# A review and classification of heuristics for permutation flow-shop scheduling with makespan objective

**Jose M. Framinan[1], Jatinder N. D. Gupta[2] and Rainer Leisten[3]**

**Technical report OI/PPC-2001/02**

**Version 1.2 - 20/07/2002**

***A revised, enhanced version of this Report is to be published in the Journal of the Operational Research Society***

## ABSTRACT

Makespan minimisation in permutation flow shop scheduling is an OR topic that has been intensively addressed in the last 40 years. Since the problem is known to be NP-complete for more than two machines, most of the research effort has been devoted to the development of heuristic procedures in order to provide good approximate solutions to the problem. However, little attention has been devoted to establish a common framework for these heuristics so they can be effectively combined or extended. In this paper, we review and classify the main contributions regarding this topic and discuss future research issues.

[1] Industrial Management, School of Engineering, University of Seville, Spain. Email: jose@esi.us.es

[2] Department of Management, Ball State University, USA.

[3] Production Management, Institute for Logistics and Information Management, Faculty of Business Administration and Economics, University of Duisburg, Germany.

## 1. Introduction

During the last 40 years, the permutation flowshop sequencing problem with the objective of makespan minimisation has held the attraction of many researchers. This problem – characterised as $F_m|prmu|C_{max}$ in the notation of Graham (1979) – involves the determination of the order of processing of $n$ jobs in $m$ machines. A detailed discussion of the basic assumptions followed throughout the relevant literature can be found e.g. in Dudek and Teuton (1964). A number of exact approaches have been suggested for the problem (see e.g. Szwarc 1971, Lageweg et al. 1978, Potts 1980, or Carlier and Rebaï 1996), although since the problem is known to be NP-complete for three or more machines (Garey et al. 1976, and Rinnooy Kan 1976), most of the effort has been concentrated in proposing heuristic procedures that produce good (but not necessarily optimal) solutions in relatively short time intervals, such as those required to take the scheduling decisions. Currently, there are many heuristics available based in very different approaches to the problem. However, no framework to fit these heuristics has been developed, although several attempts to classify them have been done (see e.g. Gupta 1971a, Pinedo 1995, or Lourenço 1996).

Hence, the aim of this paper is beyond introducing and explaining all current available heuristics for the makespan minimisation problem. Instead, we try to establish a general framework in which the existing heuristics can be fitted and – which, in our opinion, is more important – combined in order to obtain composite heuristics that lead to improve the quality of the obtained solutions. Besides, this framework may serve to indicate future points of research.

The remainder of the paper is organised as follows: In the next section, we introduce the previous work related to the review, classification or categorisation of the existing heuristics, and introduce the main phases of the suggested framework. These phases are discussed in detail in sections 3, 4, and 5, respectively. Finally, in section 6 the findings in the previous sections are summarized and lines for future development of heuristics are drawn.

## 2. Previous work and proposed framework

As mentioned in the previous section, the number of heuristics for the $F_m|prmu|C_{max}$-problem grew spectacularly since the early 60's of the last century. This process speeded up by the confirmation that the problem under consideration was NP-complete, and the application of general-purpose local search procedures to its solution. Some reviews on the development of heuristics can be found in Gupta (1979), King and Spachis (1980), and Park *et al*. (1984).

As the number of available heuristics for the $F_m|prmu|C_{max}$-problem was increasing, it became clear that not all were of the same nature and hence present very different properties, such as the complexity order, computation time, or memory requirements. The above perception was even clearer when some general-purpose local search procedures (also known as meta-heuristics) were successfully applied to the $F_m|prmu|C_{max}$-problem and the results compared to those offered by the 'old' heuristics. For instance, Widmer and Hertz (1989), or Moccellin (1995) present heuristics that require an initial solution (obtained in both cases by an analogy with the Travelling Saleman Problem) followed by a tabu search approach that, in principle, might employ as starting solution any other heuristic included in the comparison, such as the one by Nawaz et al. (1983).

Following the sense that not all the available heuristics for the $F_m|prmu|C_{max}$-problem were of the same nature, the term *constructive* heuristic was coined in many papers. However, besides being a rather coarse classification and hence of limited scope, the meaning of *constructive* remains somewhat confusing (and consequently so remain the heuristics covered under this definition). For instance, on one hand Pinedo (1995) defines constructive heuristics as opposed to *composite heuristics*, being the latter those heuristics resulting from the combination of simple (constructive) heuristics. On the other hand, Lourenço (1996) defines a constructive heuristic as "an algorithm that builds a sequence of jobs and once a decision is made, it is never changed".

Besides, there were earlier attempts to classify the existing heuristics for flowshop scheduling, such as Gupta (1971a). In this work, heuristics are classified into fixed functional heuristics, floating functional heuristics, and synthetic functional heuristics.

The first are heuristics that exploit the functional characteristics of a sorting problem. Basically, this consists in developing some index to sort the job, being the index based in the processing times of the problem instance. Gupta includes in that classification heuristics such as the ones by Palmer (1965), Campbell et al. (1970), and Gupta (1971b). Floating function heuristics generate a running function for a partial sequence. The author pointed as examples the MINIT and MINOT algorithms in Gupta (1968a, 1972). Finally, a synthetic functional heuristic is, e.g., the MINMAX heuristic in Gupta (1972), based on scheduling jobs with minimum processing time first and those with maximum processing time last.

In our opinion, all above classifications present some kind of problem. In Pinedo's classification it would be difficult to classify some heuristics that might be regarded both as simple heuristics as well as a form of generalisation of a simpler heuristic. For instance, the CDS heuristic (Campbell et al. 1970) might be considered a generalised form of the machine aggregation heuristic (Röck and Schmidt 1983). According to Lourenço (1996), many heuristics of a very different nature, such as the Nawaz et al. (1983) heuristic and the Gupta (1971a) heuristics would be grouped into the term constructive heuristic, while in practice the latter might be considered a starting point of the former. With respect to Gupta's classification, on one hand it does not cover early developments in heuristics such as the one by Page (1961), which is based on some form of local search. On the other hand, relatively new developments such as the meta-heuristics do not fit into his classification for obvious reasons. Finally, other work, such as the one by Morton and Pentico (1993) simply distinguishes between heuristics and 'local search methods' without linking them.

In this sequel, we try to develop a general framework to extend previous classifications and fit all heuristics. The framework that we propose here will cover also some heuristics originally developed for permutation flowshops with other objective functions different from makespan (i.e. flowtime), but whose design allows the immediate transfer to the makespan minimisation problem. The term 'immediate transfer' refers here to the mere replacement of the original objective function by makespan.

Before we present the general framework, we would like to make a remark on some heuristics we will not include in the framework. These are heuristics based in

decomposition/composition principles. The basic idea of these heuristics is to split the set of jobs to be scheduled into two or more separate groups (decomposition). Once the scheduling problem has been solved within these groups – either optimally or by some approximate procedure –, then these solutions are merged (composition) and the composed solution is retained as solution of the original problem. Some research dealing with these approaches can be found in Ashour (1967, 1970), and Gupta and Maykut (1973a).

The reason for not including these heuristics into the framework is because they do not address the scheduling problem 'per se', but rather are decomposition/composition approaches that rely on 'true' scheduling procedures – e.g. some sequencing heuristics – for solving the scheduling problem within each set of jobs. Nevertheless, in the final section of the paper, some research issues addressing these heuristics will be discussed.

