrangian relaxation approach for iteratively computing a heuristic solution. Let us consider the following formulation of the job shop problem (***JSP***), where $c_{i,j}$, $p_{i,j}$ and d_i are the completion time of job i on machine j, the processing time of job i on machine j and the due date of job i, respectively. We assume that when we use terms c_{i,j^*} and p_{i,j^*}, the subscript j^* represents the j-th machine according to the routing of job i.

Problem JSP:

$$\min_{c_{i,j^*}} \left(\max_i c_{i,m^*}\right)$$

subject to

$$c_{i,j^*} + p_{i,(j+1)^*} - c_{i,(j+1)^*} \leq 0 \quad i = 1, \ldots, n \quad j = 0, \ldots, m \qquad \text{(A)}$$

with

$$c_{i,0^*} = 0 \quad c_{i,(m+1)^*} = d_i \quad p_{i,(m+1)^*} = 0$$

$$\begin{aligned} &c_{i,j} - c_{l,j} \geq p_{i,j} \bigvee c_{l,j} - c_{i,j} \geq p_{l,j} \\ &i, l = 1, \ldots, n \quad j = 1, \ldots, m \quad i \neq j \end{aligned} \qquad \text{(B)}$$

To solve *JSP*, given a known upper bound of the makespan UB_{mk}, a search for a feasible solution is performed; UB_{mk} is then decreased and the search for a feasible solution is repeated. Those steps are performed iteratively until the problem becomes unfeasible.

This procedure can be summarized with the following

Algorithm (CC):

1. Initialize UB_{mk} and set $d_i = UB_{mk}$, $\forall i$

2. Find a feasible schedule with respect to constraints (A) and (B), *i.e.* solve

Problem **JSP1**:

min k (with k constant, e.g. k = 0) subject to constraints (A) and (B)

IF no solution is found END, ELSE decrease UB_{mk} and d_i and GOTO 2.

In order to solve problem *JSP*1, the authors decided to use an *Augmented Lagrangian Relaxation* approach (*see* [16] and [22]) such that the primal

problem $JSP1_{PR}$, relaxing the constraints (A) using the non-negative Lagrange multipliers λ_{i,j^*} and the positive penalty coefficient ρ, became

Problem $\mathbf{JSP1}_{PR}$:

$$\min_{c_{i,j^*}} \sum_{i=1}^{n} \sum_{j^*=0}^{m} [\lambda_{i,j^*} (c_{i,j^*} + p_{i,(j+1)^*} - c_{i,(j+1)^*}) + \rho (c_{i,j^*} + p_{i,(j+1)^*} - c_{i,(j+1)^*})^2]$$

subject to:

$$\begin{gathered} c_{i,j} - c_{l,j} \geq p_{i,j} \bigvee c_{l,j} - c_{i,j} \geq p_{l,j} \\ i, l = 1, \ldots, n \quad j = 1, \ldots, m \quad i \neq j \quad \lambda_{i,j^*} \geq 0 \quad \rho > 0 \end{gathered} \tag{B}$$

By setting:

$$\alpha_{i,j^*} = \begin{cases} (\lambda_{i,j^*} - \lambda_{i,(j-1)^*} + 2\rho p_{i,(j+1)^*} \\ \quad - 2\rho c_{i,(j+1)^*} - 2\rho c_{i,(j-1)^*} - 2\rho p_{i,j^*}) \\ \qquad \text{if } j < m \\ (\lambda_{i,j^*} - \lambda_{i,(j-1)^*} \\ \quad - 2\rho d_i - 2\rho c_{i,(j-1)^*} - 2\rho p_{i,j^*}) \\ \qquad \text{if } j = m \end{cases}$$

the objective function for primal problem $JSP1_{PR}$ can be rewritten as:

$$\min_{c_{i,j^*}} L(c, \lambda, \rho) = \min_{c_{i,j^*}} \sum_{i=1}^{n} \sum_{j=1}^{m} (2\rho c_{i,j^*}^2 + \alpha_{i,j^*} c_{i,j^*})$$

