ELSEVIER

European Journal of Operational Research 113 (1999) 390–434

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

Deterministic job-shop scheduling: Past, present and future

A.S. Jain [1], S. Meeran *

Department of Applied Physics and Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, DD1 4HN, UK

Received 1 January 1997; accepted 1 March 1998

Abstract

Due to the stubborn nature of the deterministic job-shop scheduling problem many solutions proposed are of hybrid construction cutting across the traditional disciplines. The problem has been investigated from a variety of perspectives resulting in several analytical techniques combining generic as well as problem specific strategies. We seek to assess a subclass of this problem in which the objective is minimising makespan, by providing an overview of the history, the techniques used and the researchers involved. The sense and direction of their work is evaluated by assessing the reported results of their techniques on the available benchmark problems. From these results the current situation and pointers for future work are provided.

Keywords: Scheduling theory; Job-shop; Review; Computational study

1. Introduction

Current market trends such as consumer demand for variety, shorter product life cycles and competitive pressure to reduce costs have resulted in the need for zero inventory systems. However to maintain market share the system must be fast responding which implies that more stock has to be maintained. These conflicting requirements demand efficient, effective and accurate scheduling which is complex in all but the simplest production environments. As a result there is a great need for good scheduling algorithms and heuristics. Scheduling is essentially concerned with solving a Constraint Optimisation Problem (COP) and in the context of manufacturing it involves finding a sequential allocation of competing resources that optimises a particular objective function. The deterministic job-shop scheduling problem, hereinafter referred to as Π_J, is the most general of the classical scheduling problems.

Π_J consists of a finite set J of n jobs $\{J_i\}_{i=1}^n$ to be processed on a finite set $\mathscr{M}$ of m machines $\{\mathscr{M}_k\}_{k=1}^m$. Each job J_i must be processed on every machine and consists of a chain or complex of m_i operations $O_{i1}, O_{i2}, \ldots, O_{im_i}$, which have to be scheduled in a predetermined given order (precedence constraint). There are N operations in total, $N = \sum_{i=1}^n m_i$. O_{ik} is the operation of job J_i which

* Corresponding author. Tel.: 44 1382 344503; fax: 44 1382 345415; e-mail: s.meeran@dundee.ac.uk

[1] E-mail: a.z.jain@dundee.ac.uk

0377-2217/99/$ – see front matter
PII: S0377-2217(98)00113-1

has to be processed on machine $\mathscr{M}_k$ for an uninterrupted processing time period τ_{ik} and no operation may be pre-empted. [2] Each job has its own individual flow pattern through the machines which is independent of the other jobs. Each machine can process only one job and each job can be processed by only one machine at a time (capacity constraints). The duration in which all operations for all jobs are completed is referred to as the makespan $C_{\max}$. The only objective considered in this work is to determine starting times for each operation, $t_{ik} \geqslant 0$, in order to minimise the makespan while satisfying all the precedence and capacity constraints:

$$C^*_{\max} = \min(C_{\max}) = \min_{\text{feasible schedules}} (\max(t_{ik}+\tau_{ik}): \forall J_t \in J,\ m_k \in \mathscr{M}).$$

Note the dimensionality of each Π_J instance is specified as $n \times m$ and N is often assumed to be nm provided that $m_i = m$ for each job $J_i \in J$ and that each job has to be processed exactly once on each machine. In a more general statement of the job-shop problem, machine repetitions (or machine absence) are allowed in the given order of the job $J_i \in J$, and thus m_i may be greater (or smaller) than m. The main focus of this work is given to the case of $m_i = m$ non-pre-emptive operations per job J_i, unless otherwise stated.

Although makespan minimisation may not be perceived as a good theoretical objective function, its usage in academic and industrial practice is wide spread. This criterion has much historical significance and was the first objective applied by researchers in the early 1950s to the so-called easy problems which later were found to be combinatorially exploding. This criterion is simple to deal with from a mathematical point of view and is easy to formulate. Consequently it has been the principal criterion for academic research and is able to capture the fundamental computational difficulty which exists implicitly in determining an optimal schedule. A scheduling problem with the objective to minimise the makespan can be considered analogous to the well-known Travelling Salesman Problem (TSP) which is in a real sense a hypothetical problem. However solving either the TSP or Π_J will provide an insight into finding solutions to more practical problems which have non-regular performance measures. As the makespan minimisation problem is so well defined, with an abundance of available literature, it is simple to understand and hence it acts as a training problem to the more difficult and less well documented practical problems. Therefore as Π_J is an important model in scheduling theory serving as a proving ground for new algorithmic ideas and providing a starting point for more practically relevant models, it is worth understanding.

An $n \times m$ size Π_J has an upper bound of $(n!)^m$, thus a 20×10 problem may have at most 7.2651×10^{183} possible solutions. Complete enumeration of all these possibilities to identify feasible schedules and the optimal one is not practical. Due to this factorial explosion Π_J is considered to be a member of a large class of intractable numerical problems known as NP-hard (NP stands for non-deterministic polynomial). Here we digress to explore the computational complexity. If x is the size of the input and y is a constant then a problem with a time requirement of order $O(x^y)$ is said to have a polynomial complexity and is denoted by P. Whereas if the problem complexity is of order $O(y^x)$ then it is considered to be NP-hard (if $P \neq NP$) (Cook, 1971; Garey et al., 1976). Here an optimal algorithm would require a number of computational steps that grows exponentially with the input. The common belief is that no efficient algorithm can ever possibly be found to solve these inherently hard problems. Even most of the special cases of Π_J are also NP-hard or strongly NP-hard which makes this problem one of the most stubborn members of this class (Nakano and Yamada, 1991; Lawler et al., 1993). This is highlighted by the fact that algorithms can optimally solve other NP instances such as randomly generated travelling salesman problems, with more than 4000 cities,

[2] In this analysis each operation is given a unique number from 1 to $(m*n)$. If $\mathscr{M}_k$ is the lth machine required by O_{ik}, with respect to precedence constraints, then its operation number is $n(l-1)+i$ (cf. Table 4 and Fig. 3).

and set covering problems with tens of thousands of variables. However, as yet strategies have not been devised that can guarantee optimal solutions for Π_J instances which are larger than 20×10.

This work aims to explore the research direction within Π_J examining the origins of the problem; its current standing as well as providing suggestions for areas that deserve investigation. The paper is organised as follows: after detailing the history of the problem, attention is focused on principal techniques used in this field. The best solutions are then given for the 242 benchmark problems available in the literature. The paper then explores parameters that allow an improved comparison to be made between techniques and a computational comparison is provided on some of the best methods available. The paper is then concluded with pointers for future work.

2. The phases of Π_J history – the techniques applied

Opinion in the Π_J field is mixed about who first proposed the job-shop problem in its current form. Roy and Sussmann (1964) were the first to propose the disjunctive graph representation and Balas (1969) was the first to apply an enumerative approach based on the disjunctive graph. Nevertheless there were earlier works in job-shop scheduling. Giffler and Thompson (1960) proposed a priority dispatch rule template, Jackson (1956) generalised Johnsons's flow-shop algorithm (Johnson, 1954) to the job-shop and Akers and Friedman (1955) applied a Boolean Algebra approach in order to represent processing sequences. These works dealt with a problem consisting of n jobs, m machines and precedence relationships. Thus each job in this problem is processed through the machines in a different order. The objective in all the above works is to complete all the tasks as quickly as possible. Not surprisingly these works cite even earlier references. For example Akers and Friedman (1955) cite a working paper from 1951 concerning industrial production (Salvesen and Anderson, 1951). Although it is not clear who can be credited with first proposing Π_J, it is widely accepted that the book "Industrial Scheduling" edited by Muth and Thompson which collated all the major findings at that time, is the basis for most subsequently conducted research.

Fig. 1 summarises the main techniques applied to solve Π_J and the category each technique belongs to, i.e. whether it is an approximation or optimisation method and whether it is constructive, builds a solution from the problem data, or iterative, modifies a solution by continually reordering the sequence of operations. Also given are the benchmark problems on which these techniques have been applied to. Tables 1 and 2 summarise the main researchers, who have applied these various techniques to solve Π_J, and the most common representation and classification methods. Although the Gantt chart (cf. Fig. 3), described in Gantt (1919), Clark (1922) and Porter (1968), has traditionally been the most popular method of solution representation Blazewicz et al. (1996) indicate that the disjunctive graph model, $\mathcal{G} = \{\mathcal{N}, \mathcal{A}, \mathcal{E}\}$ (Roy and Sussmann, 1964) is now more prevalent. We take this opportunity to describe the disjunctive graph representation in the following before returning to the discussion on classification methods.

In this node-weighted graph there is a vertex for each operation where $\mathcal{N}$ is the set of nodes representing operations to be processed on the set of machines $\mathcal{M}$. Included within $\mathcal{N}$ are two special (fictitious) nodes ⓪ and ☆ which correspond to the initial and final operations respectively and are known as the source and sink: $\mathcal{N} = \{⓪, 1, 2, \ldots, ☆\}$ The positive weight of each node j is equivalent to the processing time, τ_j, of the corresponding operation, where $\tau_⓪ = \tau_☆ = 0$. The starting and completion times of these nodes represents the start and finishing times of Π_J respectively. ⓪ is connected to the initial operation of each job and similarly the final operation of each job is connected to ☆.

$\mathcal{A}$ is the set of directed conjunctive arcs representing the precedence constraints for each job, such that $(i, j) \in \mathcal{A}$ indicates that operation i is an immediate predecessor of operation j $(i \prec j)$ in the chain of operations of the job. Capacity constraints ensure that two operations i and j processed by the same machine cannot be executed simultaneously. Such operations belong to a set $\mathcal{E}$ of uni-directed but orientable edges where each member of $\mathcal{E}$ is associated with a pair of disjunc-

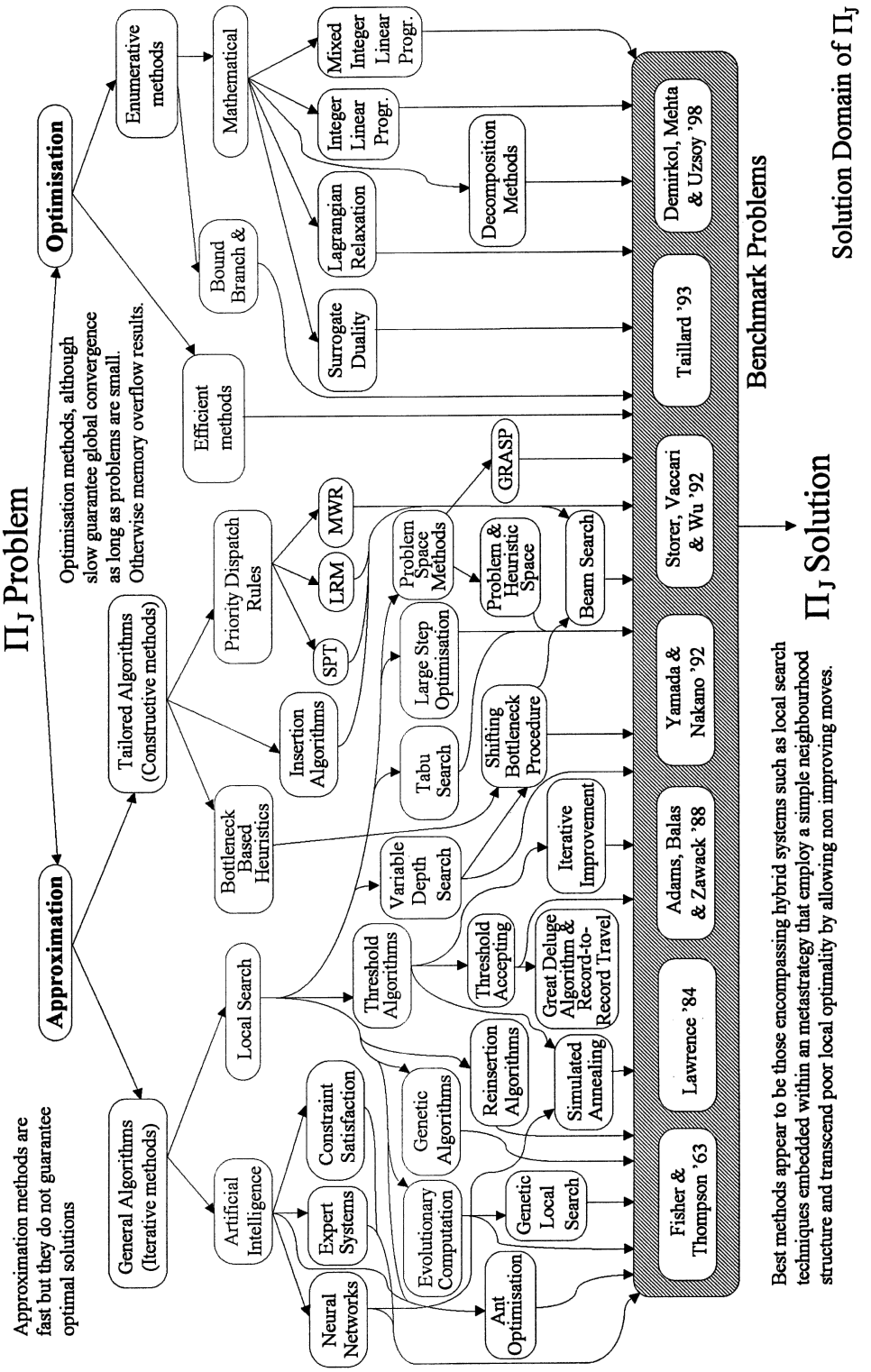

Fig. 1. The phases of Π_J research.