Besides, we will not include the approximate development in Averbakh and Berman (1999), consisting in accepting the best makespan between a randomly chosen sequence and its reverse sequence. Although its worst-case may not be very bad and the foundations of this procedure is statistically supported by the experiments by Azim *et al*. (1989) in the sense that they show to be likely that reversing a sequence with high makespan will produce a sequence with low makespan, we do not believe the Averbakh and Berman procedure to be competitive against the heuristics reviewed here. In contrast, the idea exploited by Azim *et al*. is interesting and could be further exploited by current heuristics employed for makespan minimisation.

In the general framework that we propose here, the development of a heuristic may consist of three phases:

- Phase I: Index development
- Phase II: Solution construction
- Phase III: Solution improvement

A single heuristic may consist of one or more of these phases, that are, in general, independent one from each other. However, a heuristic consisting of more than one of

these phases must perform the phases according to the above order. The three phases are explained in detail in the following sections.

## 3. **Phase I: Index development**

In index development, jobs are arranged according to a certain property based on the data of the $F_m|prmu|C_{max}$-problem instance, i.e. based on the processing times of each job on each machine. In general, this arrangement does not need to make any assumption on the ordering of the jobs in the sequence. The output of this phase is a ranking of the jobs that might be employed either as input for the next phase, or as a solution itself.

To derive the index, one might use a problem analogy or not. 'Problem analogy' refers to employ the data of the $F_m|prmu|C_{max}$-problem to construct and solve an instance of a different class of problems. Once the latter has been solved by some means, the obtained solution is converted into a ranking of the jobs for the original $F_m|prmu|C_{max}$-problem. Perhaps the most obvious example of problem analogy is the $F_2|\ |C_{max}$-analogy, which is employed in heuristics such as the one by Campbell et al. (1970). In this heuristic, an $F_2|\ |C_{max}$-problem instance is constructed by machine aggregation from the data given by the original $F_m|prmu|C_{max}$-problem instance to be solved. The so obtained $F_2|\ |C_{max}$-problem is then optimally solved in polynomial time employing Johnson's rule, and its optimal solution is accepted as (heuristic) solution for the original $F_m|prmu|C_{max}$-problem instance.

If no analogy is used, the property has to be merely deducted from some form of relationship assumed involving the processing times of the jobs and their corresponding ranking within a schedule. Note that this absence of a problem analogy – that is, ranking jobs according to a given function – can be interpreted as well as a 'sorting analogy', term coined by Gupta (1971a).

Historically, the first of heuristic approaches ranking the jobs according to their processing times is due to Palmer (1965). He specified a job priority function named 'slope index'. The form of the priority function was chosen to give priority to jobs progressing from short to long processing times. An improved slope index priority

function has been developed by Gupta (1971b) and Gupta (1976). A variation of this scheme has been produced by Bonney and Gundry (1976), which involves two slope indices for each job.

Finally, Koulamas (1998) defines an index based on the conditions given by Burns and Rooker (1976) to make an $F_3|\ |C_{max}$-problem instance to be optimally solved by Johnson's rule (Johnson 1954), that is, when there is coincidence among the optimal solutions for the three $F_2|\ |C_{max}$-subproblems given by the first and second machine, second and third machine, and first and third machine respectively. Koulamas' heuristic also includes a second phase to allow non-permutation schedules. Therefore, as we are discussing here only $F_m|prmu|C_{max}$-problems, we only refer to the first phase of his heuristic. Also, we want to note that Koulamas' heuristic could be considered to employ a problem analogy with the $F_2|\ |C_{max}$-problem, since it uses some assumptions valid for $F_2|\ |C_{max}$. Hence, without loss of generality of the proposed framework, it could be included among these heuristics using the $F_2|\ |C_{max}$ analogy.

In case that some problem analogy is to be employed, the following two issues have to be considered:

1) The simplicity of the problem analogy employed, i.e. whether there are polynomial methods available to optimally solve the problem analogy, or at least some efficient heuristics for the latter.
2) The correspondence between solutions of the analogy and solutions of the $F_m|prmu|C_{max}$-problem, i.e. whether it can be proved that good or optimal solutions to the problem analogy yield good (or optimal) solutions to the original problem.

The first issue implicitly points to the complexity of the analogy problem. Although this cannot be formally proved, it seems anti-intuitive to obtain heuristic solutions (based in some form of analogy) whose average behaviour outperform these heuristics designed for the original problem when the complexity of the employed problem analogy is equal or higher than the one of the original problem. With respect to the second question, the proof of the correspondence could be established in a formal way or through statistical evidence. Obviously, the former is preferred if possible, since this would provide

validity for any single problem instance, which cannot be guaranteed by statistical evidence.

Three problem analogies have been used so far for deriving an ordering of the jobs. These are the $F_2|\ |C_{max}$-analogy, the Travelling Salesman Problem (TSP) analogy, and the vector summation problem analogy. These are discussed in detail in the following sections.

### *3.1. $F_2|\ |C_{max}$-analogy*

As mentioned before, the clearest and most widely used type of analogy is the analogy with a $F_2|\ |C_{max}$-problem instance via machine aggregation. Machine aggregation refers to a simplification of the original problem with respect to machines. As a result, the original problem instance is converted into a new problem – named *aggregated problem instance* in the remainder – with a number of (fictitious) machines – named *m'* in the sequel – lower than in the original one. More specifically, an *n* job, *m* machine-problem instance is transformed into an *n* job, *m'* machine-problem instance, being $m' < m$.

With respect to the two above issues regarding analogy mentioned in the previous section (simplicity and correspondence), the first is achieved by setting *m'* to 2, since $F_2|\ |C_{max}$ is optimally solvable in polynomial time, according to Johnson's rule (Johnson 1954). Therefore, although *m'* may hold – in theory – any value $m' < m$, *m'* is set to 2 in all existing heuristics using the machine aggregation procedure. On the other hand, machine aggregation involving $m' > 2$ implies solving a problem whose complexity is identical to the original one. Therefore, it seems to be unlikely to develop efficient machine aggregation heuristics with $m' > 2$ (see above comments on the complexity of the analogy), unless specific heuristic approaches are developed for some $F_{m'}|\ |C_{max}$ being $2 < m' < m$, or there exist heuristics performing extraordinarily better for a certain $m' < m$. Finally, note that for $m' < 2$, i.e. $m' = 1$, any schedule is optimal, and hence it does not seem to be possible to convert the solution of the corresponding aggregated problem into a meaningful solution of the original problem.