Problem $JSP1_{PR}$ can be decomposed into m local problems $JSP1_{PR_j}$, one for each machine j (independently from the job routing):

$$\min_{c_{i,j}} \sum_{i=1}^{n} (2\rho c_{i,j}^2 + \alpha_{i,j} c_{i,j})$$

subject to:

$$\begin{gathered} c_{i,j} - c_{l,j} \geq p_{i,j} \bigvee c_{l,j} - c_{i,j} \geq p_{l,j} \\ i, l = 1, \ldots, n \quad j = 1, \ldots, m \quad i \neq j \end{gathered} \tag{B}$$

As $\alpha_{i,j}$ coefficients produce coupling between different machines due to the presence of terms $c_{i,(j+1)^*}$ and $c_{i,(j-1)^*}$, in order to enforce separability a Jacobi-type iterative technique bounded by a maximum number of iterations

is used here (although this technique may not find a global optimal solution for the problem). For each machine, a quadratic programming problem with a set of disjunctive constraints must be solved. The authors solved it heuristically by computing the unconstrained optimum and using the result to define the sequence of jobs according to the non-decreasing value of their completion times (obtained by calculating the unconstrained optimum). Given the sequence, the problem is solved by using a quadratic programming library package. Before carrying out the dual step, the feasibility of the resulting schedule is tested and, in the case of violations, a constructive algorithm is performed. In this algorithm the sequence represents the priority rule for the choice of the job to be processed among all the jobs ready to work on a particular machine. The dual problem is solved by using a subgradient procedure.

3. COMPARISON BETWEEN THE HEURISTICS

3.1. Computational results

All the algorithms presented in section 2 (except CC) were tested on 40 benchmarks proposed by Lawrence [14]. The results are summarized in Table 1 where we indicate with n the number of jobs, with m the number of machines. For each method the machine used, the average time required and the best solution obtained are presented. As far as the Tabu Search methods are concerned, the variability of the computation times among instances having the same size is due to the fact that one of the stopping criteria is the comparison between its current solution and the optimum of the problem (when known). If the values are coincident the algorithm ends. SBP uses only SBI procedure for problems LA6 to LA15 and LA31 to LA35. The cooling parameter for SA it set to 0.1 for problems 31 to 35 and to 0.01 for the other problems. Though it is hard to compare the performance of different types of computer, we can reasonably affirm that a PC 486-25 Mhz is practically twice as fast as a PC 386-33 Mhz which is comparable to a VAX 785 (though this depends on the number of contemporaneous users and the priority of the considered user). Notice that a "–" entry in the CPU time columns means that the method required less than a second to find the solution. From the results we evince that TS-2 performs generally best both in terms of computation time and quality of the solution. In general, however, the quality of the solution of all methods is fairly high and the computation times are reasonably short. The Cellular Control method (CC) is compared in Table 2 with the genetic algorithm for some job shop problems.

In this case the genetic procedure outperforms the cellular one (both for solution quality and computation time) but we should bear in mind that on a real shop-floor the information resulting from the cellular algorithm (e.g. the evolution of the objective function and the lagrangian multipliers) gives the final user a much greater understanding of the problem. A random-oriented heuristic (like GA) is normally seen by the final user as a black box which provides an output but no further information.

TABLE 1

Comparison between the heuristics on Lawrence's problems (CPU time in seconds)