Table 1
Main methodologies used to solve Π_J

Method	Authors	
Problem Representation		
Gantt Chart	Gantt (1919)	Clark (1922)
	Porter (1968)	
Disjunctive Graph	Roy and Sussmann (1964)	Sussmann (1972)
	Radermacher (1985)	Bartusch et al. (1988)
	White and Rogers (1990)	Sotskov (1997)
Problem Classification	Conway et al. (1967)	Graham et al. (1979)
$\mathscr{NP}$ Classification	Cook (1971)	Garey et al. (1976)
	Lenstra et al. (1977)	Gonzalez and Sahni (1978)
	Garey and Johnson (1979)	Lenstra and Rinnooy Kan (1979)
	Sotskov (1991)	Timkovsky (1995)
	Sotskov and Shakhlevich (1995)	Brucker et al. (1996a, b)
Optimisation Algorithms	Lawler et al. (1993)	
Efficient Methods – Solvable in	Johnson (1954)	Akers (1956)
Polynomial Time	Jackson (1956)	Szwarc (1960)
	Hardgrave and Nemhauser (1963)	
	Hefetz and Adiri (1982)	Sotskov (1985)
	Brucker (1988, 1994)	Kravchenko (1995)
	Kravchenko and Sotskov (1996)	Brucker et al. (1997a, b)
	Williamson et al. (1997)	Kubiak and Timkovsky, 1996
Enumerative Methods	Lenstra (1976)	Rinnooy Kan (1976)
Mathematical Formulations	Lawler (1983)	Blazewicz et al. (1991)
(Mixed) (Integer) Linear Programming	Bowman (1959)	Wagner (1959)
	Balas (1965)	Dyer and Wolsey (1990)
	Van den Akker (1994)	
Decomposition Techniques	Ashour (1967)	Kanzedal (1983)
	Chu et al. (1992)	Krüger et al. (1995)
(Augmented) Lagrangian Relaxation	Fisher (1973a, b, 1976)	Fisher et al. (1975)
	Hoitomt et al. (1993)	Van De Velde (1991)
	Hoogeveen and Van De Velde (1995)	
Surrogate (Constraint) Duality	Glover (1968, 1975, 1977)	Fisher et al. (1983)
Miscellaneous	Balas (1979, 1985)	Applegate and Cook (1991)
Branch and Bound (BB)	Lageweg et al. (1977)	Pinson (1995)
Disjunctive Graph	Brooks and White (1965)	Greenberg (1968)
	Balas (1969)	Charlton and Death (1970a, b)
	Florian et al. (1971)	Nabeshima (1971)
	Ashour and Hiremath (1973)	Ashour and Parker (1973)
	Bratley et al. (1973)	Ashour et al. (1974)
	McMahon and Florian (1975)	Barker and McMahon (1985)
	Applegate and Cook (1991)	Carlier and Pinson (1989, 1990, 1994)
	Brucker et al. (1994)	Perregaard and Clausen (1995)
	Boyd and Burlingame (1996)	
Time Orientation	Martin (1996)	

tive arcs, requiring a common machine, such that $[i,j] = \{(i \prec j),(j \prec i)\}$ and $\{i,j \in O\}$. An example of a 4×3 disjunctive graph is provided in Fig. 2 which represents the problem defined in Table 3. The conjunctive arcs, which are members of $\mathscr{A}$, are shown by complete lines while the dotted and dashed lines represent edges in the disjunctive set $\mathscr{E}$. Extensions and limitations of this model are provided by White and Rogers (1990) who indicate that applying this model to industrial situations can be difficult as parallel processing and indefinite cyclical process flows cannot be modelled directly. Nevertheless the same authors (White and Rogers, 1990) succeeded in extending the disjunctive graph

Table 2

Method	Authors	
Approximation Algorithms	Fisher and Rinnooy Kan (1988)	Blazewicz et al. (1996)
Constructive Methods		
Priority Dispatch Rules	Jackson (1955)	Smith (1956)
	Giffler and Thompson (1960)	Fisher and Thompson (1963)
	Crowston et al. (1963)	Jeremiah et al. (1964)
	Gere (1966)	Moore (1968)
	Panwalkar and Iskander (1977)	Blackstone et al. (1982)
	Lawrence (1984)	Haupt (1989)
	Chang et al. (1996)	Sabuncuoglu and Bayiz (1997)
Insertion Algorithms	Werner and Winkler (1995)	
Bottleneck based heuristics	Alvehus (1997)	
Shifting Bottleneck Procedure (SBP)	Adams et al. (1988)	Applegate and Cook (1991)
	Dauzère-Pérès and Lasserre (1993)	Balas et al. (1995)
	Balas and Vazacopoulos, 1998	Holtsclaw and Uzsoy (1996)
	Demirkol et al. (1997)	
Iterative Methods		
Artificial Intelligence (AI)	Glover and Greenberg (1989)	
Constraint Satisfaction (CSPs)	Erschler et al. (1976)	Fox (1987)
	Sadeh (1991)	Caseau and Laburthe (1994, 1995)
	Nuijten and Aarts (1994, 1996)	Harvey (1995)
	Harvey and Ginsberg (1995)	Sadeh et al. (1995)
	Baptiste and Le Pape (1995)	Baptiste et al. (1995)
	Pesch and Tetzlaff (1996)	Sadeh and Fox (1996)
	Cheng and Smith (1997)	Nuijten and Le Pape (1998)
Neural Networks (NNs)	Wang and Brunn (1995)	Jain and Meeran, 1998
Hopfield Networks	Foo and Takefuji (1988a–c)	Zhou et al. (1990, 1991)
	Van Hulle (1991)	Lo and Bavarian (1993)
	Willems and Rooda (1994)	Satake et al. (1994)
	Foo et al. (1994, 1995)	Sabuncuoglu and Gurgun (1996)
Back-Error Propagation (BEP)	Dagli et al. (1991)	Watanabe et al. (1993)
	Cedimoglu (1993)	Sim et al. (1994)
	Kim et al. (1995)	Dagli and Sittisathanchai (1995)
Expert Systems (ESs)	Alexander (1987)	Kusiak and Chen (1988)
	Biegel and Wink (1989)	Charalambous and Hindi (1991)
	Shakhlevich et al. (1996)	Sotskov (1996)
Ant Optimisation (AO)	Colorni et al. (1994)	
Local Search	Evans (1987)	Vaessens (1995)
	Vaessens et al. (1995, 1996)	Aarts and Lenstra (1997)
	Mattfeld et al. (to appear)	
Problem Space Methods		
Problem & Heuristic Space	Storer et al. (1992, 1995)	
GRASP	Resende (1997)	
Genetic Algorithms (GAS)	Davis (1985)	Falkenauer and Bouffouix (1991)
	Nakano and Yamada (1991)	Tamaki and Nishikawa (1992)
	Yamada and Nakano (1992)	Davidor et al. (1993)
	Ross et al. (1993)	Mattfeld et al. (1994)
	Della Croce et al. (1995)	Kobayashi et al. (1995)
	Norman and Bean (1995)	Bierwirth (1995)
	Bierwirth et al. (1996)	Cheng et al. (1996)
	Shi (1997)	

Table 2 (continued)

Method	Authors	
Genetic Local Search (GLS)	Aarts et al. (1991, 1994)	Pesch (1993)
	Della Croce et al. (1994)	Dorndorf and Pesch (1995)
	Mattfeld (1996)	Yamada and Nakano (1995b, 1996b, c)
Reinsertion Algorithms	Werner and Winkler (1995)	
Threshold Algorithms	Aarts et al. (1991, 1994)	
Iterative Improvement (IM)	Aarts et al. (1991, 1994)	Storer et al. (1992)
Simulated Annealing (SA)	Matsuo et al. (1988)	Van Laarhoven et al. (1988, 1992)
	Aarts et al. (1991, 1994)	Yamada et al. (1994)
	Sadeh and Nakakuki (1996)	Yamada and Nakano (1995a, 1996a)
	Kolonko (to appear)	
Threshold Acceptance (TA)	Aarts et al. (1991, 1994)	
Large Step Optimisation	Lourenço (1993, 1995)	Lourenço and Zwijnenburg (1996)
	Brucker et al. (1996a, 1997a)	
Tabu Search (TS)	Taillard (1989, 1994)	Dell'Amico and Trubian (1993)
	Barnes and Chambers (1995)	Sun et al. (1995)
	Nowicki and Smutnicki (1996)	Ten Eikelder et al. (1997)
	Thomsen (1997)	

model to represent regular performance criteria as well as assembly and disassembly operations, due dates, schedule maintenance, ready times, priority jobs, certain sequence-dependent set-up times and material handling delays.

The most popular method of classifying scheduling problems is the four field notation ($A/B/C/D$) of Conway et al. (1967): A is the number of jobs, B the number of machines, C the flow pattern within the machine shop and D the performance measure by which the schedule is evaluated. While this descriptive technique is suitable for basic problems, when non-basic problems (involving pre-emption, dependent jobs, etc.) require classification then the three field notation ($\alpha/\beta/\gamma$) of Graham et al. (1979) is more appropriate: α is the flow pattern and the number of machines, β the constraints on the jobs and γ the scheduling criteria. MacCarthy and Liu (1993) indicate that the four field technique has been widely used and is familiar to most scheduling researchers. Consequently they propose a combination of the two methods where the C field is modified to take into account non-basic models.

It can be safely assumed that the work on Π_J was initiated by Johnson (1954) which is believed to be one of the earliest works in scheduling theory. An optimal algorithm for a two machine flow-shop was developed in this research. Then other efficient methods, solvable in polynomial time, for the job-shop problems of the sizes $2 \times m$ (Akers, 1956), $n \times 2$ (where there are no more than 2 operations per job) (Jackson, 1956) and $n \times 2$ (where all operations are of unit processing time) (Hefetz and Adiri, 1982) have been proposed. In general one of the most productive periods in the domains of Management Science and Operations Research was the 1950s. During this time many problems were solved by the application of crude but effective heuristics which formed the basis of the development of classical scheduling theory.

During the 1960s the emphasis shifted and was directed at finding exact solutions by the application of enumerative algorithms which adopted more elaborate and sophisticated mathematical constructs. Although these strategies are of tremendous theoretical value providing a significant research accomplishment, as Rinnooy Kan, (1976,

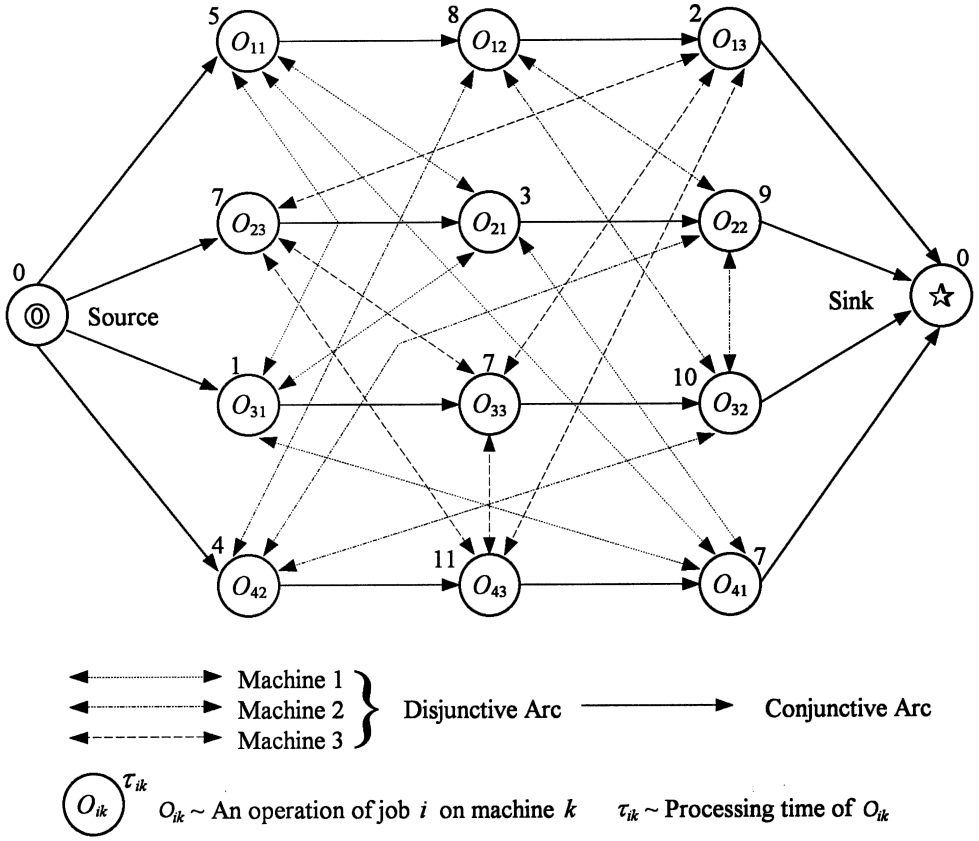

Fig. 2. A Disjunctive graph representation of a 4×3 instance given in Table 3.

p. 36) indicates "*a natural way to attack scheduling problems is to formulate them as mathematical programming models*", the results from many of these works are extremely disappointing. The majority of these techniques are unable to achieve feasible solutions to many problems and are therefore of limited practical use. Consequently they are only applied in the calculation of lower bounds.

The main enumerative strategy is Branch and Bound (BB) where a dynamically constructed tree representing the solution space of all feasible schedules is implicitly searched. This technique formulates procedures and rules to allow large

Table 3
Processing times and operation orders for a 4 × 3 instance

Operation number	JOB							
	J_1		J_2		J_3		J_4	
	$\mathcal{M}_i$	τ_i	$\mathcal{M}_i$	τ_i	$\mathcal{M}_i$	τ_i	$\mathcal{M}_i$	τ_i
1–4	1	5	3	7	1	1	2	4
5–8	2	8	1	3	3	7	3	11
9–12	3	2	2	9	2	10	1	7

portions of the tree to be removed from the search and for many years it was the most popular Π_J technique. Although this method is very suitable for instances with $N < 250$ its excessive computing requirement prohibits its application to problems of interest. In addition their performance is quite sensitive to individual instances and initial upper bound values (Lawler et al., 1993). Current research emphasises the construction of improved branching and bounding strategies and the generation of more powerful elimination rules in order to remove large numbers of nodes from consideration at early stages of the search.