With respect to correspondence between solutions, to the best of our knowledge, no formal proof of this correspondence has been developed so far. In fact, providing a

formal correspondence between the solutions of both problems seems to be difficult, since the solution of the analogy is given (Johnson's rule) by the processing times of the (fictitious) machines, which in turn depend on the processing times of the jobs in the machines of the original problem. Therefore, since the processing times are involved, the correspondence seem to be instance-dependent, and hence 'good' approaches for the aggregation can be distinguished from 'bad' approaches only on a statistical basis, i.e. the percentage of problem instances where a certain approach to assign processing times produces aggregate instances whose optimal solution represents a 'good' solution for the original problem instance. The main approaches to assign the processing times to two machines are due to Campbell et al. (1970), Dannenbring (1977), Röck and Schmidt (1983), Selim and Al-Turki (1987), and Lai (1996). These approaches are summarised in Table 1.

| Reference | $p_{i1'}$ | $p_{i2'}$ |
|---|---|---|
| Campbell et al. (1970) | $\sum_{j=1}^{k} p_{ij} \quad (k = 1,\ldots,m-1)$ | $\sum_{j=m-k+1}^{m} p_{ij} \quad (k = 1,\ldots,m-1)$ |
| Dannenbring (1977) | $\sum_{j=1}^{m} (m-j+1)\, p_{ij}$ | $\sum_{j=1}^{m} j\, p_{ij}$ |
| Lai (1996) | $\sum_{j=1}^{\lfloor m/2 \rfloor} p_{ij}$ | $\sum_{j=m-\lfloor m/2 \rfloor+1}^{m} p_{ij}$ |
| Röck and Schmidt (1983) | $\sum_{j=1}^{\lceil m/2 \rceil} p_{ij}$ | $\sum_{j=\lceil m/2 \rceil+1}^{m} p_{ij}$ |
| Selim and Al-Turki (1987) | $\sum_{j=1}^{m} \lambda_j\, p_{ij}$ | $\sum_{j=1}^{m} \mu_j\, p_{ij}$ |

*Table 1. Machine aggregations suggested by the heuristics using $F_2|\ |C_{max}$-analogy*

Perhaps the most complex approach among the ones shown in table 1 is the one employed by Selim and Al-Turki (1987), which tries to optimise the values of $\lambda_j$ and $\mu_j$ ($j$ = 1,…., $m$) that minimises the makespan of the original problem. This approach is reported to obtain very good results for small problems, although its computational requirements seems not to make it effective for larger number of problems.

It has to be noted that all approaches listed in table 1 optimally solve the $F_2|\ |C_{max}$-problem (by Johnson's rule) and then use that solution as a solution of the $F_m|prmu|C_{max}$-problem, with the exception of the heuristic of Lai (1996), which simply classifies a job $i$ into one of two groups $U := \{i : p_{i1'} \leq p_{i2'}\}$, or $V := \{i : p_{i1'} > p_{i2'}\}$ in the manner of Johnson's rule. However, the jobs in each of the groups are not sorted but sequenced arbitrarily, and merged with the only condition that the jobs in $U$ must precede the jobs in $V$. This procedure allows the heuristic to be very fast while presenting a worst-case analysis not very different from those of the other machine aggregation procedures. However, no computational experiments have been carried out to test the performance of this heuristic.

*3.2. TSP analogy*

The analogy of the $F_m|prmu|C_{max}$-problem with the TSP was first pointed out by Gupta (1968b) and has been studied by Stinson and Smith (1982), Widmer and Hertz (1989), and Moccellin (1995). In these three papers, a distance-matrix defining a TSP-instance of $n$ cities is set up given by the processing times of the machines (in the case of the Stinson and Smith-paper, up to six different ways to build the distance matrix are presented). Then, the TSP instance is solved, and the solution retained as solution of the original $F_m|prmu|C_{max}$-problem. The ways employed in the above papers to construct the distance matrix are summarised in table 2.

| Reference | Distance matrix ( $1 \le u, v \le n$ ) |
|---|---|
| Gupta (1968b) | $C_{uv} = ( CT_m(u,v) - \sum_{j=1}^{m} p_{uj} )$ where $CT_0(u,v) = 0$, and<br>$CT_j(u,v) = max \{ CT_{j-1}(u,v) ; \sum_{k=1}^{j} p_{vk} \}$ |
| Stinson and Smith (1982) | $C_{uv} = \sum_{j=2}^{m} \lvert p_{uj} - p_{v,j-1} \rvert$ |
| | $C_{uv} = \sum_{j=2}^{m} \{ p_{uj} - p_{v,j-1} \}^2$ |
| | $C_{uv} = \sum_{j=2}^{m} \lvert min\{p_{uj} - p_{v,j-1} + min\{(p_{u,j-1} - p_{v,j-2});0\};0\} \rvert$ |
| | $C_{uv} = \sum_{j=2}^{m} \lvert p_{uj} - p_{v,j-1} + min\{(p_{u,j-1} - p_{v,j-2});0\} \rvert$ |
| | $C_{uv} = \sum_{j=2}^{m} \{ p_{uj} - p_{v,j-1} + min\{(p_{u,j-1} - p_{v,j-2});0\} \}^2$ |
| | $C_{uv} = \sum_{j=2}^{m} max\{(p_{uj} - p_{v,j-1});0\} + 2 \lvert min\{(p_{u,j} - p_{vj-1});0\} \rvert$ |
| Widmer and Hertz (1989) | $C_{uv} = p_{u1} + \sum_{j=2}^{m} (m-k) \lvert p_{uj} - p_{v,j-1} \rvert + p_{vm}$ |
| Moccellin (1995) | $C_{uv} = UBX(m)_{uv}$ where $UBX(1)_{uv} = 0$, and<br>$UBX(k+1)_{uv} = max \{ 0; UBX(k)_{uv} + (p_{vk} - p_{u,k+1}) \}$ |

*Table 2. Distance-matrices constructed by the heuristics using TSP analogy*

Since the TSP is known to be NP-hard, there is no method available guaranteeing an optimal solution within the decision interval. To obtain good approximate solutions, Stinson and Smith suggest a constructive polynomial heuristic developed by Stinson (1977), while Widmer and Hertz, and Moccellin use the Farthest Insertion Travelling Salesman Procedure (FITSP) to obtain the solution to the TSP problem. Furthermore, both Widmer and Hertz as well as Moccellin, improve this solution by a tabu search procedure (which will be discussed in phase III).

### *3.3. Vector summation analogy*

The vector summation analogy was presented by Sevast'janov (1995) from a theoretical point of view, and several versions of it were implemented by Lourenço (1996). With respect to the two issues discussed above when employing a problem analogy (simplicity and correspondence), the analogy employed can be solved using a polynomial-time algorithm based on linear programming (Lawler *et al*. 1993). Regarding correspondence issues, the experimental results carried out by Lourenço conclude that there is not necessarily a correspondence between good solutions with respect to the maximum norm of the partial sums, and good solutions of the $F_m|prmu|C_{max}$-problem.

## 4. **Phase II: Solution construction**

In this phase, a solution is constructed in a recursive manner trying one or more unscheduled jobs to be inserted in one or more positions of a partial schedule until the schedule is completed. Hence, this phase consists of $n$ loops that divide the set of jobs into two subsets: the already scheduled jobs in subset $S$ and the non-scheduled in $R$. Note that in the $k$th loop $S_k$ defines a partial schedule of $k$ jobs. In a loop, one job from the non-scheduled set is removed from the non-scheduled and placed into the scheduled jobs set.

In this phase there are two issues at any loop $k$:

1. Job selection, i.e. which job $j \in R_k$ is to be inserted into $S_k$.

2. Job insertion, i.e. where to insert the chosen job into the partial schedule $S_k$.

These two issues are discussed in detail in the following sections.

*4.1. Job selection*

Job selection must be decided according to one of the following approaches:

- According to any index obtained by the application of phase I (as done e.g. by the heuristic by Nawaz et al. 1983), or
- Several jobs in $R_k$ might be tried. In this case, the selection about the job to be finally inserted is done according to a certain schedule measure, i.e. to minimise some property of the resulting partial schedule (e.g. in the heuristic by Sarin and Lefoka 1993, several jobs are tried, and the schedule property to be minimised is the idle time on the last machine).