Problem	n	m	OPT	SA (CPU)	TS-1 (CPU)	TS-2 (CPU)	SBP (CPU)	GA (CPU)
			(LB, UB)	VAX 785	PC 386-33 Mhz	PC 386DX	VAX 785	PC 486-25 Mhz
LA01	10	5	666	666 (123)	666 (–)	666 (–)	666 (1)	666 (282)
LA02	10	5	655	655 (117)	655 (18.8)	655 (8)	669 (6)	680 (284)
LA03	10	5	597	606 (129)	597 (21.6)	597 (11)	605 (32)	604 (281)
LA04	10	5	590	590 (121)	590 (32.2)	593 (–)	593 (23)	590 (278)
LA05	10	5	593	593 (118)	593 (–)	593 (–)	593 (–)	593 (275)
LA06	15	5	926	926 (286)	926 (–)	926 (–)	926 (1)	926 (473)
LA07	15	5	890	890 (376)	890 (–)	890 (–)	890 (1)	890 (462)
LA08	15	5	863	863 (292)	863 (–)	863 (–)	863 (2)	863 (457)
LA09	15	5	951	951 (283)	951 (–)	951 (–)	951 (–)	951 (467)
LA10	15	5	958	958 (243)	958 (–)	958 (–)	958 (–)	958 (459)
LA11	20	5	1,222	1,222 (627)	1,222 (–)	1,222 (–)	1,222 (1)	1,222 (717)
LA12	20	5	1,039	1,039 (655)	1,039 (–)	1,039 (–)	1,039 (–)	1,039 (726)
LA13	20	5	1,150	1,150 (564)	1,150 (–)	1,150 (–)	1,150 (1)	1,150 (733)
LA14	20	5	1,292	1,292 (462)	1,292 (–)	1,292 (–)	1,292 (–)	1,292 (699)
LA15	20	5	1,207	1,207 (736)	1,207 (1.2)	1,207 (–)	1,207 (2)	1,207 (689)
LA16	10	10	945	956 (686)	945 (97.4)	946 (64)	978 (120)	979 (637)
LA17	10	10	784	785 (720)	784 (21.7)	784 (3)	787 (96)	784 (628)
LA18	10	10	848	861 (673)	848 (63.1)	848 (66)	859 (112)	853 (642)
LA19	10	10	842	848 (830)	842 (103.8)	842 (60)	860 (120)	850 (651)
LA20	10	10	902	902 (667)	902 (71.7)	902 (150)	914 (144)	920 (640)
LA21	15	10	(1,040, 1,047)	1,063 (1,991)	1,048 (198.8)	1,055 (21)	1,084 (181)	1,097 (1,062)

Problem	n	m	OPT	SA (CPU)	TS-1 (CPU)	TS-2 (CPU)	SBP (CPU)	GA (CPU)
			(LB, UB)	VAX 785	PC 386-33 Mhz	PC 386DX	VAX 785	PC 486-25 Mhz
LA22	15	10	927	938 (2,163)	933 (191.4)	927 (9)	933 (210)	972 (1,020)
LA23	15	10	1,032	1,032 (2,093)	1,032 (1.8)	1,032 (1)	1,032 (113)	1,032 (1,063)
LA24	15	10	935	952 (2,098)	941 (181.8)	948 (184)	976 (217)	984 (1,045)
LA25	15	10	977	992 (2,133)	979 (191.7)	988 (155)	1,017 (215)	1,018 (1,052)
LA26	20	10	1,218	1,218 (4,342)	1,218 (22.1)	1,218 (16)	1,224 (372)	1,231 (1,542)
LA27	20	10	(1,235, 1,236)	1,269 (4,535)	1,242 (254.2)	1,259 (66)	1,291 (419)	1,308 (1,555)
LA28	20	10	1,216	1,224 (4,354)	1,216 (186.4)	1,216 (107)	1,250 (451)	1,300 (1,535)
LA29	20	10	(1,120, 1,160)	1,218 (4,408)	1,182 (281.3)	1,164 (493)	1,239 (446)	1,238 (1,550)
LA30	20	10	1,355	1,355 (3,956)	1,355 (10.4)	1,355 (2)	1,355 (276)	1,355 (1,537)
LA31	30	10	1,784	1,784 (1,517)	1,784 (1.5)	1,784 (1)	1,784 (19)	1,784 (2,762)
LA32	30	10	1,850	1,850 (1,752)	1,850 (1.7)	1,850 (–)	1,850 (15)	1,850 (2,744)
LA33	30	10	1,719	1,719 (1,880)	1,719 (1.3)	1,719 (–)	1,719 (14)	1,719 (2,738)
LA34	30	10	1,721	1,721 (1,886)	1,721 (4.6)	1,721 (4)	1,721 (11)	1,721 (2,755)
LA35	30	10	1,888	1,888 (1,668)	1,888 (–)	1,888 (1)	1,888 (11)	1,888 (2,770)
LA36	15	15	1,268	1,293 (5,346)	1,278 (238.4)	1,275 (623)	1,305 (268)	1,305 (1,880)
LA37	15	15	1,397	1,433 (5,287)	1,409 (242.2)	1,422 (443)	1,423 (419)	1,519 (1,872)
LA38	15	15	(1,171, 1,184)	1,215 (5,480)	1,203 (256.6)	1,209 (165)	1,255 (540)	1,273 (1,887)
LA39	15	15	1,233	1,248 (5,766)	1,242 (237.8)	1,235 (325)	1,273 (335)	1,315 (1,870)
LA40	15	15	1,233	1,234 (5,373)	1,233 (236.6)	1,234 (322)	1,269 (450)	1,278 (1,853)