During the 1970s and until the mid 1980s the emphasis was placed on justifying the complexity. Numerous works spanning from Cook (1971) to Lawler et al. (1982) clearly highlighted the extreme intractability of Π_J demonstrating that only a very few special instances are solvable in polynomial time. As a result Parker (1995) refers to the era before 1970 as BC (Before Complexity). Consequently we designate the era since then as AD (Advanced Difficulty). The AD research soon showed that the problems which were solved as a specific instance in the 1950s as well as many others are NP-Hard problems if they are generalised. Hence not surprisingly Garey and Johnson (1979) cite 320 NP-Hard problems.

Due to the general limitation of exact enumeration techniques, approximation methods became a viable alternative. While such methods forego guarantees of an optimal solution for gains in speed they can be used to solve larger problems. The earliest approximation algorithms are priority dispatch rules (pdrs). These construction techniques assign a priority to all the operations which are available to be sequenced and then choose the operation with the highest priority. They are very easy to implement and have a low computation burden. A plethora of different rules have been created (Panwalkar and Iskander, 1977) and the research applied in this domain indicates that the best techniques involve a linear or randomised combination of several priority dispatch rules (Fisher and Thompson, 1963; Panwalkar and Iskander, 1977; Lawrence, 1984). Two of the more novel approaches involve Fuzzy Logic (Grabot and Geneste, 1994) and Genetic Local Search (Dorndorf and Pesch, 1995). Nevertheless these works highlight: the highly problem dependent nature of pdrs, as in the case of makespan minimisation no single rule shows superiority; their myopic nature in making decisions, as they only consider the current state of the machine and its immediate surroundings and that solution quality degrades as problem dimensionality increases.

By the end of the 1980s the full realisation of this advanced difficulty resulted in the main research focus being diverted away from emphasising the NP-Hard status of Π_J and creating optimisation techniques. Instead attention concentrated on solving this problem by the application of approximation methods. However due to the general deficiencies exhibited by priority dispatch rules there was a growing need for more appropriate techniques which apply a more enriched perspective. Work on such applications was initiated by Fisher and Rinnooy Kan (1988) who emphasise the fundamental properties of generating good heuristic techniques: design, analysis and implementation. The suitability of these techniques is further highlighted by the Committee On the Next Decade of Operations Research (CONDOR, 1988) who indicate that they are extremely promising. While the survey of Rodammer and White (1988) emphasises the flexibility and strength of such heuristic approaches in comparison with optimisation techniques. They indicate innovative heuristic search procedures are legitimate application tools for not just Π_J but more real world scheduling problems. Not surprisingly with this new research emphasis substantial progress was made and in the short period of 1988–1991, which we shall refer to as the boom period, some of the most innovating algorithms were formulated.

Glover and Greenberg (1989) indicate the emergence of heuristic strategies to difficult problems, such as Π_J, that are inspired by natural phenomena and intelligent problem solving. Although research in the boom period was mainly focused at approximation methods some effort was still being directed at optimisation methods, however applying principles derived from these new heuristic strategies. For example Carlier and Pinson (1989) prescribed a BB method which proved

for the first time the optimal solution to FT 10 (see Section 3.1), a problem that had been eluding researchers for over 25 years. Even larger instances than FT 10 were solved by other BB methods (Carlier and Pinson, 1990; Applegate and Cook, 1991). Prior to 1988 the only technique available to solve instances with more than 100 operations was priority dispatch rules (Lawrence, 1984).

Fox (1987) and Sadeh (1991) were tackling problems from an industrial perspective using Constraint Satisfaction techniques while neural network (NN) applications (Foo and Takefuji, 1988a–c; Zhou et al., 1990, 1991) provided instantaneous solutions to problems as large as 20×20.

Examples of some of the innovative algorithms formulated during the late 1980s and early 1990s to solve Π_J include Large Step Optimisation (Martin et al., 1989); Tabu Search (TS) (Glover, 1989, 1990; Taillard, 1989); Simulated Annealing (SA) (Matsuo et al., 1988 Van Laarhoven et al., 1988 Aarts et al., 1991); Genetic Algorithms (GAS) (Falkenauer and Bouffouix, 1991; Nakano and Yamada, 1991) and Genetic Local Search (GLS) (Moscato, 1989; Aarts et al., 1991) which are described later in this paper. The main contribution of these works is the notion of local search (Johnson et al., 1988; Yannakakis, 1990) to Π_J. In local search methods a metastrategy is able to guide a myopic algorithm to optimality by accepting non-improving solutions. The works of Van Laarhoven et al., 1988; Matsuo et al., 1988, which were derived from Balas (1969), and Grabowski et al. (1988) laid the foundations for powerful Π_J neighbourhood structures which use local search techniques that are respectively based on the permutation of critical operations and the notion of blocks.

The boom period was started by the technique that has had the greatest influence on approximation methods and was the first heuristic to solve FT 10, the Shifting Bottleneck Procedure (SBP) (Adams et al., 1988). One of the main reasons this constructive algorithm achieves good results is due to the well-developed algorithms and software for the one-machine scheduling problem. SBP is characterised by the following tasks: Subproblem identification, bottleneck selection, subproblem solution, and schedule reoptimisation (Demirkol et al., 1997). The actual strategy involves relaxing the problem into m one machine problems and solving each subproblem one at a time. Each one machine solution is compared with all the others and the machines are ranked on the basis of their solution. The machine having the largest lower bound is identified as the bottleneck machine. SBP sequences the bottleneck machine first, with the remaining unsequenced machines ignored and the machines already scheduled held fixed. Every time the bottleneck machine is scheduled each previously sequenced machine susceptible to improvement is locally reoptimised by solving the one machine problem again. The one machine problem is iteratively solved using the approach of Carlier (1982) which provides an exact and rapid solution. The main contribution of this approach is the way the one machine relaxation is used to decide the order in which machines should be scheduled. A comprehensive computational analysis of this method is provided by Holtsclaw and Uzsoy (1996) and Demirkol et al. (1997) and a thorough review is given by Alvehus (1997).

Nevertheless the fundamental problem with SBP is the difficulty in performing reoptimisation and the generation of infeasible solutions. Dauzère-Pérès and Lasserre (1993) and Balas et al. (1995) note substantial differences in the results of Adams et al. (1988) when applying 4 rather than 3 reoptimisation cycles. This is because the work of Adams et al. (1988) does not take into account delayed precedence constraints (DPCS), which are relations between non-independent operations on the same machine. As a result Dauzère-Pérès and Lasserre (1993) propose a heuristic strategy while Dauzère-Pérès (1995) and Balas et al. (1995) utilise an exact scheme to deal with DPCS. The most recent work (Balas and Vazacopoulos, 1998) amalgamates guided variable search with SBP producing one of the best Π_J approaches available. Even though Balas and Vazacopoulos (1998) suggest several elaborate reoptimisation schemes as yet there is no strategy to indicate how this should be done. In addition no method is available to decide the size of the subproblem or which machine(s) to fix and in order to solve problems

where job routings are more structured SBP will need to be modified.

Nevertheless as SBP generated much of the initial excitement in the boom period then not surprisingly this procedure has been incorporated in many other works (Caseau and Laburthe, 1995 Balas and Vazacopoulos, 1998; Yamada and Nakano, 1996a; Vaessens, 1996) which have improved the upper and lower bounds of several hard problems. SBP has also been applied to many generalisations of Π_J. For example Morton (1990) extends SBP to project scheduling and applies several different performance criteria. Ovacik and Uzsoy (1992, 1996) apply this technique to a semiconductor testing facility. Ramudhin and Marier (1996) adapt the procedure to assembly, dag and open shop applications while Ivens and Lambrecht (1996) extend SBP to deal with a variety of parameters such as set-up time and parallel machines. The most recent generalisation is by Balas et al. (1998) who combine their exact one machine approach with their guided local search procedure and apply this to the job-shop problem with deadlines.

As indicated earlier the euphoria generated in the boom period resulted in the creation and formalisation of many approximation methods which shall now be described.

The insertion algorithm of Werner and Winkler (1995) consists of two phases. The first phase applies a constructive strategy in which an operation is positioned in the schedule such that it minimises the length of the longest path passing through it. The second phase applies a reinsertion strategy to iteratively improve the initial solution. In both phases beam search (Morton and Pentico, 1993) is applied to improve the search. Beam search incorporates heuristics into the search preventing the need to follow every possible selection from a given point thereby trying to keep the combinatorial nature of the problem in check. The search is performed in a breadthwise fashion but without backtracking.

If a beam width of β is applied then at each step the β best possibilities of a given partial schedule are chosen for further consideration. This allows a limited number of partial sequences to be evaluated further. A filter width α can also be introduced such that a partial evaluation is made of all β descendants and then α of the β most promising descendants are evaluated fully to determine the beam node. Once the beam node is selected the process then continues by selecting the β best descendants for further evaluation and finishes once a complete selection is made. Such an approach is similar to the fan candidate list strategy in tabu search (Glover and Laguna, 1997) which also allows a limited number of solutions to be searched in parallel. As in BB strategies best descendants are chosen based on lower bound calculations. A filtered beam search strategy has been superimposed by Sabuncuoglu and Bayiz (1997) on selection decisions made by a set of pdrs and results indicate that their technique provides substantial improvement over the results achieved purely by pdrs.

Constraint Satisfaction techniques are examples of iterative approximation methods which apply many of the rules and strategies used in branch and bound algorithms. They aim at reducing the effective size of the search space by applying constraints that restrict the order in which variables are selected and the sequence in which possible values are assigned to each variable. Although these techniques are concerned with trying to achieve a feasible schedule such that the overall deadline can be met many of these methods have difficulty representing constraints and require excessive backtracking. As a result they commonly converge to dead end states. Despite the ebullience witnessed in the boom period generally poor results have been achieved by these methods while the computational effort required is high (Caseau and Laburthe, 1995; Pesch and Tetzlaff, 1996; Nuijten and Le Pape, 1998). Similar conclusions are made about neural networks, which provide the capability to perform distributed processing using a simple but massively interconnected structure of parallel processing units, and only the work of Sabuncuoglu and Gurgun (1996) has been successfully applied to the benchmark problems.

Other iterative methods of note are problem-space methods which generate many different starting solutions, using fast problem-specific constructive procedures, which are then improved

by local search. This class of techniques include problem and heuristic space based search (Storer et al., 1992, 1995) and greedy random adaptive search procedure (GRASP) (Resende, 1997).

The most popular iterative methods are Threshold Algorithms which choose a new configuration if the difference in cost between the current solution and a neighbour is below a given threshold. Members of this group of algorithms include iterative improvement, simulated annealing and threshold acceptance. Due to the limitations of iterative improvement and the relative infancy of threshold acceptance, simulated annealing is the most popular technique in this category and has been applied extensively to Π_J. In simulated annealing (SA) the thresholds are positive and stochastic. SA is a random oriented local search technique that was introduced as an analogy from statistical physics of the computer simulation of the annealing process of a hot metal until its minimum energy state is reached. However as SA is a generic technique it is unable to quickly achieve good solutions to Π_J problems. As a result research is currently directed at combining SA with other methods in order to improve results and reduce the required computing time. Amalgamation with critical neighbourhoods and active schedule generation (Yamada and Nakano, 1995a, 1996a) and genetic algorithms (Kolonko, to appear) has made SA competitive with other methods with respect to solution quality but they still require excessive computational effort.

While SA is based on principles of physical science, Genetic Algorithms (GAs) are search techniques based on an abstract model of natural evolution such that the quality of individuals builds to the highest level compatible with the environment (constraints of the problem). Despite the fact that many elaborate schemes have been proposed GAs are unable to successfully represent Π_J and crossover operators cannot generate feasible schedules without losing their efficiency. In addition many GA methods are unable to converge to an optimal solution. These deficiencies initiated the work on Genetic Local Search (GLS), which is also referred to as population based local search or memetic search (Grefenstette, 1987; Moscato, 1989; Ulder et al., 1990, 1991). GLS is a two level local search technique where a child conceived from the GA operators is used as the initial solution for the subsequent local search. The local search directs the offspring to the nearest locally optimal point which is operated on in the next generation by the traditional genetic recombination operators. The most recent works (Mattfeld, 1996; Yamada and Nakano, 1996b, c) have been able to overcome representation and crossover difficulties as well as poor solution convergence suffered by traditional GAs.

Large Step Optimisation, as GLS, is an example of a bilevel local search technique. Developed by Martin et al. (1989, 1992) it encompasses a dual phase optimisation method consisting of a large optimised transition (large step) and then a local search method (small step). Small steps are most commonly performed by iterative improvement or SA. Large steps involve the application of problem specific techniques allowing local minima to be transcended even at low temperatures. It is a relatively new technique and has only been applied to Π_J by Lourenço (1993, 1995) and Lourenço and Zwijnenburg (1996). The limited analysis indicates that this technique has a very high computational requirement but provides better results than if only the local search method is applied. A similar type of bilevel strategy has also been applied by Brucker et al. (1996a, 1997a), to a wider domain of scheduling results achieving some promising results.

Tabu Search (TS) proposed by Glover (1977, 1986, 1989, 1990) is an iterative approximation approach which has led to several successful formulations for Π_J. TS is a simple technique that intelligently guides a search process away from solutions that (based on available information) appear to duplicate or resemble previously achieved solutions. More elaborate schemes can be applied to intensify the search in areas which have historically been good or diversify the search to unexplored regions of the solution space. The technique of Nowicki and Smutnicki (1996) is currently one of the most powerful TS approaches allowing good solutions to be achieved very quickly. A computational analysis is provided for many of these approaches

(cf. Tables 15–17) indicating their strengths and weaknesses.