The property of the partial schedule employed for the job selection set might be the objective function of the problem itself (makespan), as done e.g. in Aggarwal and Stafford (1980), or Woo and Yim (1998), or some other measure related to the objective function. Among the latter, some expressions involving machine idle time have been chosen in several heuristics. The rationale for using this measure is clear, since the makespan can be de-composed into the time to process all jobs on all machines plus the idle time of each machine for each job. Taking into account that the first term of the expression is a constant, it is clear that minimising some form of idle time may lead to schedules with low makespan.

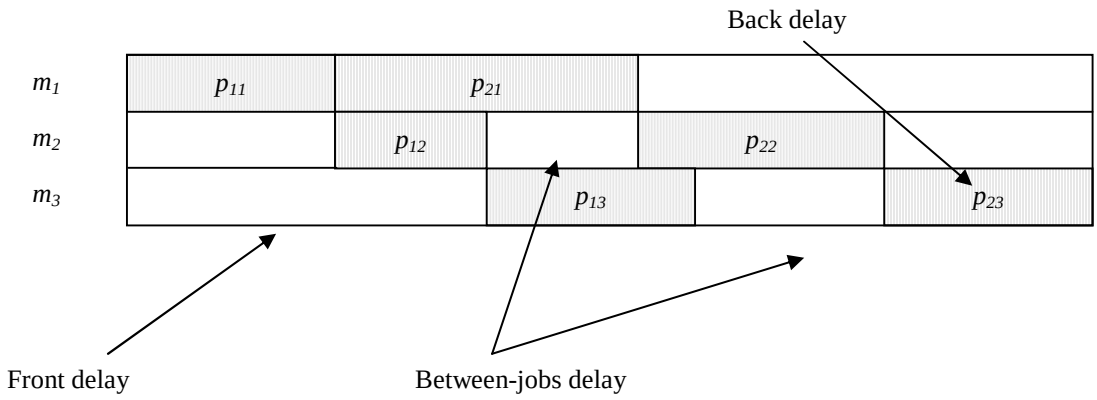

*Figure 1. Different measures related to idle-time.*

To clarify the different expressions involving idle-time, we depict figure 1. Here, we represent different measures connected to the idle-time. We use the terminology of King and Spachis (1980), since they treated intensively the different ways to measure it.

As can be seen from figure 1, King and Spachis (1980) distinguish three types of delays causing machine idle-time. The total idle-time (as employed e.g. by Gupta 1972, and McCormick et al. 1989) is the sum of front delay, between-jobs delay, and back delay. Sarin and Lefoka (1982) employ the idle-time on the last machine, that is: the front and between-jobs delays corresponding to machine $m_3$ in figure 1. There are alternative definitions of idle-time, such as the one in Sridhar and Rajendran (1996), including front and between-jobs delays, but excluding back delays. However, these are not employed in the reviewed heuristics.

Finally, in case that more than one job is tried, it is possible that ties occur, so some tie-breaking rule has to be devised. This aspect has not been explicitly covered by most of the heuristics. To our knowledge, only Gupta's MINOT and MINIT heuristics (1972) present some method for breaking ties other than random.

### *4.2. Job insertion*

With respect to job insertion, a job $j$ can be inserted into a fixed position in the partial schedule $S_k$ (e.g. Sarin and Lefoka insert the chosen job at the end of $S_k$), or several positions can be tried (e.g. Aggarwal and Stafford define a set of positions where the new job can be inserted, Rajendran 1993 limits the possible positions from $\left\lfloor k/2 \right\rfloor$ to ($k$ + 1), while Nawaz et al. 1983, and Woo and Yim 1998 try to insert the job into all possible positions). King and Spachis (1980) distinguish between 'single chain' and 'multiple chain' schedules, depending on whether one or several positions are tried, respectively.

Again, when several positions are tried, the selection of the position should be done to minimise some property of the resulting partial schedule. Obviously, when several jobs

from $R_k$ have to be inserted in several positions of the partial schedule $S_k$, both schedule measures must be the same or at least consistent to one another (e.g. Aggarwal and Stafford, or Woo and Yim use partial makespan minimisation for both job selection and job insertion). Also obviously, to assure a good performance of this phase, both partial schedule measures must coincide – or at least be consistent – with the objective function (makespan minimisation).

Similar to the case of job selection, there must be some rule for breaking ties when multiple positions are tried. Again, this aspect is ignored by most of the existing heuristics (see e.g. Nawaz et al. 1983, Rajendran 1993, or, Woo and Yim 1998).

It has to be noted that the job selection and job insertion decisions (if to be taken) are taken using partial (local) information and therefore only myopic improvements might be obtained. Hence, it cannot be guaranteed that the resulting schedule provides a lower makespan than the initial schedule developed in phase I or than a random schedule. This is true even in the extreme case that all jobs in $R_k$ are tried for selection, all positions in $S_k$ are tried, and the measure for both job selection and job insertion is (partial) makespan minimisation.

Table 3 summarises the choices regarding job selection and job insertion done in the literature.

| Reference | Job selection (loop $k$) | Job insertion (loop $k$) |
|---|---|---|
| Aggarwal and Stafford (1980) | subset in $R_k$ obtained from phase I (makespan) | subset in $S_k$ (makespan) |
| Gupta – MINIT (1972) | 1 job obtained from phase I for $k = 1$<br>all jobs in $R_k$ for $1 < k \leq (n-2)$ (idle time)<br>all jobs in $R_k$ for $(n-2) < k \leq n$ (makespan) | at position $(k+1)$ |

*Table 3. Heuristics using phase II. When some options are tried both in job selection and in job insertion, the selection criterion is given in parentheses.*

** indicates that the original heuristic was conceived to minimise an objective function different from makespan, although it can be directly applied to makespan by replacing the original objective function by makespan.*

| Reference | Job selection (loop $k$) | Job insertion (loop $k$) |
|---|---|---|
| Gupta – MICOT (1972) | 1 job obtained from phase I for $k = 1$<br>all jobs in $R_k$ for $1 < k \leq (n-2)$<br>(completion time)<br>all jobs in $R_k$ for $(n-2) < k \leq n$<br>(makespan) | at position $(k+1)$ |
| Gupta – MINIMAX (1972) | all jobs in $R_k$<br>( if $k$ odd, $\min_{\substack{i \in R_k \\ 1 \leq j \leq m}} \{p_{ij}\}$ , else $\max_{\substack{i \in R_k \\ 1 \leq j \leq m}} \{p_{ij}\}$ ) | if $k$ odd at position $(k+1)$, else at position 0 |
| Gupta and Maykut (1973b) | 1 job obtained from phase I for $k = 1$<br>all jobs in $R_k$ for $1 < k \leq (n-2)$<br>(idle time in last machine) | at position $(k+1)$ |
| Gupta (1979) | all jobs in $R_k$<br>( lower bound of total makespan ) | at position $(k+1)$ |
| King and Spachis (1980) | all jobs in $R_k$<br>(LFD – front delay)<br>(LBD – back delay)<br>(LBJD – between-jobs delay)<br>(LWBJD – weighted between-jobs delay)<br>(MLSS – savings of between-jobs delay) | at position $(k+1)$ |
| McCormick et al. (1989) | 1 job obtained from phase I for $k = 1$<br>all jobs in $R_k$ for $1 < k \leq n$<br>(idle time) | at position $(k+1)$ |
| Nawaz et al. 1983 | 1 job obtained from phase I | all $(k+1)$ possible positions are tried<br>(makespan) |
| Rajendran* (1993) | 1 job obtained from phase I | positions $\lceil k/2 \rceil$ to $(k+1)$ are tried<br>(makespan) |
| Sarin and Lefoka (1993) | all jobs in $R_k$<br>(idle time in last machine) | at position $(k+1)$ |
| Woo and Yim* (1998) | all jobs in $R_k$<br>(makespan) | all $(k+1)$ possible positions are tried<br>(makespan) |

*Table 3 (continued). Heuristics using phase II. When some options are tried both in job selection and in job insertion, the selection criterion is given in parentheses.*

** indicates that the original heuristic was conceived to minimise an objective function different from makespan, although it can be directly applied to makespan by replacing the original objective function by makespan.*

## 5. Phase III: Solution improvement

In this phase, an existing solution is improved by means of some procedure. The two main characteristics of this phase are:

- An initial solution (input solution) is required.
- The solution output of the phase is always equal or better than the input solution.