3.2. The objective function

As far as the objective function is concerned some observations should be made:

• The Shifting Bottleneck procedure is tailored for situations where we take the makespan as objective function and cannot be easily generalized for other performance measures. With a different objective function it is not possible to provide rapid and exact solution to the Single Machine related subproblem such as, for instance, the single machine total tardiness problem with different release dates (*see* Chu [3]) which is much harder in practice than the single machine maximum lateness problem.

• Though in a different way, the papers relating to Simulated Annealing and Tabu Search are also tailored to the makespan as objective function:

TABLE 2
Comparison between Genetic Algorithm and Cellular Control on job shop instances (CPU time in seconds)

Problem	n	m	OPT	GA (CPU)	CC (CPU)
			(best UB)	PC 486-25 Mhz	VAX 9,000
G1	6	6	55	55 (223)	58 (7)
G2	6	6	65	65 (222)	67 (45)
G3	6	6	113	113 (220)	114 (136)
G4	10	10	930	946 (628)	980 (1,585)
G5	10	10	725	740 (630)	788 (129)
G6	10	10	848	854 (626)	909 (446)
G7	20	5	(1,178)	1,178 (675)	1,296 (2,593)
G8	20	5	(1,122)	1,122 (670)	1,225 (298)
G9	20	5	(949)	949 (661)	1,013 (149)
G10	5	20	1,322	1,328 (717)	1,329 (636)
G11	5	20	1,237	1,264 (712)	1,264 (2,034)
G12	5	20	1,188	1,204 (713)	1,204 (170)

the structure of the problem allows us to consider an inversion only of those couples of operations for which the corresponding arc in the associated *disjunctive graph* is on a critical path. Therefore, given the sequence, it is possible to automatically tighthen the neighborhood search and to use faster procedures to compute the schedule (which obviously reduces the total computation time and improves the quality of the results).

• As far Cellular Control is concerned, in [4] it is shown that the objective function is a generalization of the makespan. Nonetheless only min-max criteria can be considered and no generalization for different performance measures is possible. Still the introduction of *Released* and *Planned* lots leads to a *Just-in-time* philosophy, *i.e.* to an objective function that can keep track of both early and tardy penalties.

• The Genetic Algorithm can operate with other objective functions such as total weighted completion time, total weighted tardiness and weighted number of late jobs and with some modifications (which should be expected however to worsen and slow down their performance) Tabu Search and Simulated Annealing can do the same. Nevertheless, none of those methods can easily handle problems with a non-regular performance measure as

objective function, e.g. sum of earlinesses and tardinesses. With such performance measures the optimum may not belong to the set of active schedules and therefore it is not straightforward to desume the schedule from a given sequence (even on a single machine the optimal insertion of idle time given the sequence requires an O(nlogn) algorithm – *see* [10]). The above observation also applies to the *bottleneck dynamics* approach in which the priorities of the operations and/or of the machines require not only the sequence of the operations already set, but also their schedule as preliminary information.

• Even for regular performance measures, the computation time remains acceptable only for medium-size problems. For larger problems (e.g. those involving more than 10,000 operations) it shoots up: in these cases a rougher but faster way of estimating the schedule for a given sequence has to be found.