It is clearly evident that work appears to have gone full circle returning to the heuristics notion of the 1950s however with some novel and elaborate modifications. Currently the most dominant methods are of hybrid construction and transcend poor minima by integrating problem specific local heuristics with general metasolvers. To underline the fact that work continues to follow the same cyclic phases, research has currently returned to complexity analysis and the construction of polynomial time ρ-approximation algorithms that guarantee solutions of $\rho C^*_{\max}$. For example, Shmoys et al. (1994) have constructed an algorithm with a worst case performance guarantee of $(\prod_{\max} + (m-1)\tau_{\max})$, where $\prod_{\max} = \max_{\mathscr{M}_k} \sum_k \tau_{ik}$. The most recent contribution in this area (Williamson et al., 1997) prove that Π_J is difficult to solve even approximately as determining the existence of a schedule with a fixed length of 4 is hard, ie. NP-complete, even in the case that all processing times are one and for any $\rho < 5/4$ there does not exist a polynomial time ρ-approximation algorithm for Π_J unless $P = NP$.

Although the success achieved until now is limited, even with the tremendous excitement generated from the boom period, with the superior technology available and improved understanding of the intractability of Π_J a solution to this problem does not appear to be far away. Encouragement should also be taken from the fact that many hard problems have been insurmountable for decades before finally being solved. A perfect example is Hilbert's 13th problem (Hilbert, 1900) which was eventually solved by Kolmogorov (Kolmogorov, 1957).

3. Benchmark problems

In the previous section we have briefly described various techniques and algorithms which are used to solve Π_J. However to find the comparative merits of these techniques and algorithms they need to be tested on the same problems. Hence the birth of "benchmark problems" which provide a common standard on which all Π_J algorithms can be tested and compared. Hence it will be easy to gauge the strength and power of various algorithms from a statement such as "Algorithm A achieved a makespan of x in time y on benchmark problem B". As the benchmark problems are of different dimensions and grades of difficulty it is simple to determine the capabilities and limitations of a given method by testing it on the benchmark problems. Also the test findings may suggest the improvements required and where they should be made.

These benchmark problems are formulated by various authors (Fisher and Thompson, 1963 (FT); Lawrence, 1984 (LA); Adams et al., 1988 (ABZ); Applegate and Cook, 1991 (ORB); Storer et al., 1992 (SWV); Yamada and Nakano, 1992 (YN); Taillard, 1993 (TA); Demirkol et al., to appear (DMU)) and are freely available. [3] The processing times for these problems are formulated randomly within a given interval (cf. Section 3.2) from a generator constructed by the authors. In order to create the various problems the generator is initiated from different seed values. Hence given the seed values and the interval of the processing times these problems can be generated by anyone. This section describes all the known benchmark problems for Π_J, 242 of them (Beasley, 1990, 1996), highlighting optimal or, in the case of open problems, best solutions and the first author(s) to achieve these bounds. Attention is initially drawn to FT 10, the most famous benchmark instance.

[3] The FT, LA, ABZ, ORB, SWV and YN problems are available from anonymous ftp site ftp://mscmga.ms.ic.ac.uk/pub/jobshop1.txt while the TA problems are available from ftp://mscmga.ms.ic.ac.uk/pub/jobshop2.txt. Both sites are at the Operations Research Library of the Management School, Imperial College, London, UK. The DMU problems are available from Professor Reha Uzsoy at his Electronics Manufacturing Research Group in the Department of Engineering, School of Engineering, Purdue University, Indiana, USA. The URL is http://gilbreth.ecn.purdue.edu/~uzsoy2/benchmark/problems.html.

3.1. FT 10

The benchmark problems which have received the greatest analysis are the instances generated by Fisher and Thompson (Fisher and Thompson, 1963, p. 236): FT 06, 6×6; FT 10, 10×10; FT 20, 20×5. While FT 06 and FT 20 had been solved optimally by 1975, the solution to FT 10 remained elusive until 1987. Florian et al. (1971) indicate that their implementation of the algorithm of Balas (1969) is able to achieve the optimum solution for FT 06. FT 20 however required 12 years to be solved optimally (McMahon and Florian, 1975) while FT 10 emphasises the difficulty involved in solving Π_J.

FT 10, as generated by Fisher and Thompson (1963) is given in Table 4. Table 5 highlights the historical progress of solutions to this instance emphasising the tremendous computational effort directed at the problem. Even though solutions for FT 10 have gradually improved over the years because of the steady progress made by various algorithms, only after 26 years was it proved that the optimal makespan is 930 (Fig. 3) (Carlier and Pinson, 1989). Perregaard and Clausen (1995) indicate that the success of Carlier and Pinson (1989) is realised by the algorithm's ability to add many directed arcs, to the subproblems, between successive branchings. One of the fundamental reasons FT 10 took so long to solve is the large gap, 15%, between the one machine lower bound of 808 and the optimal makespan. Pesch and Tetzlaff (1996) also note that there is one critical edge linking O_{13} and O_{66} which if wrongly orientated will not allow optimum to be achieved. The best makespan that can be achieved when $O_{13} \prec O_{66}$, even if all the other edges are orientated correctly, is 937. Pesch and Tetzlaff (1996) also highlight the importance of this edge by showing that if this disjunction is fixed then the algorithm of Brucker et al. (1994) is able to solve FT 10 within 448 s on a PC 386 while if no edges are orientated the algorithm takes 1138 s. Lawler et al. (1993) report that within 6000 s when applying a deterministic local search to FT 10 more than 9000 local optima have been generated with a best makespan value of 1006, thereby further emphasising the difficulty and hardness of this problem.

3.2. Other benchmark problems

When defining benchmark problems the sequence of machines for each job is randomly generated. The processing time for each operation is also generated randomly and is drawn from a discrete uniform distribution (except for the ORB instances) and the objective in each problem is to minimise the makespan.

FT: 3 problems of 3 different sizes due to Fisher and Thompson (1963): 6×6, 10×10, 20×5. (The name FT has been given by Applegate and Cook (1991)). The processing times are generated from the interval [1, 10] for FT 06 and the interval [1, 99] for FT 10 & FT 20. In the latter two instances in order to simulate a typical machine shop, lower numbered machines are assigned for earlier operations and higher numbered machines for later operations.

LA: 40 problems of 8 different sizes due to Lawrence (1984): 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10, 15×15. (Lawrence (1984) respectively named the instances as F1-5, G1-5, H1-5, A1-5, B1-5, C1-5, D1-5, I1-5. However the name LA as given by Applegate and Cook (1991) is the one more commonly used). The processing times are generated from the interval [5, 99].

ABZ: 5 problems of 2 different sizes due to Adams et al. (1988): 10×10, 20×15. (The name ABZ has been given by Applegate and Cook (1991)). The processing times for ABZ 5, ABZ 6 and ABZ 7–9 are generated from the intervals [50, 100], [25, 100] and [11, 40], respectively.

ORB: 10 problems used by Applegate and Cook (1991): 10×10. (They were formulated in Bonn in 1986 and are characterised as specially created "tougher" problems. Consequently some of these instances are referred to by their creators as doomed, deadlier, etc. The name ORB has been given by Applegate and Cook (1991)).

SWV: 20 problems of 4 different sizes due to Storer et al. (1992): 20×10, 20×15, 50×10, 50×10. (The name SWV has been given by Vaessens (1996)). The processing times are generated from the interval [1,100]. These problems are considered respectively as: Hard, Hard, Hard, Easy and in these problems the set of machines are

Table 4
Processing times and operation orders for FT 10

Operation number	JOB																			
	J_1		J_2		J_3		J_4		J_5		J_6		J_7		J_8		J_9		J_{10}	
	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	τ_i
1–10	1	29	1	43	2	91	2	81	3	14	3	84	2	46	3	31	1	76	2	85
11–20	2	78	3	90	1	85	3	95	1	6	2	2	1	37	1	86	2	69	1	13
21–30	3	9	5	75	4	39	1	71	2	22	6	52	4	61	2	46	4	76	3	61
31–40	4	36	10	11	3	74	5	99	6	61	4	95	3	13	6	74	6	51	7	7
41–50	5	49	4	69	9	90	7	9	4	26	9	48	7	32	5	32	3	85	9	64
51–60	6	11	2	28	6	10	9	52	5	69	10	72	6	21	7	88	10	11	10	76
61–70	7	62	7	46	8	12	8	85	9	21	1	47	10	32	9	19	7	40	6	47
71–80	8	56	6	46	7	89	4	98	8	49	7	65	9	89	10	48	8	89	4	52
81–90	9	44	8	72	10	45	10	22	10	72	5	6	8	30	8	36	5	26	5	90
91–100	10	21	9	30	5	33	6	43	7	53	8	25	5	55	4	79	9	74	8	45

Consider J_3, its first task is on $\mathscr{M}_2$ for 91 time units and is referred to as operation number 3, J_3's next task requires $\mathscr{M}_1$ for 85 time units and is numbered as 13, the next task is on $\mathscr{M}_4$ for 39 units and is numbered 23 and so on until the last task for J_3 is on $\mathscr{M}_5$ requiring 33 units of processing and is categorised as operation number 93.

Table 5
Progress of the FT 10 benchmark instance

Researchers who achieved solution	Number of nodes, iterations, etc. required	Time to achieve, CPU seconds (machine used)		Makespan (UB)	LB
Fisher and Thompson (1963)	Non-BB approach	Not given	(IBM 650)	1101	Not given
Balas (1969) [a]	156	Not given	(CDC 6400)	1177	Not given
Schrage (1970) [a]	923	Not given	(CDC 6400)	1156	Not given
Florian et al. (1971)	100	Not given	(CDC 6400)	1041	Not given
Bratley et al. (1973)	Not given	Not given	(CYBER 74)	980	Not given
McMahon and Florian (1975)	26 692	698.05	(CYBER 74)	972	808
Lageweg et al. (1977)	Not given	Not given	(CDC 73-28)	1082	Not given
Lageweg (1982)	119 344	512	(CYBER 170-750)	935	808
Fisher et al. (1983)	1	700	(CYBER 73-28)	1084	813
Lageweg (1984) [b]	Not given	6420	(CYBER 170-750)	930	907
Barker and McMahon (1985)	164	193	(CYBER 171)	960	860
Adams et al. (1988) (SBII)	270	851	(VAX 780/11)	930	808
Carlier and Pinson (1989)	4039	3305	(PRIME 2655)	930	808

[a] These results are achieved from experiments performed by Florian et al. (1971).
[b] Van Laarhoven et al. (1992) mention that in a private communication with Lageweg (Lageweg, 1984), Lageweg actually solved FT 10 in 1984. His method obtained a makespan of 930 after 1h 47 min of running time of which no improvement was found after 9 h 6 min.

divided into k ($1 \leqslant k \leqslant m$) equally sized subsets. A machine sequence for a job is generated by passing all machines of one set using uniformly distributed assignments before it turns into a machine of the next set. For the easy set $k = 1$ while for the harder instances $k = 2$ (cf. Fisher and Thompson, 1963).

YN: 4 problems due to Yamada and Nakano (1992): 20×20. (The name YN has been given by Vaessens (1996)). The processing times are generated from the interval [10, 50].

TD: 80 problems of 8 different sizes due to Taillard (1993): 15×15, 20×15, 20×20, 30×15, 30×20, 50×15, 50×20, 100×20. (The name TD has been given by Balas and Vazacopoulos, (1995)). The processing times are generated from the interval [1, 99]. Note Taillard has also generated 120 Flow-shop problems and 60 Open-shop problems.

DMU: 80 problems of 8 different sizes due to Demirkol et al. (to appear): 20×15, 20×20, 30×15, 30×20, 40×15, 40×20, 50×15, 50×20. (The name DMU has been given by us. Also given in Table 13 are the names used by the Demirkol et al. to appear) The processing times are generated from the interval [1, 200]. Two sets of problems are constructed. For the first 40 problems (J//$C_{\max}$) $k = 1$ while for the second set of 40 problems (J/2SETS/$C_{\max}$) $k = 2$, hence the name 2SETS (cf. Fisher and Thompson, 1963; Storer et al., 1992). Note Demirkol et al. (to appear) have also generated 320 Job-shop problems with the aim to minimise maximum lateness ($L_{\max}$) as well as 40 Flow-shop problems ($k = m$) with an objective of $C_{\max}$ and 160 Flow-shop problems with an objective of $L_{\max}$.

The authors have run the algorithm of Nowicki and Smutnicki (1996) (NS'96) on the problems of Demirkol et al. (to appear) and the best upper bound achieved from three runs of (NS'96), using the same parameters as Nowicki and Smutnicki (1996), and Demirkol et al. (1997) (DMU'97) is presented. The lower bound achieved by (DMU'97) is compared with the lower bound technique of Taillard (1993) and if, and only if, it is better, credit of the best lower bound is given to (DMU'97). For (DMU'97) the lower bounds are achieved by relaxing the capacity constraints on all but one machine and solving the resulting single machine problem. Upper bounds are the best result achieved from three variations of SBP and five pdrs.