Starting from an input solution, the procedure seeks to change the job sequence to improve the solution. This can be done on an individual basis – e.g. two jobs are exchanged as done in Dannenbring (1977) – or on a group basis – e.g. a set of adjacent jobs is exchanged as done in Page (1961). In the second case, some procedure has to be developed in order to group (and perhaps regroup) the set of jobs.

Usually, improvement approaches are classified into descending local searches and metaheuristics (Nowicki and Smutnicki 1996, Stützle 1998). Heuristics using descending local search are those by Page (1961), the final phase of the heuristic RAES by Dannenbring (1977), the final phase of the heuristic by Aggarwal and Stafford (1980), and the heuristic by Ho and Chang (1991). In addition, the five heuristics proposed by King and Spachis (1980) and discussed in the previous section include a final improvement phase by exchanging the last two jobs inserted in the search of makespan improvements.

With respect to metaheuristics, several researchers have addressed their application to the problem under consideration. Table 4 summarises the contributions classified according to the type of metaheuristic employed.

In addition to the classification between metaheuristics and descending local search, we would like to distinguish between general-neighbourhood and specific-neighbourhood approaches. General-neighbourhood approaches are the approaches whose neighbourhood may be suitable for any combinatorial problem, while specific-neighbourhood approaches are those where the definition of neighbourhood (including the allowed or preferred moves) incorporate some characteristics specific of the $F_m|prmu|C_{max}$-problem.

| Technique | Reference |
|---|---|
| Simulated Annealing | Osman and Potts (1989) |
| | Ogbu and Smith (1990, 1991) |
| | Zegordi et al. (1995). |
| Tabu Search | Widmer and Hertz (1989) |
| | Taillard (1990) |
| | Moccellin (1995) |
| | Nowicki and Smutnicki (1996) |
| Genetic Algorithms | Reeves (1993, 1995) |
| | Chen et al. (1995) |
| Ant Colony Optimisation | Stützle (1998) |
| Path algorithms | Werner (1993) |

*Table 4. Application of meta-heuristics to the $F_m|prmu|C_{max}$-problem*

Taking a metaheuristic like simulated annealing (SA) as example, a general-neighbourhood approach is the one of SA done by Osman and Potts (1989), where the neighbourhood is defined through pairwise interchange between jobs (given a solution, $\frac{n(n-1)}{2}$ neighbours can be explored). This neighbourhood is common to other combinatorial problems, such as the TSP, or the Quadratic Assignment Problem, among others. In contrast, a specific-neighbourhood approach of SA is the implementation by Zegordi et al. (1995), where the pairwise interchange is controlled by the value of an index named MDJ (Move Desirability of Jobs) that controls the subset of neighbours (within all possible $\frac{n(n-1)}{2}$ neighbours) to be explored. The MDJ index is constructed in each iteration of the simulated annealing procedure and it is a measure of the (possible) improvement in makespan by exchanging the position of the jobs in the current solution. Hence, although based on a standard definition of neighbourhood, this approach is problem-specific for the $F_m|prmu|C_{max}$-problem.

Using a general-neighbourhood cannot be regarded as a simpler approach to the problem in contrast to the more elaborated specific-neighbourhood approaches. First of all, it is known that the efficient implementation of a metaheuristic involves choosing and tuning several of its parameters, and that this process largely influences the heuristic's performance. From that point of view, the choice of a general/specific neighbourhood is only one of the multiple choices to be done, although we stress it here since it allows us to classify existing approaches to the problem. Second, the motivation of the approximate approaches is to obtain good solutions, and hence parameter design is justified itself by the results achieved by the heuristic, no matter how general or specific these are.

| | **Descending local search** | **Metaheuristic** |
|---|---|---|
| **General neighbourhood** | Aggarwal and Stafford (1975)<br>King and Spachis (1980)<br>Dannenbring – RAES (1977)<br>Page (1961) | Chen et al. (1995)<br>Moccellin (1995)<br>Ogbu and Smith (1990, 1991)<br>Osman and Potts (1989)<br>Reeves (1993, 1995)<br>Taillard (1990)<br>Stützle (1998)<br>Widmer and Hertz (1989) |
| **Specific neighborhood** | Ho and Chang (1991) | Nowicki and Smutnicki (1996)<br>Werner (1993)<br>Zegordi et al. (1995) |

*Table 5. Heuristics using phase III*

## 6. Conclusions. Future development of heuristics

In the previous sections, we have presented a framework to review and classify existing heuristics for makespan minimisation on permutation flowshops. This review may serve to obtain a comprehensive view of past developments and henceforth, to identify new directions for future research, either based on the development of new heuristics or in the extension/combination of current ones. These are summarised in table 6.

| Reference | Phase I Index development | Phase II Solution construction | Phase III Solution improvement |
|---|---|---|---|
| Aggarwal and Stafford (1975) | x | x (MS, MI) | x (DS, GN) |
| Bonney and Grundy (1976) | x | | |
| Campbell et al. (1970) | x ($F_2$\| \|$C_{max}$) | | |
| Chen et al. (1995) | | | x (MH[GA], GN) |
| Dannenbring (1977) – RA | x ($F_2$\| \|$C_{max}$) | | |
| Dannebring (1977) – RAES | x ($F_2$\| \|$C_{max}$) | | x (DS, GN) |
| Gupta (1968b) | x (TSP) | | |
| Gupta (1971b) | x | | |
| Gupta – MICOT, MINIT (1972) | x | x (MS, SI) | |
| Gupta – MINIMAX (1972) | | x (MS, SI) | |
| Gupta and Maykut (1973b) | x | x (MS, SI) | |
| Gupta (1976) | x | | |
| Gupta (1979) | x | x (MS, SI) | |
| Ho and Chang (1991) | | | x (DS, SN) |
| King and Spachis (1980) | | x (MS, SI) | x (DS, GN) |
| Koulamas – phase I (1998) | x | | |