3.3. Finding a feasible starting solution

Both Simulated Annealing and Tabu Search require a feasible starting solution for their execution, whereas Genetic Algorithms need a set of (possibly) different feasible solutions. Van Laarhoven, Aarts and Lenstra [24] and Della Croce, Tadei and Volta [5] opted to start from randomly generated schedules, whether Dell'Amico, Trubian [6] and Nowicki, Smutnicki preferred greedy procedures. For Simulated Annealing and Tabu Search methods it seems reasonable to choose the latter approach. This is true mostly for large size instances where the neighborhood is wide and thus convergence (even to a local minimum) may be too slow if a random start is chosen, whereas the time required for a greedy procedure remains reasonably short. For Genetic Algorithms the situation is different: although in this case too it is worthwhile tightening the neighborhood, a full family of schedules has to be generated and therefore the computational overhead cannot be neglected. If problems involving deadlines are considered, the search for a feasible starting solution obviously becomes much more difficult as there is no guarantee that this solution exists. In order to partially overcome this difficulty, we can perform a feasibility phase in which due dates are substituted by deadlines and the objective function becomes the maximum lateness (if for this new problem the optimal value of the objective function is not positive, it means that no job is processed after its deadline and so the schedule is feasible). The feasibility phase on its own will be an NP-hard problem, but it will be possible to get a heuristic solution to the problem by

using one of the above methods. If there are few feasible schedules, however, the power of the local search techniques is greatly reduced because most of the neighborhood of any solution is composed by unfeasible sequences.

3.4. Computing the schedule for a given sequence

If the sequence for all machines is given, the problem of computing the schedule can be easily formulated as a linear programming model (*see* Fry, Armstrong and Blackstone [8] for the single machine case: the extension for the job shop problem is straightforward). Nonetheless, if it is necessary to solve a linear program many times, the required CPU time may be too long. In the multi-machine environment a further fact must be taken into consideration. In a job shop not all sequences are feasible and thus a move from a feasible configuration may lead to an unfeasible one. GA and CC solve this issue by applying a constructive algorithm which derives a feasible schedule by operating on the original unfeasible sequence. This is not the case for SA, TS-1 and TS-2 in which the neighborhood presents just feasible sequences (but only if the objective function is the makespan). Given a feasible sequence, the problem of scheduling is trivial only if the objective function is a regular performance measure, as an optimum may be found in the set of active schedules. With non regular performance measures (such sum of earlinesses and tardinesses), only for single machine problems algorithms faster than the simplex method exist (*see* [10]). Two alternative ways are currently available: either the use of a rough approach to prejudge the quality of the sequence in order to avoid solving the LP model at every step, or the choice of running a simplex method each time, with the drawback of the explosion of the computing time. There is a dramatic need for a fast polynomial method for this relatively simple subproblem.

3.5. The single machine as a subproblem

Both SBP and CC require the iterated solution of single machine problems as a result of the decomposition of the main problem. As we mentioned earlier the strength of SBP is the fact that Carlier's algorithm [2] gives an exact and rapid solution to the Single Machine Maximum Lateness Problem subject to different release times. It is still to be proven that the generalization of this method in presence of more complicated one-machine problems requiring an approximate solution could lead to comparable results. This observation applies equally to CC whose less brilliant results may well be linked to the heuristic solution of its single machine total square lateness

subproblem. The main difficulty is that the decomposition generally implies the solution of NP-hard single machine problems, requiring an approximate solution (except special cases like SBP procedure). Given that the robustness of the SBP approach for other objective funtions has not been studied yet, an application of the bottleneck philosophy using for instance a local search for the solution of the single machine subproblems could be an interesting direction for future research.

4. CONCLUSIONS

In this paper a methodological and computational comparison between different multimachine scheduling heuristics has been proposed. Still much work is to be done in this field and many different topics are to be explored. In particular we point out the followings:

• As far as Genetic Algorithms are concerned, in order to improve the quality of the solution, it should be worthy to start with an initial population not randomly selected, but obtained through some fast greedy problem-oriented methods as, for instance,

– lead time iteration technique (*see* [18]) with multiple and distinct methods to obtain the starting release times and tails of the operations;

– a randomized version of the SBI technique (*see* 2.1.) with random choice of the so-called "bottleneck-machine";

– the "bidirectional method" proposed as the starting solution in [6] modified by setting the cardinality of parameter "c" at a high value.

• A rough but fast method for prejudging the quality of a given sequence without necessarily computing the whole schedule would save much computing time allowing therefore to deal with large-size problems.

• A crucial issue is to find a more efficient algorithm for computing the optimal schedule for a given sequence in the multi-machine early-tardy environment.