Although not compared here a testsuite of 60 benchmark problems has also been generated by Sadeh (1991). This testsuite consists of 6 groups of 10 problems. Each instance is 10×5 incorporating either one or two bottleneck machines which have

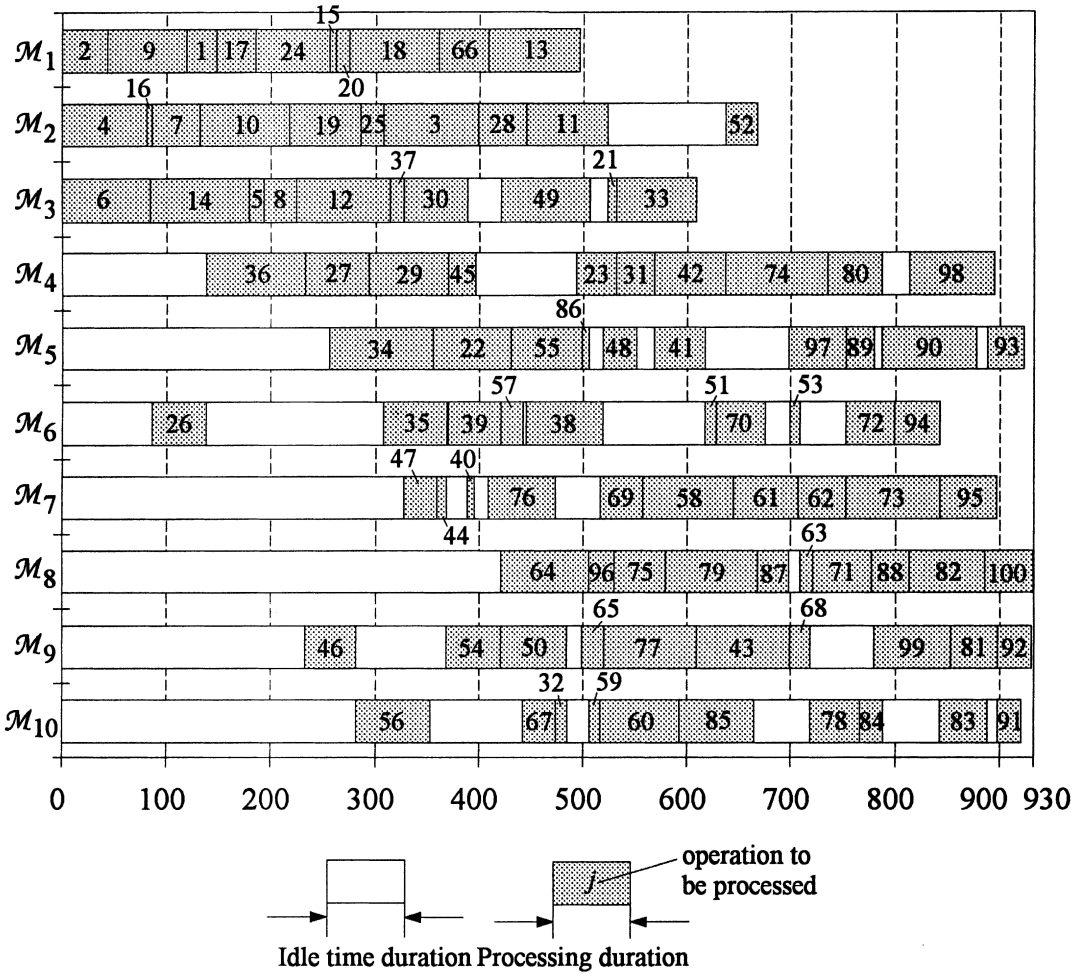

Fig. 3. Gantt Chart showing the optimum solution for FT 10.

been formulated by the spreading of release and due dates and are always processed after the same number of steps.

It should be noted that all of these benchmark problems have only integer processing times with a rather small range. In real production scheduling environments the processing times need not be integers and all be from a given interval. As a result it is felt that benchmark problems have a negative impact in the sense of their true practical

Table 6
Best solutions for the benchmark problems of Fisher and Thompson (1963)

n	*m*	Problem	Optimal makespan	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds
6	6	FT 06	55	41	(CDC 6400) [a]	Balas (1969)
10	10	FT 10	930	6420	(CYBER 170-750)	Lageweg (1984)
20	5	FT 20	1165	151.81	(CYBER 74)	McMahon and Florian (1975)

[a] This result is achieved from the algorithm of Balas (1969) created by Florian et al. (1971). Note 41 does not indicate the CPU processing time in this instance but the number of solutions to reach optimum as no times are documented by Florian et al. (1971).

usefulness. However the analysis of Amar and Gupta (1986), who evaluated the CPU time and the number of iterations of an exact optimising, heuristic and guaranteed performance algorithm on real life and simulated data indicates that real life scheduling problems are easier to solve than simulated ones, regardless of the type of algorithm used.

Tables 6–13 indicate the time to achieve the specified upper bound for the benchmark problems. This is an important distinction and it does not necessarily mean that just because optimality has been attained it has been proved. For example Florian et al. (1971) report the experiment performed on FT 06 using the algorithm of Balas (1969) can reach a makespan of 55 after 41 iterations. However even after 200 iterations the algorithm is unable to prove optimality, therefore the experiment is stopped without the global proof being achieved.

4. Criteria for classification of hard problems

It appears peculiar that problems such as SWV 3, 4 with only 200 operations, TD 3–9 with 225 operations and ABZ 7–9 with 300 operations are still open while instances such as TD 71-80 with 2000 operations were solved optimally within a relatively short period of time after being generated. Such a peculiarity, note Matsuo et al. (1988) and Taillard (1989, 1994), is indicative of a general tendency for Π_J instances to become easier as the ratio of jobs to the number of machines becomes larger (greater than 4 times). Ramudhin and Marier (1996) observe that when $n > m$ the coefficient of variation of the work load increases making it easier to select the bottleneck processor, thus reducing the possibility of becoming trapped in a local minima. This is why solutions for TD 71-80 can be easily attained where ($n = 5m$) and it is also found that if the number of machines is small the instance is further simplified as is highlighted by both Taillard (1994) and Adams et al. (1988). Taillard (1994) is able to provide optimal solutions in polynomial time for problems with 1,000,000 operations as long as no more than 10 machines are used, in other words the ratio of jobs to machines is of the order 100,000: 1 while Adams et al. (1988) solved LA 11-15 (20×5) and LA 31-35 (30×10) using the earliest elegant heuristic method. Further it is worth noting that for many easier problems several local minima equal the global optimum.

However once the size of the problem increases and the instance tends to become square in dimensionality i.e. $n \to m$ it is much harder to solve. Caseau and Laburthe (1995) indicate that for LA 31-35 (30×10) optimality can be easily proved while for LA 21 (15×10) and LA 36-40 (15×15) it requires much effort and YN 1-4 (20×20) cannot be solved. Thus highlighting the difficulties of instances which have a similar number of jobs and machines. In addition as LA 31-35 ($n = 3m$) and TD 61/63-66/68-70 ($n = 2.5m$) are all solved optimally while no global minimum can be achieved for TD 62/67 ($n = 2.5m$) it suggests that an additional criteria for hard problems is that they have more than 10 machines.

Also harder benchmark problems are the ones which satisfy the 2SETS principle, $k = 2$ (Demirkol et al., to appear), e.g. FT 10; SWV 1-15; DMU 41-80. In their SBP analysis Demirkol et al. (1997) note that 2SETS problems require 1–2 orders of magnitude higher computation than the standard problems. In summary a problem can be considered hard if it has the following structure: $N \geqslant 200$ where $n \geqslant 15$, $m \geqslant 10$, $n < 2.5m$. The problem is made more intractable when $k = 2$ and in such a situation n need not be less than or equal to $2.5m$ (cf. SWV 11, 12 and 15). Credibility of this summary comes from the fact that all open problems conform to this structure.

The advent of more powerful computers and better techniques has shifted the difficulty barrier to instances of dimensionality 15. Even though the size of the limiting instances has only increased by 5 machines, from FT 10, the difficulty is highlighted by the fact that 15! is 360,360 times larger than 10!. Many methods which adopt unsophisticated techniques are unable to cope with this factorial increase. For example (Brucker et al., 1994) require to find the position of every preceding and succeeding operation on each critical block before branching. As a result for many large problems the search is terminated before

Table 7
Best solutions for the benchmark problems of Lawrence (1984)

Problem		Optimal makespan	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds [a]
10 Jobs · 5 Machines					
LA 1	(F1)	666	1.26	(VAX 780/11)	Adams et al. (1988)
LA 2	(F2)	655	3.03	(VAX 780/11)	Matsuo et al. (1988)
LA 3	(F3)	597	34.1	(VAX 780/11)	Matsuo et al. (1988)
LA 4	(F4)	590	33.3	(VAX 780/11)	Matsuo et al. (1988)
LA 5	(F5)	593	0.52	(VAX 780/11)	Adams et al. (1988)
15 Jobs · 5 Machines					
LA 6	(G1)	926	1.28	(VAX 780/11)	Adams et al. (1988)
LA 7	(G2)	890	1.51	(VAX 780/11)	Adams et al. (1988)
LA 8	(G3)	863	4.52	(VAX 780/11)	Adams et al. (1988)
LA 9	(G4)	951	0.85	(VAX 780/11)	Adams et al. (1988)
LA 10	(G5)	958	14	(VAX 785)	Van Laarhoven et al. (1992)
20 Jobs · 5 Machines					
LA 11	(H1)	1222	2.03	(VAX 780/11)	Adams et al. (1988)
LA 12	(H2)	1039	0.87	(VAX 780/11)	Adams et al. (1988)
LA 13	(H3)	1150	1.23	(VAX 780/11)	Adams et al. (1988)
LA 14	(H4)	1292	0.94	(VAX 780/11)	Adams et al. (1988)
LA 15	(H5)	1207	3.09	(VAX 780/11)	Adams et al. (1988)
10 Jobs · 10 Machines					
LA 16	(A1)	945	59	(PRIME 2655)	Carlier and Pinson (1990)
LA 17	(A2)	784	94	(VAX 780/11)	Matsuo et al. (1988)
LA 18	(A3)	848	106	(VAX 780/11)	Matsuo et al. (1988)
LA 19	(A4)	842	115	(VAX 780/11)	Matsuo et al. (1988)
LA 20	(A5)	902	667	(VAX 785)	Van Laarhoven et al. (1992)
15 Jobs · 10 Machines					
LA 21	(B1)	1046	87 478.0	(Sparc station ELC) [b]	Applegate and Cook (1991)
LA 22	(B2)	927	183	(VAX 780/11)	Matsuo et al. (1988)
LA 23	(B3)	1032	225	(VAX 780/11)	Adams et al. (1988)
LA 24	(B4)	935	65 422.0	(Sparc station ELC) [b]	Applegate and Cook (1991)
LA 25	(B5)	977	98.2	(Sparc station ELC) [b]	Applegate and Cook (1991)

Table 7 (Continued)

Problem		Optimal make-span	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds [a]
20 Jobs · 10 Machines					
LA 26	(C1)	1218	53	(VAX 780/11)	Matsuo et al. (1988)
LA 27	(C2)	1235	25 307	(IBM RS 6000/320 H)	Carlier and Pinson (1994)
LA 28	(C3)	1216	305	(VAX 780/11)	Matsuo et al. (1988)
LA 29	(C4)	1152	604 800	(Pentimum 90 MHz)	Martin (1996)
LA 30	(C5)	1355	551	(VAX 780/11)	Adams et al. (1988)
30 Jobs · 10 Machines					
LA 31	(D1)	1784	38.3	(VAX 780/11)	Adams et al. (1988)
LA 32	(D2)	1850	29.1	(VAX 780/11)	Adams et al. (1988)
LA 33	(D3)	1719	25.6	(VAX 780/11)	Adams et al. (1988)
LA 34	(D4)	1721	27.6	(VAX 780/11)	Adams et al. (1988)
LA 35	(D5)	1888	21.3	(VAX 780/11)	Adams et al. (1988)
15 Jobs · 15 Machines					
LA 36	(I1)	1268	1303	(PRIME 2655)	Carlier and Pinson (1990)
LA 37	(I2)	1397	1577.0	(Sparc station ELC) [b]	Applegate and Cook (1991)
LA 38	(I3)	1196	165	(AT 386 DX)	Nowicki and Smutnicki (3) (1996)
LA 39	(I4)	1233	6745.0	(Sparc station ELC) [b]	Applegate and Cook (1991)
LA 40	(I5)	1222	150.1	(Sparc station ELC) [b]	Applegate and Cook (1991)

[a] For the instances involving Nowicki and Smutnicki (k) (1996) the k indicates the makespan is achieved after k runs of their algorithm. If no number is given then the algorithm is only run once.

[b] Makespans are achieved by Vaessens in 1994 (Vaessens, 1996) using optimised versions of Bottle (Applegate and Cook, 1991) (for initial solutions), Edge-Finder (Applegate and Cook, 1991) (For lower bounds) and Shuffle (Applegate and Cook, 1991) (for upper bounds).

Table 8
Best solutions for the benchmark problems of Adams et al., 1988 [a]

n	m	Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds [c]	
10	10	ABZ 5	1234	951.5	(SUNSparc 1)	Applegate and Cook (1991)	
		ABZ 6	943	1101	(VAX 780/11)	Adams et al. (1988)	
20	15	ABZ 7	656	65 835	(Pentimum 90 MHz)	Martin (1996)	
		ABZ 8	665 [b] (645)	No time	(Pentimum 90 MHz)	UB – Martin (1996)	LB – Martin (1996)
		ABZ 9	679 [b] (661)	4949	(SUNSparc 330)	UB – SB-*R*GLS*5*	LB – Martin (1996)

[a] The legend for this table is as in the previous tables.
[b] Best upper bound (Best lower bound), this indicates that the instance is still open.
[c] SB – GLS*k* or SB – *R*GLS*k* indicates the result has been achieved by Balas and Vazacopoulos (1998) where *R* denotes that reoptimisation involves a random selection of machines and *k* represents the number of reoptimisations performed.