*Table 6. Summary of existing heuristics*

*SI/MI: Single job insertion/Multiple job insertion*
*SS/MS: Single job selection/Multiple job selection*
*DS/MH[x]: Descending search/Metaheuristic. The term within square brackets may hold the following values: SA: Simulated Annealing /TS: Tabu Search/GA: Genetic Algorithm/ACO: Ant Colony Optimisation/PA: Path Algorithms*
*GN/SN: General neighbourhood/Specific neighbourhood*

| Reference | Phase I Index development | Phase II Solution construction | Phase III Solution improvement |
|---|---|---|---|
| Lai (1996) | x ($F_2$\| \|$C_{max}$) | | |
| McCormick et al. (1989) | x | x (MS, SI) | |
| Mocellin (1995) | x (TSP) | | x (MH[TS], GN) |
| Nawaz et al. (1983) | x | x (SS, MI) | |
| Nowicki and Smutnicki (1996) | | | x (MH[TS], SN) |
| Ogbu and Smith (1990, 1991) | | | x (MH[SA], GN) |
| Osman and Potts (1989) | | | x (MH[SA], GN) |
| Page (1961) | | | x (DS, GN) |
| Palmer (1965)/Hundal and Rajgopal (1988) | x | | |
| Rajendran (1993) | x | x (SS, MI) | |
| Reeves (1993, 1995) | | | x (MH[GA], GN) |
| Röck and Schmidt (1983) | x ($F_2$\| \|$C_{max}$) | | |
| Sarin and Lefoka (1993) | | x (MS, SI) | |
| Selim and Al-Turki (1987) | x ($F_2$\| \|$C_{max}$) | | |
| Sevas'janov (1995) | x (vector summation) | | |
| Stinson and Smith (1982) | x (TSP) | | |
| Stützle (1998) | | | x (MH[ACO], GN) |
| Taillard (1990) | | | x (MH[TS], GN) |
| Werner (1993) | | | x (MH[PA], SN) |
| Widmer and Hertz (1989) | x (TSP) | | x (MH[TS], GN) |
| Woo and Yim (1998) | | x (MS/MI) | |
| Zegordi et al. (1995) | | | x (MH[SA], SN) |

*Table 6 (continued). Summary of existing heuristics*

*SI/MI: Single job insertion/Multiple job insertion*
*SS/MS: Single job selection/Multiple job selection*
*DS/MH[x]: Descending search/Metaheuristic. The term within square brackets may hold the following values: SA: Simulated Annealing /TS: Tabu Search/GA: Genetic Algorithm/ACO: Ant Colony Optimisation/PA: Path Algorithms*
*GN/SN: General neighbourhood/Specific neighbourhood*

Nevertheless, the above classification itself provide little specific information to the schedulers on using any heuristic or combination of heuristics to solve a particular problem. To do so, an extensive comparison of the performance of existing heuristics is still needed, since most of the comparisons among heuristics presented in the literature only partially cover the current heuristics. Besides, the lack of an extensive comparison does not allow exploiting the possibilities for efficiently combining the best heuristics on each phase. These aspects are discussed in detail in the next subsections:

### *6.1. Comparison of existing heuristics*

A critical issue on heuristic development is the trade-off between the quality of the solution obtained by the heuristic, and the computation time required to obtain such a solution. However, in most of the practical environments, such a trade-off cannot be seen as a weighted function, but rather as a goal programming problem. Most of the time, the decision maker wishes to obtain the highest quality of the solution (the lowest makespan) subject to a time constraint – given by the decision interval. Since this decision interval might vary from seconds to hours, it is likely that there does not exist a universal 'recipe' covering all situations.

A second issue is the term 'quality of the solution' itself. First, we have to accept that this term can only be used within a statistical context. Therefore, it has to be taken into account Taillard's (1990) study on the distribution of the makespan values in the space of solutions. This study should be extended to cover a higher number of jobs, and to study the relationship of the distribution of the solutions for a given number of jobs, and the number of machines.

For the above reasons, every time a new heuristic was suggested in the literature, it was compared with the best available heuristics. Besides, some papers (e.g. Smith and Stafford 1980, Turner and Booth 1987, or Taillard 1990) are solely devoted to compare and rank the existing heuristics. However, in our opinion, both the latter studies as well as former papers introducing (and comparing) new heuristic present some of these problems:

1) *Lack of comprehensiveness*. Not all heuristics are covered and the rank or relative performance of some heuristics remains unknown. For instance, when Aggarwal and Stafford (1975) presented their new heuristic – namely AS –, their results in the experimental comparisons against existing heuristics lead to the conclusion that the AS heuristic outperforms CDS heuristic (Campbell et al. 1970) in terms of the quality of the solution. Nevertheless, after the NEH heuristic was introduced (Nawaz et al. 1983), the existing comparative studies claim NEH to be the best current heuristic in terms of the quality of the solution since it outperforms CDS (see e.g. Nawaz et al. 1983, Turner and Booth 1987, or Taillard 1990). However, in all previously mentioned studies, no indication is given on the relative performance NEH vs. AS in both terms of quality of the solution and computational requirements. Besides, other studies indicate that NEH is outperformed by other heuristics when $n$ is very large (Sarin and Lefoka 1993), or smaller than 20 jobs (Selim and Al-Turki 1987). Therefore, a most comprehensive comparison is required that covers all heuristics, or at least the most competitive ones.
2) *Little robustness of conclusions*. In most of the comparisons, the claiming of the superior performance of certain heuristics is based on the superior quality of the average makespan obtained over a set of problem instances. However, almost no indication on the confidence levels of the results is provided, and hence the significance of the differences between the average results could not be significant when requiring a high confidence level. Besides, only one paper (Smith and Stafford 1980) addresses the comparison issue from a hypothesis acceptance/rejection viewpoint. The lack of such analysis might seriously compromise the robustness of the conclusions.
3) *Weak/partial experimental design*. The basic parameters for the experimental design are the range of jobs and machines to be covered, and the way to generate the processing times. With respect to the latter issue, most of the studies use a uniform random distribution with high dispersion ([1,99] or [1,100]) to generate the processing times. This is known to produce more difficult problem instances (Campbell et al. 1970, Dannenbring 1977). In absence of a formal worst case analysis, this might serve as some 'statistical worst case analysis' (particularly when the average percentage deviation with respect to the optimum is provided) since it is believed to represent the most difficult instances for the heuristics.

However, this distribution of the processing times is unlike to occur in practice (Amar and Gupta 1986). On one hand, the Erlang-distribution is claimed in several studies to best map the processing times (see e.g. Conway et al. (1967), King and Spachis 1980, or Park et al. 1984). On the other hand, it is considered to be more realistic to employ a trend of the processing times of a job between machines, and a correlation between the processing times of a job, in the sense that, for a job, the processing times in the machines are consistently relatively large or relatively small (Rinnooy Kan 1976, Lageweg *et al*. 1978). Therefore, testbeds developed taking into account the above issues are expected to provide the performance of the different heuristics when applied to real-life problem instances, performance that could have been underestimated when applied to instances with random distribution of the processing times. Hence, testbeds such as the suggested by Taillard's (Taillard 1993) are not useful for this purpose, despite its acceptance as a benchmark in some studies (e.g. Taillard 1990, or Reeves 1995). First, the processing times of this test-bed are randomly generated, and secondly, the number of instances so generated is too small to make tight estimations on the confidence levels of the results. Other option would be to built a large test-bed of real-life instances, such as these described in Bestwick and Hastings (1976), or Lahiri *et al*. (1993).

*6.2. New heuristic development/improvement of existing heuristics*

With respect to future enhancements of the heuristics within the three phases presented in this paper, the phase of index development heuristics without using any kind of analogy (or alternatively, using a sorting analogy) seems not to have advanced too much recently. However, the first phase is required for both solution construction and improvement, and hence, in our opinion, its importance should not be underestimated. An important question to be investigated is the influence of the choices done in this phase on the subsequent phases.