REFERENCES

1. J. Adams, E. Balas and D. Zawack, The shifting bottleneck procedure for job shop scheduling, *Management Science*, 1988, *34*, pp. 391-401.
2. J. Carlier, The one-machine sequencing problem, *European Journal of Operational Research*, 1982, *11*, pp. 42-47.
3. C. Chu, A Branch-And-Bound algorithm to minimize Total Tardiness with Different Release Dates, *Naval Research Logistics*, 1992, *39*, pp. 265-283.

4. F. Della Croce, G. Menga, R. Tadei, M. Cavalotto and L. Petri, Cellular control of manufacturing systems, *European Journal of Operational Research*, 1993, *69*, pp. 498-509.
5. F. Della Croce, R. Tadei and G. Volta, A genetic algorithm for the job shop problem, to appear on *Computers and Operations Research*, 1994, *8*.
6. M. Dell'Amico and M. Trubian, Applying tabu search to the job shop scheduling problem, *Annals of Operations Research*, 1993, *41*, pp. 231-252.
7. E. Falkenauer and S. Bouffoix, A Genetic Algorithm for Job Shop, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 824-829.
8. T. D. Fry, R. D. Armstrong and J. H. Blackstone, Minimizing Weighted Absolute Deviation in Single Machine Scheduling, *IIE Transactions*, 1987, 4, *vol. 19*, pp. 445-450.
9. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
10. M. R. Garey, R. E. Tarjan and G. T. Wilfgong, One-processor scheduling with symmetric earliness and tardiness penalties, *Mathematics of Operations Research*, 1988, *13*, pp. 330-348.
11. F. Glover, Tabu Search, Part I, *Orsa Journal on Computing*, 1989, 1, *3*, pp. 190-206.
12. F. Glover, Tabu Search, Part II, *Orsa Journal on Computing*, 1990, 2, *1*, pp. 4-32.
13. D. J. Hoitomt, P. B. Luh and K. R. Pattipati, A practical approach to job shop scheduling problems, *IEEE Transactions on Robotics and Automation*, 1993, 1, *vol. 9*, pp. 1-13.
14. S. Lawrence, *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques*, GSIA Carnegie Mellon University, 1984.
15. S. R. Lawrence and T. E. Morton, Resource-constrained Multi-project Scheduling with Tardy Costs: Comparing Myopic, Bottleneck and Resource Pricing heuristics, *European Journal of Operational Research*, 1993, 2, *vol. 64*, pp. 168-187.
16. M. Minoux, *Mathematical Programming: Theory and Algorithms*, Wiley & Sons, 1986.
17. T. E. Morton, S. Kekre, S. R. Lawrence and S. Rajagopalan, SCHED-STAR: A Price Based Shop Scheduling Module, *Journal of Manufacturing and Operations Management*, 1988, *1*, pp. 131-181.
18. T. E. Morton and D. W. Pentico, *Heuristic Scheduling Systems*, Wiley & Sons, 1993.
19. R. Nakano and T. Yamada, Conventional Genetic Algorithm for Job Shop Problems, *Proceedings of the 4th International Conference on Genetic Algorithms*, San Diego, California, July 13-16, 1991, pp. 474-479.
20. E. Nowicki and C. Smutnicki, *A fast Taboo Search algorithm for the Job Shop Problem*, Internal Report, Polytechnic Institute of Wroclaw, Poland, 1993.
21. S. S. Panwalker and W. Iskander, A Survey of scheduling rules, *Operations Research*, 1977, *25*, pp. 45-61.
22. D. A. Pierre and M. J. Lowe, *Mathematical Programming via Augmented Lagrangians*, Addison-Wesley, 1975.
23. B. Roy and B. Sussmann, *Les Problèmes d'Ordonnancement avec Contraintes disjonctives, Note DS n. 9 bis*, 1964, SEMA, Montrouge.
24. P. J. M. Van Laarhoven, E. H. L. Aarts and J. K. Lenstra, Job Shop Scheduling by Simulated Annealing, *Operations Research*, 1992, *40*, pp. 113-125.
25. F. Werner and A. Winkler, *Insertion Techniques for the heuristic solution of the Job Shop Problem*, Internal Report, Technical University Otto von Guericke, Magdeburg, Germany, 1992.