Table 9
Best solutions for the benchmark problems of Applegate and Cook (1991) [a]

Problem	Optimal makespan	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds
10 Jobs · 10 Machines				
ORB 1	1059	1482.6	(SUNSparc 1)	Applegate and Cook (1991)
ORB 2	888	2484.6	(SUNSparc 1)	Applegate and Cook (1991)
ORB 3	1005	2297.6	(SUNSparc 1)	Applegate and Cook (1991)
ORB 4	1005	1013.3	(SUNSparc 1)	Applegate and Cook (1991)
ORB 5	887	526.0	(SUNSparc 1)	Applegate and Cook (1991)
ORB 6	1010	No time given (Sparc station ELC) [b]		Applegate and Cook (1991)
ORB 7	397	No time given (Sparc station ELC) [b]		Applegate and Cook (1991)
ORB 8	899	No time given (Sparc station ELC) [b]		Applegate and Cook (1991)
ORB 9	934	No time given (Sparc station ELC) [b]		Applegate and Cook (1991)
ORB 10	944	No time given (Sparc station ELC) [b]		Applegate and Cook (1991)

[a] The legend for this table is as the previous tables.
[b] Makespans are achieved by Vaessens in 1994 (Vaessens, 1996) using optimised versions of Bottle (Applegate and Cook, 1991) (for initial solutions), Edge-Finder (Applegate and Cook, 1991) (for lower bounds) and Shuffle (Applegate and Cook, 1991) (for upper bounds).

each position can be checked as the computational requirements exceed the capacity of the machine, resulting in memory overflow and no valid result. Hence FT 20 could not be solved optimally within 3 days (Brucker et al., 1994). The poor performance is also attributed to the fact that such methods make poor initial choices in deciding which search path to follow and therefore get trapped in weak local minima.

Pesch and Tetzlaff (1996) indicate that hard instances are categorised by a substantial difference between the optimal makespan of the subproblem and the makespan of this subproblem in an optimal schedule of the full problem, i.e. a decomposition approach is unable to provide tight lower and upper bounds. These bounds are discussed in detail in the next section. Martin (1996) classifies hard instances into two categories. In the first category the lower bound is very tight or nearly tight to the optimal makespan but achieving the optimal upper bound is very difficult. LA 27 and LA 37 are members of this set. The second group exhibits the opposite characteristic in that obtaining the upper bound is relatively easy however the lower bound is far from the optimal makespan. LA 38 is an instance belonging to this set. It is clearly evident that the attainment of strong bounds is the crux of solving problems as good initial settings can make computation to optimality much faster.

Table 10
Best solutions for the benchmark problems of Storer et al. (1992) [a]

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds	
20 Jobs · 10 Machines					
SWV 1-Hard	1407	No time	(Pentimum 90 MHz)	Martin (1996)	
SWV 2-Hard	1475	No time	(Pentimum 90 MHz)	Martin (1996)	
SWV 3-Hard	1398 [b] (1369)	No time	(or machine) given	UB-Vaessens (1996)	LB – Vaessens (1996)
SWV 4-Hard	1483 [b] (1450)	2433	(SUNSparc 330)	UB-SB-*RGLS10*	LB – Vaessens (1996)
SWV 5-Hard	1424	No time	(Pentimum 90 MHz)	Martin (1996)	
20 Jobs · 15 Machines					
SWV 6-Hard	1678 [b] (1591)	20 957	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
SWV 7-Hard	1620 [b] (1446)	No time	(or machine) given	UB – Vaessens (1996)	LB – Vaessens (1996)
SWV 8-Hard	1763 [b] (1640)	2229	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
SWV 9-Hard	1663 [b] (1604)	No time	(or machine) given	UB – Vaessens (1996)	LB – Vaessens (1996)
SWV 10-Hard	1767 [b] (1631)	23 552	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
50 Jobs · 10 Machines					
SWV 11-Hard	2991 [b] (2983)	7767	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
SWV 12-Hard	3003 [b] (2972)	22 066	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
SWV 13-Hard	3104	14 302	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
SWV 14-Hard	2968	6112	(SUNSparc 330)	SB – *RGLS5*	
SWV 15-Hard	2904 [b] (2885)	30 619	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
SWV 16-Easy	2924	2000 iterations (SunSystem 4/280)		Storer et al. (1992)	
SWV 17-Easy	2794	2000 iterations (SunSystem 4/280)		Storer et al. (1992)	
SWV 18-Easy	2852	2000 iterations (SunSystem 4/280)		Storer et al. (1992)	
SWV 19-Easy	2843	2000 iterations (SunSystem 4/280)		Storer et al. (1992)	
SWV 20-Easy	2823	2000 iterations (SunSystem 4/280)		Storer et al. (1992)	

[a] The legend for this table is as the previous tables.
[b] Best upper bound (Best lower bound), this indicates that the instance is still open.

Table 11
Best solutions for the benchmark problems of Yamada and Nakano (1992) [a]

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds	
20 Jobs · 20 Machines					
YN 1	888 [b] (826)	No time	(or machine) given	UB – Wennink (1995)	LB – Caseau and Laburthe (1995)
YN 2	909 [b] (861)	No time	(or machine) given	UB – Vaessens (1996)	LB – Caseau and Laburthe (1995)
YN 3	893 [b] (827)	670	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)
YN 4	968 [b] (918)	23 032	(AMD-K6/166 MHz)	UB – Thomsen (1997)	LB – Vaessens (1996)

[a] The legend for this table is as the previous tables.
[b] Best upper bound (Best lower bound), this indicates that the instance is still open.

4.1. Upper and lower bounds

The lower bound (LB) is the length of the longest path passing through a given partial sequence. As all operations have not been scheduled then the LB is an estimate of the makespan of a solution containing this partial selection and is the minimum value it can have. As more and more operations are sequenced this value tends towards $C^*_{\max}$ from below. The strongest LBs are currently attained by Martin (1996) and Vaessens (1996) as highlighted in Tables 7–12. Martin (1996) applies

Table 12
Best solutions for the benchmark problems of Taillard (1993) [a]

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds	
15 Jobs · 15 Machines					
TD 1	1231	No time	(or machine) given	Taillard (1994)	
TD 2	1244	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 3	1218 [c] (1206)	1835	(SUNSparc 330)	UB – SB-*R*GLS*10*	LB – Vaessens (1996)
TD 4	1175 [c] (1170)	No time	(or machine) given	UB – Wennink (1995)	LB – Vaessens (1996)
TD 5	1228 [c] (1210)	No time	(or machine) given	UB – Wennink (1995)	LB – Vaessens (1996)
TD 6	1240 [c] (1210)	No time	(or machine) given	UB – Wennink (1995)	LB – Vaessens (1996)
TD 7	1228 [c] (1223)	No time	(or machine) given	UB – Taillard (1994)	LB – Vaessens (1996)
TD 8	1217 [c] (1187)	382	(SUNSparc 330)	UB – SB-GLS*2*	LB – Vaessens (1996)
TD 9	1274 [c] (1247)	646	(SUNSparc 330)	UB – SB-*R*GLS*1*	LB – Vaessens (1996)
TD 10	1241	1306	(SUNSparc 330)	SB-*R*GLS*5*	
20 Jobs · 15 Machines					
TD 11	1364 [c] (1321)	5959	(SUNSparc 330)	UB – SB-*R*GLS*10*	LB – Vaessens (1996)
TD 12	1367 [c] (1321)	6410	(SUNSparc 330)	UB – SB-*R*GLS*15*	LB – Vaessens (1996)
TD 13	1350 [c] (1271)	3169	(SUNSparc 330)	UB – SB-*R*GLS*5*	LB – Vaessens (1996)
TD 14	1345	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 15	1342 [c] (1293)	No time	(or machine) given	UB – Vaessens (1996)	LB – Vaessens (1996)
TD 16	1368 [c] (1300)	No time	(network of 32 machines) [b]	UB – Aarts (1996)	LB – Vaessens (1996)
TD 17	1478 [c] (1458)	3107	(SUNSparc 330)	UB – SB-*R*GLS*10*	LB – Vaessens (1996)
TD 18	1396 [c] (1369)	3239	(SUNSparc 330)	UB – SB-*R*GLS*5*	LB – Vaessens (1996)
TD 19	1341 [c] (1276)	5329	(SUNSparc 330)	UB – SB-*R*GLS*10*	LB – Vaessens (1996)
TD 20	1353 [c] (1316)	No time	(or machine) given	UB – Wennink (1995)	LB – Vaessens (1996)
20 Jobs · 20 Machines					
TD 21	1647 [c] (1539)	No time	(network of 32 machines) [b]	UB – Aarts (1996)	LB – Vaessens (1996)
TD 22	1603 [c] (1511)	10008	(SUNSparc 330)	UB – SB-*R*GLS*15*	LB – Vaessens (1996)
TD 23	1558 [c] (1472)	8286	(SUNSparc 330)	UB – SB-*R*GLS*10*	LB – Vaessens (1996)
TD 24	1651 [c] (1594)	No time	(network of 32 machines) [b]	UB – Aarts (1996)	LB – Vaessens (1996)
TD 25	1598 [c] (1496)	No time	(or machine) given	UB – Taillard (1994)	LB – Vaessens (1996)
TD 26	1655 [c] (1539)	No time	(or machine) given	UB – Wennink (1995)	LB – Vaessens (1996)
TD 27	1689 [c] (1616)	6604	(SUNSparc 330)	UB – SB-*R*GLS*5*	LB – Vaessens (1996)
TD 28	1615 [c] (1591)	6630	(SUNSparc 330)	UB – SB-*R*GLS*5*	LB – Vaessens (1996)
TD 29	1625 [c] (1514)	No time	(network of 32 machines) [b]	UB – Aarts (1996)	LB – Vaessens (1996)
TD 30	1596 [c] (1468)	No time	(or machine) given	UB – Vaessens (1996)	LB – Vaessens (1996)
30 Jobs · 15 Machines					
TD 31	1766 [c] (1764)	No time given	AT 386DX)	UB – Nowicki and Smutnicki (2) (1996)	LB – Taillard (1993)
TD 32	1803 [c] (1774)	6428	(SUNSparc 330)	UB – SB-*R*GLS*5*	LB – Vaessens (1996)
TD 33	1796 [c] (1778)	11 756	(SUNSparc 330)	UB – SB-*R*GLS*10*	LB – Vaessens (1996)
TD 34	1832 [c] (1828)	14 277	(SUNSparc 330)	UB – SB-*R*GLS*18*	LB – Vaessens (1996)
TD 35	2007	No time	(or machine) given	Taillard (1994)	

Table 12 (Continued)

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds	
TD 36	1823 [c] (1819)	11 165	(SUNSparc 330)	UB – SB-*RGLS18*	LB – Vaessens (1996)
TD 37	1784 [c] (1771)	12 233	(SUNSparc 330)	UB – SB-*RGLS18*	LB – Taillard (1993)
TD 38	1681 [c] (1673)	6283	(SUNSparc 330)	UB – SB-*RGLS5*	LB – Taillard (1993)
TD 39	1798 [c] (1795)	3638	(SUNSparc 330)	UB – SB-*RGLS5*	LB – Vaessens (1996)
TD 40	1686 [c] (1631)	16 628	(SUNSparc 330)	UB – SB-*RGLS18*	LB – Vaessens (1996)
30 Jobs · 20 Machines					
TD 41	2026 [c] (1859)	13 198	(SUNSparc 330)	UB – SB-*RGLS5*	LB – Vaessens (1996)
TD 42	1967 [c] (1867)	24 832	(SUNSparc 330)	UB – SB-*RGLS10*	LB – Vaessens (1996)
TD 43	1881 [c] (1809)	12 593	(SUNSparc 330)	UB – SB-*RGLS5*	LB – Vaessens (1996)
TD 44	2004 [c] (1927)	22 045	(SUNSparc 330)	UB – SB-*RGLS10*	LB – Vaessens (1996)
TD 45	2008 [c] (1997)	18 367	(SUNSparc 330)	UB – SB-*RGLS10*	LB – Vaessens (1996)
TD 46	2040 [c] (1940)	23 845	(SUNSparc 330)	UB – SB-*RGLS10*	LB – Taillard (1993)
TD 47	1921 [c] (1789)	5617	(SUNSparc 330)	UB – SB-*RGLS1*	LB – Vaessens (1996)
TD 48	1982 [c] (1912)	14 658	(SUNSparc 330)	UB – SB-*RGLS5*	LB – Vaessens (1996)
TD 49	1994 [c] (1905)	22 632	(SUNSparc 330)	UB – SB-*RGLS10*	LB – Vaessens (1996)
TD 50	1951 [c] (1807)	No time	(network of 32 machines) [b]	UB – Aarts (1996)	LB – Vaessens (1996)
50 Jobs · 15 Machines					
TD 51	2760	No time	(or machine) given	Taillard (1994)	
TD 52	2756	No time	(or machine) given	Taillard (1994)	
TD 53	2717	No time	(or machine) given	Taillard (1994)	
TD 54	2839	No time	(or machine) given	Taillard (1994)	
TD 55	2679	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 56	2781	No time	(or machine) given	Taillard (1994)	
TD 57	2943	No time	(or machine) given	Taillard (1994)	
TD 58	2885	No time	(or machine) given	Taillard (1994)	
TD 59	2655	No time	(or machine) given	Taillard (1994)	
TD 60	2723	No time	(or machine) given	Taillard (1994)	
50 Jobs · 20 Machines					
TD 61	2868	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 62	2900 [c] (2869)	7192	(SUNSparc 330)	UB – SB-GLS*2*	LB – Vaessens (1996)
TD 63	2755	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 64	2702	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 65	2725	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 66	2845	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 67	2826 [c] (2825)	4095	(SUNSparc 330)	UB – SB-GLS*2*	LB – Vaessens (1996)
TD 68	2784	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 69	3071	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	
TD 70	2995	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)	

Table 12 (Continued)

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds
100 Jobs · 20 Machines				
TD 71	5464	No time	(or machine) given	Taillard (1994)
TD 72	5181	No time	(or machine) given	Taillard (1994)
TD 73	5568	No time	(or machine) given	Taillard (1994)
TD 74	5339	No time	(or machine) given	Taillard (1994)
TD 75	5392	No time	(or machine) given	Taillard (1994)
TD 76	5342	No time	(or machine) given	Taillard (1994)
TD 77	5436	No time	(or machine) given	Taillard (1994)
TD 78	5394	No time	(or machine) given	Taillard (1994)
TD 79	5358	No time	(or machine) given	Taillard (1994)
TD 80	5183	No time given	(AT 386 DX)	Nowicki and Smutnicki (2) (1996)

[a] The legend for this table is as the previous tables.
[b] The upper bound is achieved by running a parallel tabu search algorithm on a network of 22 Sparc ELC's, 7 Sparc 5's and 3 Sparc Classic's.
[c] Best upper bound (Best lower bound), this indicates that the instance is still open.