When employing some form of analogy, the field remains open for new analogies or the improvement of the existing ones. Referring to the last issue, it is clear that new developments on optimal or heuristic procedures for the problems taken as analogy may influence the performance of this type of heuristics. However, whenever the question of

correspondence of the solutions between both problems cannot be answered in a formal way, there would be no guarantee of the performance of these heuristics for particular cases. Hence, we believe that correspondence between solutions of the employed analogy has to be better understood as a necessary step for improving the results of the heuristics based on analogies. It should be also mentioned that, with respect to the statistical correspondence, no work has been carried out in this field with the exception of the experiments by Lourenço (1996).

Regarding phase II heuristics, it has already been mentioned that with respect to job selection and job insertion, ties are not adequately handled in most of the heuristics. However, these ties are likely to occur, particularly when there is a multiple job insertion, since the insertion position of a non-critical job into the partial sequence might have no influence in the value of the partial property in which the selection is based. However, once this job has been inserted in one position or another, it certainly determines future values of the partial property for the subsequent non-scheduled jobs. Therefore, we believe that this issue has to be treated more intensively than in the past.

With respect to phase III heuristics, the stress has been done in the method to generate new solutions, adding to local search new and useful features such as memory, or ability to escape from local optima. It is foreseeable that as long as new general metaheuristics are proposed, these will be applied to the problem under consideration. However, when using general neighbourhood approaches, the question of which one is the most suitable has not been sufficiently addressed. Therefore, it could be useful to establish – at least from a statistical point of view – which neighbourhood structures are more efficient, both to incorporate them into a metaheuristic approach or simply to use them within a local descending search to guarantee the local optimality of the employed heuristic.

### *6.3. Extension/combination of existing heuristics*

At the sight of Table 6, it seems clear that the field for the development of more efficient approaches for the $F_m|prmu|C_{max}$ problem based on the extension or combination of existing heuristics remains open. Precisely the suggested classification – summarised in Table 6 – helps to locate single heuristics within a wider framework and hence facilitates the identification of these heuristics whose combination can improve

their individual results. Nevertheless, it has to be stressed again the need of a comprehensive comparison of existing heuristics to render the framework really useful to the scheduler. Until this comparison is done, little hope on the systematic development of the combination of heuristics can be deposited.

Within the extension/combination of existing heuristics we can include the decomposition/composition approaches discussed in section 2. One of the motivations of these approaches – explored in the beginning of 1970 and later abandoned – was the computing power available at that time, which made difficult even obtaining heuristic solutions for larger problems: "...larger machine scheduling problems can be partitioned into a number of smaller problems. Therefore, less computational effort is required" (Ashour 1970). This obstacle has been removed nowadays. It might be argued that the sub-problems containing a small number of jobs may nowadays be optimally solved – i.e. by some branch and bound procedure –, making the approach more appealing. However, it should be kept in mind that solving the sub-problems is just a part of the whole procedure, and that even in the case that sub-problems are optimally solved, obtaining an optimal solution for the original problem additionally requires an optimal decomposition (with respect to the number of sets and the jobs belonging to each set) as well as optimal composition (i.e. merging the solutions of the sub-problems). Therefore, since both last problems are rather complex and interdependent, it is difficult to think about forms of this approach being competitive with more straightforward procedures. Nevertheless, additional research in this area will be interesting under the light of real-time scheduling or rescheduling.

Finally, we would like to stress the fact that, although exact approaches for the permutation flowshop problem such as branch and bound techniques have been developed separately from the research on heuristics, the former are employed in practice as heuristic approaches since their allowed computation time is limited to the decision interval. In contrast, current branch and bound approaches are not designed to provide good solutions when stopped before finishing the exploration, and hence the efficient combination/integration of exact approaches and heuristics may appear as a promising field of research.

## 7. References


Aggarwal, S.C., and Stafford, E., 1975, A heuristic algorithm for the flowshop problem with a common sequence on all machines, *Decision Science*, 6, 237-251.

Amar, A.D., and Gupta, J.N.D., 1986, Simulated versus real life data in testing the efficiency of scheduling algorithms, *IIE Transactions*, 18, 16-25.

Ashour, S., 1967, A decomposition approach for the machine scheduling problem, *International Journal of Production Research*, 6, 109-122.

Ashour, 1970, A modified decomposition algorithm for scheduling problems, *International Journal of Production Research*, 8, 281-284.

Averbakh, I., and Berman, O., 1999, A simple heuristic for m-machine flow-shop and its applications in routing-scheduling problems, *Operations Research*, 47, pp. 165-170.

Azim, M.A., Moras, R.G., and Smith, M.L., 1989, Antithetic sequences in flow shop scheduling, *Computers and Industrial Engineering*, 17, pp. 353-358.

Bestwick, P., and Hastings, N., 1976, A new bound for machine scheduling, *Operations Research Quarterly*, 27, 479-487.

Bonney, M.C., and Gundry, S. W., 1976, Solutions to the constrained flowshop sequencing problem. *Operational Research Quarterly*, 27, 869.

Burns, F., and Rooker, J., 1976, Johnson's three machine flow-shop conjecture, *Operations Research*, 24, 578-580.

Campbell, H. G., Dudek, R. A. and Smith, M. L., 1970, A heuristic algorithm for the n-job, m-machine sequencing problem. *Management Science*, 16, B630-B637.

Carlier, J., and Rebaï, I., 1996, Two branch and bound algorithms for the permutation flowshop problem, *European Journal of Operational Research*, 90, 238-251.

Chen, C.L., Vempati, V.S., and Aljaber, N., 1995, An application of genetic algorithms for flow shop problems, *European Journal of Operational Research*, 80, 389-396.

Conway, R.W., Maxwell, W.L., and Miller, L.W., 1967, *Theory of scheduling*, Addison-Wesley, Reading, MA, USA.

Dannenbring, D. G., 1977, An evaluation of flow-shop sequence heuristics. *Management Science*, 23, 1174-1182.

Dudek, R.A., and Teuton, O.F., 1964, Development of m stage decision rule for scheduling n jobs through m machines, *Operations Research*, 12, 471-497.

Garey, M.R., Johnson, D.S., and Sethi, R., 1976, The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research*, 1, 117-129.

Graham, R. L, Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G., 1979, Optimisation and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, 5, 287-326.

Gupta, J.N.D., 1968a, Heuristic rules for n × m flowshop scheduling problem, *Opsearch* (India), 5, 165-170.

Gupta, J.N.D., 1968b, Travelling salesman problem – a survey of theoretical developments and applications, *Opsearch* (India), 5, 181-192.

Gupta, J.N.D., 1971a, Flowshop scheduling via sorting analogy, *UARI Research Report No. 109*, University of Alabama.

Gupta, J.N.D., 1971b, A functional heuristic for the flow-shop scheduling problem, *Operational Research Quarterly*, 22, 39-47.

Gupta, J.N.D., 1972, Heuristic algorithms for multistage flowshop scheduling problem, AIIE Transactions, 4, 11-18.

Gupta, J.N.D., 1979, A review of flowshop scheduling research, in *Disaggregation Problems in Manufacturing and Service Operations*, Martin Nijhoff Publishers, 363-388.