Table 13
Best solutions for the benchmark problems of Demirkol et al. (to appear) [a]

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds	
20 Jobs · 15 Machines ($J//C_{max}$)					
DMU 1 (rcmax_20_15_4)	2607 [c] (2363)	158.43	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 2 (rcmax_20_15_10)	2760 [c] (2452)	222.56	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 3 (rcmax_20_15_5)	2813 [c] (2540)	236.27	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)
DMU 4 (rcmax_20_15_8)	2691 [c] (2486)	193.53	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 5 (rcmax_20_15_1)	2791 [c] (2654)	341.55	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
20 Jobs · 20 Machines ($J//C_{max}$)					
DMU 6 (rcmax_20_20_6)	3316 [c] (2834)	715.37	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 7 (rcmax_20_20_4)	3127 [c] (2677)	539.38	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
DMU 8 (rcmax_20_20_7)	3234 [c] (2901)	301.72	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 9 (rcmax_20_20_8)	3185 [c] (2739)	552.34	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 10 (rcmax_20_20_5)	3022 [c] (2716)	419.42	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
30 Jobs · 15 Machines ($J//C_{max}$)					
DMU 11 (rcmax_30_15_9)	3539 [c] (3395)	260.71	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 12 (rcmax_30_15_10)	3605 [c] (3481)	401.30	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 13 (rcmax_30_15_5)	3725 [c] (3681)	231.92	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 14 (rcmax_30_15_4)	3439 [c] (3394)	111.41	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 15 (rcmax_30_15_1)	3413 [c] (3332)	208.37	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
30 Jobs · 20 Machines ($J//C_{max}$)					
DMU 16 (rcmax_30_20_7)	3882 [c] (3726)	1019.61	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)
DMU 17 (rcmax_30_20_10)	4009 [c] (3697)	544.86	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 18 (rcmax_30_20_9)	3939 [c] (3844)	444.37	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 19 (rcmax_30_20_8)	3967 [c] (3650)	748.75	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
DMU 20 (rcmax_30_20_2)	3833 [c] (3604)	497.48	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
40 Jobs · 15 Machines ($J//C_{max}$)					
DMU 21 (rcmax_40_15_5)	4380	197.18	(DEC Alpha 2000)	NS (3) (1996) [b]	
DMU 22 (rcmax_40_15_9)	4769 [c] (4725)	436.15	(Sun SPARC 1000)	UB – DMU (1997)	LB – DMU (1997)
DMU 23 (rcmax_40_15_10)	4695 [c] (4668)	417.36	(Sun SPARC 1000)	UB – DMU (1997)	LB – DMU (1997)
DMU 24 (rcmax_40_15_8)	4648	69.51	(DEC Alpha 2000)	NS (1) (1996) [b]	
DMU 25 (rcmax_40_15_2)	4168 [c] (4164)	58.63	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
40 Jobs · 20 Machines ($J//C_{max}$)					
DMU 26 (rcmax_40_20_1)	4850 [c] (4647)	601.43	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 27 (rcmax_40_20_3)	4945 [c] (4848)	1462.26	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 28 (rcmax_40_20_6)	4735 [c] (4692)	1299.81	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 29 (rcmax_40_20_2)	4764 [c] (4691)	1451.51	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 30 (rcmax_40_20_7)	4885 [c] (4732)	541.36	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
50 Jobs · 15 Machines ($J//C_{max}$)					
DMU 31 (rcmax_50_15_3)	5695 [c] (5640)	82.06	(DEC Alpha 2000) [b]	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 32 (rcmax_50_15_1)	5927	154.51	(Sun SPARC 1000)	DMU (1997)	
DMU 33 (rcmax_50_15_2)	5728	393.01	(Sun SPARC 1000)	DMU (1997)	
DMU 34 (rcmax_50_15_4)	5385	353.33	(Sun SPARC 1000)	DMU (1997)	
DMU 35 (rcmax_50_15_5)	5635	382.04	(Sun SPARC 1000)	DMU (1997)	
50 Jobs · 20 Machines ($J//C_{max}$)					
DMU 36 (rcmax_50_20_2)	5705 [c] (5621)	2698.97	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 37 (rcmax_50_20_7)	5940 [c] (5851)	422.28	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 38 (rcmax_50_20_6)	5806 [c] (5713)	1218.85	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)

Table 13 (continued)

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds	
DMU 39 (rcmax_50_20_9)	5747	216.51	(DEC Alpha 2000)	NS (1) (1996) [b]	
DMU 40 (rcmax_50_20_3)	5681 [c] (5577)	758.45	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
20 Jobs · 15 Machines ($J/2SETS/C_{max}$)					
DMU 41 (cscmax_20_15_10)	3395 [c] (2839)	248.69	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 42 (cscmax_20_15_5)	3610 [c] (3066)	298.22	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 43 (cscmax_20_15_8)	3647 [c] (3121)	136.94	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 44 (cscmax_20_15_7)	3710 [c] (3112)	258.46	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 45 (cscmax_20_15_1)	3408 [c] (2930)	316.70	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
20 Jobs · 20 Machines ($J/2SETS/C_{max}$)					
DMU 46 (cscmax_20_20_6)	4305 [c] (3425)	507.66	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 47 (cscmax_20_20_4)	4140 [c] (3353)	852.52	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 48 (cscmax_20_20_3)	3994 [c] (3317)	466.30	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
DMU 49 (cscmax_20_20_2)	3932 [c] (3369)	778.27	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 50 (cscmax_20_20_9)	3942 [c] (3379)	497.06	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
30 Jobs · 15 Machines ($J/2SETS/C_{max}$)					
DMU 51 (cscmax_30_15_2)	4416 [c] (3839)	459.01	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
DMU 52 (cscmax_30_15_9)	4659 [c] (4012)	329.05	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 53 (cscmax_30_15_10)	4705 [c] (4108)	281.34	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 54 (cscmax_30_15_5)	4644 [c] (4165)	540.01	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 55 (cscmax_30_15_6)	4633 [c] (4099)	405.27	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
30 Jobs · 20 Machines ($J/2SETS/C_{max}$)					
DMU 56 (cscmax_30_20_9)	5454 [c] (4366)	1288.30	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)
DMU 57 (cscmax_30_20_7)	5187 [c] (4182)	365.72	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 58 (cscmax_30_20_3)	5161 [c] (4214)	1437.94	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 59 (cscmax_30_20_6)	4977 [c] (4199)	914.01	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
DMU 60 (cscmax_30_20_4)	5259 [c] (4259)	880.56	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – DMU (1997)
40 Jobs · 15 Machines ($J/2SETS/C_{max}$)					
DMU 61 (cscmax_40_15_3)	5723 [c] (4886)	534.43	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 62 (cscmax_40_15_6)	5690 [c] (5004)	524.23	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)
DMU 63 (cscmax_40_15_8)	5763 [c] (5049)	590.69	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)
DMU 64 (cscmax_40_15_4)	5891 [c] (5130)	334.14	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
DMU 65 (cscmax_40_15_7)	5514 [c] (5072)	1067.79	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – Taillard (1993)
40 Jobs · 20 Machines ($J/2SETS/C_{max}$)					
DMU 66 (cscmax_40_20_10)	6318 [c] (5357)	1244.20	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 67 (cscmax_40_20_6)	6571 [c] (5484)	1269.40	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 68 (cscmax_40_20_8)	6523 [c] (5423)	2618.23	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)
DMU 69 (cscmax_40_20_5)	6336 [c] (5419)	1293.73	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 70 (cscmax_40_20_9)	6564 [c] (5492)	1164.07	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
50 Jobs · 15 Machines ($J/2SETS/C_{max}$)					
DMU 71 (cscmax_50_15_8)	6824 [c] (6050)	1444.73	(DEC Alpha 2000)	UB – NS (1) (1996) [b]	LB – Taillard (1993)
DMU 72 (cscmax_50_15_6)	6957 [c] (6223)	1383.41	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 73 (cscmax_50_15_10)	6853 [c] (5935)	1642.45	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)
DMU 74 (cscmax_50_15_4)	6820 [c] (6015)	1293.36	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 75 (cscmax_50_15_3)	6909 [c] (6010)	6648.01	(Sun SPARC 1000)	UB – DMU (1997)	LB – DMU (1997)
50 Jobs · 20 Machines ($J/2SETS/C_{max}$)					
DMU 76 (cscmax_50_20_1)	7757 [c] (6329)	2122.95	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)
DMU 77 (cscmax_50_20_4)	7556 [c] (6399)	2051.75	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – DMU (1997)
DMU 78 (cscmax_50_20_3)	7700 [c] (6508)	2892.43	(DEC Alpha 2000)	UB – NS (2) (1996) [b]	LB – DMU (1997)

Table 13 (continued)

Problem	Best bounds	Time to achieve optimum or UB – CPU seconds (machine used)		First group of researchers to arrive at optimum or best bounds	
DMU 79 (cscmax_50_20_7)	7841 [c] (6593)	8636.46	(Sun SPARC 1000)	UB – DMU (1997)	LB – Taillard (1993)
DMU 80 (cscmax_50_20_9)	7453 [c] (6435)	3831.65	(DEC Alpha 2000)	UB – NS (3) (1996) [b]	LB – Taillard (1993)

[a] The legend for this table is as the previous tables.

[b] Results achieved by the authors using the algorithm of Nowicki and Smutnicki (1996) – NS (k) (1996). The k denotes which of the three runs produces the best result.

[c] Best upper bound (Best lower bound), this indicates that the instance is still open.

a technique called shaving where each operation is allocated a time window in which it can be processed and based on various rules and selections one or more time units is attempted to be removed (shaved) off the current makespan, by reducing the time windows of various operations. While Vaessens (1996) combines optimised versions of "Shuffle" and "Bottle" (Applegate and Cook, 1991) with an initial solution derived from one of several techniques (Balas and Vazacopoulos, 1998; Nowicki and Smutnicki, 1996; Yamada and Nakano, 1996a, etc). Although the values for the bounds are very good, both methods are computationally expensive. Therefore the main research challenge is to construct techniques that provide tighter lower bounds but within a reasonable amount of computing time (Applegate and Cook, 1991).

The upper bound (UB) is the makespan of a full sequence and is the maximum value it can have. Vaessens et al. (1996) indicate that in general for most scheduling instances it is easier to obtain a good upper bound and in many algorithms it is the best solution achieved so far where any schedule with a makespan larger than this is disregarded. As improvements are continually made this value tends towards $C^*_{\max}$ from above and at optimum LB = UB. By deriving good bounds many solutions in the search domain can quickly be discarded and hence the search space can be reduced.

In the case of upper bounds the computational analysis (see Section 6) indicates that of the approximation techniques the approaches of Balas and Vazacopoulos (1998), Nowicki and Smutnicki (1996), Yamada and Nakano (1996b, c) are the most effective. Balas and Vazacopoulos (1998) present a variable depth guided local search procedure (GLS), embedded within SBP, that applies an interchange scheme based on a local neighbourhood structure that reverses more than one disjunction at a time. Nowicki and Smutnicki (1996) apply a tabu search approach with critical block neighbourhoods which is combined with an elite solution recovery strategy and Yamada and Nakano (1996b, c) apply a genetic local search technique. Two parent schedules are generated as far apart as possible with respect to their disjunctive graph distance. A biased stochastic selection procedure then selects children which are closer to the second parent with respect to their disjunctive graph distance.

The results of these three methods on the benchmark instances suggest that Balas and Vazacopoulos (1998) have difficulty in dealing with the second category of problems, as defined by Martin (1996), because their method requires substantially more time to solve LA 38 in comparison to LA 27 and LA 37. Although Yamada and Nakano (1996b, c) achieve results for LA 38 faster than LA 27 they are unable to optimally solve LA 38 which indicates that their method also has greater difficulty with the second category of problems. While Nowicki and Smutnicki (1996) find the first category of problems considerably harder as their method is unable to solve either LA 27 or LA 37 optimally. Therefore an effective technique should take aspects from Nowicki and Smutnicki (1996) and combine them with attributes from Yamada and Nakano (1996b, c) and Balas and Vazacopoulos (1998) such that both problem classes can be solved equally well within comparable times. We suggest a good strategy would consist of critical block neighbourhoods and a variable depth search combined with an elite

recovery solution strategy. To allow for a diverse exploration of the solution space, search paths should be created between two elite schedules where the sequences generated could be biased towards one of the elite schedules.