Gupta, J.N.D., and Maykut, 1973a, Flow-shop scheduling by heuristic decomposition, *International Journal of Production Research*, 11, 105-111.

Gupta, J.N.D., and Maykut, 1973b, Heuristic algorithms for scheduling n jobs in a flowshop, *Journal of the Operational Research Society of Japan*, 16, 131-150.

Gupta, J.N.D., 1976, A heuristic algorithm for the flowshop scheduling problem, *R.A.I.R.O. Recherche Operationnelle*, 10, 63-73.

Gupta, J.N.D., 1979, An improved lexicographic search algorithm for the flowshop scheduling problem, *Computers and Operations Research*, 6, 117-120.

Ho, J. C. and Chang, Y. L., 1991, A new heuristic for the n-job, m-machine flow-shop problem. *European Journal of Operational Research*, 52, 194-202.

Hundal, T.S. and Rajgopal, J., 1988, An extension of Palmer heuristic for the flow shop scheduling problem. *International Journal of Production Research*, 26, 1119-1124.

Johnson, S.M., 1954, Optimal two- and three-stage production schedule with setup times included, *Naval Research Logistics Quarterly*, 1, 61-68.

King, J.R., and Spachis, A.S., 1980, Heuristics for flow-shop scheduling, *International Journal of Production Research*, 18, 345-357.

Koulamas, C., 1998, A new constructive heuristic for the flowshop scheduling problem, *European Journal of Operational Research*, 105, 66-71.

Lageweg, B.J., Lenstra, J.K., and Rinnooy Kan, A.H.G., 1978, A general bounding scheme for the permutation flowshop problem, *Operations Research*, 26, 53-67.

Lahiri, S., Rajendran, C., and Narendran, T.T., 1993, Evaluation of heuristics for scheduling in a flowshop: a case study, *Production Planning and Control*, 4, 153-158.

Lai, T., 1996, A note on heuristics of flow-shop scheduling, *Operations Research*, 44, 648-652.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B., 1993, Sequencing and scheduling: algorithms and complexity, in *Logistics of Production and Inventory*, North-Holland, Amsterdam.

Lourenço, H.R., 1996, Sevast'yanov's algorithm for the flow-shop scheduling problem, *European Journal of Operational Research*, 91, 176-189.

McCormick, S. T., Pinedo, M. L., Shenker, S., and Wolf, B., 1989, Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research*, 37, 925-936.

Moccellin, J.V., 1995, A new heuristic method for the permutation flow-shop scheduling problem, *Journal of the Operational Research Society*, 46, 883-886

Morton, T.E., and Pentico, D.W., 1993, *Heuristic Scheduling Systems*, Wiley Interscience.

Nawaz, M., Enscore, E. E., and Ham, I., 1983, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA*, 11, 91-95.

Nowicki, E., and Smutnicki, C., 1996, A fast tabu search algorithm for the permutation flow-shop problem, *European Journal of Operational Research*, 91, 160-175.

Ogbu, F.A., y Smith, D.K., 1990, The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem, *Computers Operations Research*, 17, 243-253

Ogbu, F.A., y Smith, D.K., 1991, Simulated annealing algorithm for the permutation flowshop problem, *OMEGA*, 19, 64-67

Osman, I.H. and Potts, C.N., 1989, Simulated Annealing for Permutation Flow-shop Scheduling, *OMEGA*, 17, 551-557

Page, E.S., 1961, An approach to the scheduling of jobs on machines, *Journal of The Royal Statistical Society*, 323-484.

Palmer, D. S., 1965, Sequencing jobs through a multistage process in the minimum total time: a quick method of obtaining a near-optimum, *Operational Research Quarterly*, 16, 101-107.

Park, Y.B., Pegden, C.D., and Enscore, E.E., 1984, A survey and evaluation of static flowshop scheduling heuristics, *International Journal of Production Research*, 22, 127-141.

Pinedo, M., 1995, *Scheduling: theory, algorithms and systems* (Englewood Cliffs, NJ; Prentice Hall).

Potts, C.N., 1980, An adaptive branching rule for the permutation flowshop problem, *European Journal of Operational Research*, 5, 19-25.

Rajendran, C., 1993, Heuristic algorithm for scheduling in a flowshop to minimise total flowtime. *International Journal of Production Economics*, 29, 65-73.

Reeves, C. R., 1993, Improving the efficiency of tabu search for machine sequencing problems, *Journal of the Operational Research Society*, 44, 375-382.

Reeves, C. R., 1995, A genetic algorithm for flowshop sequencing, *Computers and Operations Research*, 22, 5-13.

Rinnooy Kan, A.H.G., 1976, *Machine scheduling problems*, Martinus Nijhoff, La Haya.

Röck, H., and Schmidt, G., 1983, Machine aggregation heuristics in shop-scheduling, *Methods of Operations Research*, 45, 303-314.

Sarin, S., and Lefoka, M., 1993, Scheduling heuristic for the n-job m-machine flow shop, *OMEGA*, 21, 229-234.

Selim, S.Z., and Al-Turki, U.M., 1987, A new heuristic algorithm for the flowshop problem, in *Modern Production Management Systems*, 91-96, A. Kusiak (Editor), Elsevier Science.

Sevast'janov, S., 1995, Vector summation in Banach space and polynomial algorithms for flow shops and open shops, *Mathematics of Operations Research*, 20, 90-103.

Smith A.W., and Stafford, E.F., 1980, A comparison of multiple criteria solutions of large-scale flow-shop scheduling problems, *AIDS 1980 Proceedings*, Las Vegas, Nevada, Vol. 2, pp. 35-37.

Sridhar, J. and Rajendran, C., 1996, Scheduling in flowshop an cellular manufacturing systems with multiple objectives – a genetic algorithmic approach, *Production Planning & Control*, 7, 374-382.

Stinson, J.P., 1977, A heuristic algorithm for obtaining an initial solution for the travelling salesman problem, *Working paper 77-12*, School of Management, Syracuse University, Syracuse, New York.

Stinson, J.P., and Smith, W., 1982, A heuristic programming procedure for sequencing the static flowshop, *International Journal of Production Research*, 20, 753-764.

Stützle, T., 1998, An ant approach for the flow shop problem, *EUFIT'98*, Aachen, Germany, 1560-1564.

Szwarc, W., 1971, Elimination methods in the $m \times n$ sequencing problem, *Naval Research Logistics Quarterly*, 18, 295-305.

Taillard, E., 1990, Some efficient heuristic methods for the flow-shop sequencing problem, *European Journal of Operational Research*, 47:65-74

Taillard, E., 1993, Benchmark for basic scheduling problems, *European Journal of Operational Research*, 64:278-285.

Turner, S. and Booth, D., 1987, Comparison of heuristics for flowshop sequencing, *OMEGA*, 15, 75-85.

Werner, F., 1993, On the heuristic solution of the permutation flow shop problem by path algorithms, *Computers and Operations Research*, 20, 707-722.

Widmer, M. and Hertz, A., 1991, A new heuristic method for the flow-shop sequencing problem. *European Journal of Operational Research*, 41, 186-193

Woo, D.S. and Yim, H.S., 1998, A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers and Operations Research*, 25, 175-182.

Zegordi, S.H., Itoh, K., and Enkawa, T., 1995, Minimizing makespan for flow-shop scheduling by combining simulated annealing with sequencing knowledge, *European Journal of Operational Research*, 85, 515-531