5. Comparisons

Cherkassky et al. (1996) highlight some of the general difficulties encountered in comparing heuristic techniques. They suggest that it is more accurate to discuss comparisons between software implementations which is very much true for the comparisons of Π_J techniques as well. However when making such comparisons it is well known that some machines are obviously more powerful than others and can therefore achieve solutions faster. As a result the Computer-Independent Central Processing Unit (CI-CPU) time has been formulated (Vaessens, 1995), $\text{CI-CPU} = \text{TF} \times \text{CPU}$, which takes into account the capabilities of the machine used. TF is the transformation factor and is based on our interpretation of the performance comparisons made by Dongarra (1998). Some typical TF values are highlighted in Table 14. Nevertheless these factors have to be interpreted with a great deal of care. For fair comparisons Cherkassky et al. (1996) also suggest:

- Careful specification of the goals, methodology and design of experiments.
- The creation of fully or semi automatic methods so that they can be easily applied by non-expert users. The only true way to remove the power of the method from the expertise of the creator is to ensure, when dealing with a problem, that no parameters require modification (fully automatic) or only a few parameters require adjustment by the user (semi-automatic).

Table 14
Some examples of transformation factors

Computer	Transformation factor (TF)
HP 48GX	0.00081
Apollo DN 3000	0.071
CDC CYBER 170-835	0.47
IBM 4381-23	1.3
DEC Station 5000/200	3.7
IBM RISC Sys/6000-930	15
Cray-2/4-256	48
Fujitsu VP2200/10	127
NEC SX-3/1LR	201
Cray C90	479

The main issue is whether methods can be made semi-automatic without compromising their performance. For techniques such as GAS, TS, SA etc this is very difficult. Due to their structure many parameters need to be optimised and Johnson et al. (1988) indicate that the selection of these parameters is inherent to these techniques. Therefore if (semi-) automatic methods are to be encouraged then a generic set of parameters must be created. Cherkassky et al. (1996) note that (semi-) automatism is achieved by relying on (compute-intensive) internal optimisation rather than user expertise to choose proper parameter settings.

In many situations comparisons become invalid as they are not performed from a balanced perspective. This is due to the fact that when a comparative study is given the approach constructed by the author(s) is presented with the optimal choice of parameters, etc. however, in many cases, the other approaches included in the study are not fully optimised. Hence equivalence of testing is not achieved. One such example is seen in the implementation of the SA technique of Van Laarhoven et al., 1992 by Hara (1995) in which he is unable to solve FT 10 & FT 20 due to poor parameterisation. However Van Laarhoven et al., 1992 themselves achieved optimality for both these instances. Therefore in order to avoid this problem, unless successful parametric optimisation can be achieved, the values and times quoted should be taken directly from the article itself.

While we attempt to incorporate many of the suggestions presented by Cherkassky et al. (1996) it is important to note that care should be taken when interpreting these recommendations as they are not completely applicable to Π_J. For example the work of Cherkassky et al. (1996) is directed at the domain of sample comparisons using neural networks. Nevertheless these work are all of considerable interest and very appropriate for all optimisation fields not just this one.

6. Computational comparison

Applegate and Cook (1991) denote the seven problems which they could not solve: LA (21, 27, 29, 38); ABZ (7–9) as computational challenges as they are much harder than FT 10 and until recently their optimal solutions were unknown even though every algorithm had been tried on them. Of these seven instances ABZ 8 and ABZ 9 are still open. Vaessens et al. (1996) also indicate that LA (24, 25, 40) are challenging. As a result Vaessens et al. (1996) include these seven challenging instances of Lawrence as well as the remaining 15×15 problems LA (36, 37, 39), two smaller instances LA (2, 19) and FT 10 when comparing the performance of several methods. Balas and Vazacopoulos (1998) have also applied these 13 instances in their computational study of several techniques. Consequently as these problems provide a suitable comparative test bed Tables 15–17 present, as far as they are available, the performance of several techniques on these instances.

Boyd and Burlingame (1996) also note that the instances quoted by Applegate and Cook (1991) are orders of magnitude harder than those which have been already solved. Consequently Boyd and Burlingame (1996) believe that solving such instances within 24 h will require an algorithm based on a fundamentally new mathematical approach. This further highlights the deficiencies of exact enumerative methods as they require excessive computing effort. On the contrary Vaessens et al. (1996) believe that a good approximation approach should be able to solve all 13 instances within 100 CI-CPU seconds with a total MRE of no more than 2%.

As only limited computational results are available, i.e. solutions do not exist for all 13 instances, mean values are used and comparisons are made with the optimum solution just purely for those instances attempted. The study indicates that hybrid methods involving TS, SBP, GLS and SA techniques are most dominant as these methods obtain solutions that are uniformly as good as or better than the best previously known solutions even for the majority of available benchmark instances. Especially good results with regards to solution quality and time are achieved by the SB-RGLSk technique (Balas and Vazacopoulos, 1998). This method is able to attain an average MRE of 0.05% in 999 CI-CPU seconds, solving 11 out of the 13 problems optimally. The TS algorithm of Nowicki and Smutnicki (1996) is also able to provide good results solving 7 out of the 13 problems optimally. Although it has a slightly higher MRE, 0.20%, than the SBP method it is considerably faster (nearly 10 times). Note the time presented for their algorithm does not include the initial solution. As the initial solution has a complexity of $O(n^3m^5)$, Aarts (1996) indicates that this is somewhat misleading.

The other TS strategies (Dell'Amico and Trubian, 1993; Taillard, 1994; Barnes and Chambers, 1995), although not presented here, are all integrated with several other methods and are able to provide solutions of high quality within reasonable computing times. In comparison the remaining SBP techniques (Adams et al., 1988; Dauzère-Pérès and Lasserre, 1993; Balas et al., 1995) are not competitive as they are not of hybrid formulation. Such conclusions are also made by Ovacik and Uzsoy (1992, 1996) and Holtsclaw and Uzsoy (1996) who show that the performance of SBP clearly improves when incorporated with local search. Without local search SBP is concluded to be no better than pdrs, while requiring much more computing time.

Although SA is not a powerful Π_J technique, hybrid SA approaches are providing strong competition to other methods and the work of Yamada and Nakano (1995a, 1996a) is able to solve 7 of the 10 instances attempted optimally with an average MRE and time of 0.08% and 132,961 CI-CPU seconds respectively. Pure GA implementations are also very poor. The GA method of Bierwirth (1995) which applies a generalised permutation representation is extremely weak only achieving an average MRE of 3.10% in 11,445 CI-CPU seconds. In addition no instance is solved optimally. However the GLS approach of Yamada and Nakano (1996b, c) successfully combining a simple crossover operator into local search has allowed significant advances to be made. Their technique optimally solves 6 out of the 8 instances attempted and achieves an average MRE of 0.17% within 161,977 CI-CPU seconds.

Table 15
Makespan results for a selection of Π_J methods

Author	Type of algorithm	Makespans achieved for the hard benchmark problems												
		FT 10	LA 2	LA 19	LA 21	LA 24	LA 25	LA 27	LA 29	LA 36	LA 37	LA 38	LA 39	LA 40
Perregaard and Clausen (1995)	Branch and bound	930	655	842	1046	935	977	NR	NR	1268	1397	1196	1233	1222
Martin (1996)	Branch and bound	930	655	842	1046	935	977	1235	1152	1268	1397	1196	1233	1222
Jain et al. (1997)	LRM priority rule	1131	806	954	1253	1118	1098	1564	1353	1424	1643	1431	1554	1463
Sabuncuoglu and Bayiz (1997)	Beam search + pdrs	1016	704	882	1154	992	1073	1361	1252	1401	1503	1297	1369	1347
Balas et al. (1995)	Shifting bottleneck	940	667	878	1071	976	1012	1272	1227	1319	1425	1294	1278	1262
Balas and Vazacopoulos (1998)	Variable depth and SBP	930	655	842	1046	935	977	1235	1157	1268	1397	1196	1233	1224
Nuijten and Le Pape (1998)	Constraint satisfaction	930	655	842	1046	938	977	1235	1165	1268	1397	1196	1233	1224
Sabuncuoglu and Gurgun (1996)	Neural networks	940	655	842	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Aarts et al. (1994)	Iterative improvement	1056.6	688.0	899.6	1181.2	1076.0	1126.4	1447.6	1426.4	1456.6	1598.8	1421.8	1445.4	1439.0
Aarts et al. (1994)	Threshold accepting	1003.8	693.8	925.8	1104.2	1014.6	1075.4	1289.8	1262.4	1385.8	1469.2	1323.0	1305.8	1295.0
Lourenço (1993, 1995)	Large step optimisation	937	NA	842	1047	NA	NA	NA	NA	NA	NA	NA	1239	NA
Werner and Winkler (1995)	Reinsertion algorithm	951	655	852	1090	976	1046	1318	1235	1330	1448	1290	1297	1263
Yamada and Nakano (1996a)	Simulated annealing	930	655	842	1046	935	977	1235	1154	NG	NG	1198	NG	1228
Bierwirth (1995)	Genetic algorithm	936	NG	NG	NG	NG	NG	1269	1233	1297	1447	1251	1251	1252
Mattfeld (1996)	Genetic local search	930	655	842	1047	938	977	1236	1180	1269	1402	1201	1240	1228
Yamada and Nakano (1996b, c)	Genetic local search	930	NA	NA	1046	935	977	1235	1166	NA	NA	1196	NA	1224
Nowicki and Smutnicki (1996)	Tabu search	930	655	842	1047 [a]	939 [a]	977 [a]	1236 [a]	1160 [a]	1268 [a]	1407 [a]	1196 [a]	1233 [a]	1229 [a]
Thomsen (1997)	BB + tabu search	930	655	842	1047	935	977	1235	1157	1268	1397	1196	1233	1224
Optimum (Opt)	Any	930	655	842	1046	935	977	1235	1152	1268	1397	1196	1233	1222

FT – Fisher and Thompson (1963) LA – Lawrence (1984)

NA – These instances not attempted; NG – This information is not given; NR – No result is possible due to memory overflow.
[a] Indicates results achieved over 3 runs of the algorithm. Otherwise result achieved from 1 run.

Table 16
CI-CPU times for a selection of Π_J methods

Author	Machine used	CI-CPU times for the hard benchmark problems													
		TF	FT 10	LA 2	LA 19	LA 21	LA 24	LA 25	LA 27	LA 29	LA 36	LA 37	LA 38	LA 39	LA 40
PC (1995)	MEIKO 16-860	21.9	346.02	6.57	232.14	109,809	26,632.6	65,170.0	NR	NR	17016.3	2590.77	86,5092	25,620.8	131,667
M (1996)	Pentium (90 MHz)	1.64	39.36	16.4	32.8	5884.32	3813	8623.12	33,721.7	99,1872	318.16	203.36	217,405	324.72	831.48
JRG (1997)	IBM RISC 6000	19	0.38	0.19	0.38	0.76	0.76	0.76	1.14	1.14	0.95	0.76	0.76	0.76	0.95
SB (1997)	Sparc station classic	1.8	133.2	5.22	14.4	79.2	70.74	77.58	233.64	253.26	177.66	178.56	168.66	172.44	180
BLV (1995)	SUNSparc 330	2.5	27.925	3.575	22	49.7	49.55	56.75	94.75	97.5	139.575	133.2	148.15	126.575	131.025
BV (1998)	SUNSparc 330	2.5	32.25	18.25	175	1529.25	1704.25	559.75	788	389.25	138.5	93.25	3202.75	384.75	3974.25
NLP (1998)	SUN UltraSPARC 1	70	6160	420	2730	30,170	24,010	23,800	64,610	69,370	27,720	13,160	42,630	27,720	48,860
SG (1996)	SUNSparc Station 2	4	27,200	3680	27,200	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
ALLU (1994)	VAX 8650	0.7	69.58	13.02	65.66	170.38	164.36	178.36	344.4	329.7	421.54	445.34	444.92	414.54	417.76
ALLU (1994)	VAX 8650	0.7	69.58	13.02	65.66	170.38	164.36	178.36	344.4	329.7	421.54	445.34	444.92	414.54	417.76
Lo (1993)Lo (1995)	SUNSparc Station 1	1.4	8120	NA	7742	348,600	NA	NA	NA	NA	NA	NA	NA	375,200	NA
WW (1995)	PS/2 - 55 SX	0.15	9.9	1.35	8.7	19.05	33.45	33.45	85.95	85.95	100.95	100.95	100.95	100.95	100.95
YN (1996a)	HP 730	34	18,084.6	NG	NG	12,274	211,684	139,978	265,370	184,756	NG	NG	118,286	NG	113,254
B (1995)	SUNSparc Station 10	23	3105	NG	NG	NG	NG	NG	11,546	11,523	13,179	13,294	13,110	13,041	12,765
Mf (1996)	SUN 10 / 41	23	920	368	667	1495	1288	1196	2484	2392	1817	2185	2116	2047	2277
YN (1996b)YN (1996c)	DEC Alpha 600 5/266	122	10,736	NA	NA	83,899.4	10,5420	93,403.2	288,500	292,800	NA	NA	128,259	NA	292,800
NS (1996)	AT 386 DX	0.48	14.4	3.84	28.8	10.08	88.32	74.4	31.68	236.64	299.04	212.64	79.2	156	154.56
T (1997)	AMD-K6/166 MHz	5.49	115.29	0	5.49	549	10,425.5	181.17	269.01	86,928.7	263.52	197.64	60.39	71.37	384.3

FT – Fisher and Thompson (1963) LA – Lawrence (1984)

NA – These instances not attempted; NG – This information is not given NR; – No result is possible due to memory overflow.

Table 17
Combined CI-CPU and MRE values for these Π_J methods

Author	Makespan							MRE (%) [a]				CI-CPU (s)			
	$\sum$	μ	σ	CV	μ_{Opt}	σ_{Opt}	CV_{Opt}	$\sum$	μ	σ	CV	$\sum$	μ	σ	CV
PC (1995)	11,701	1064	209	0.197	1064	209	0.197	0.00	0.00	0.00	1.000 [b]	1,244,183	113,108	241,766	2.137
M (1996)	14,088	1084	199	0.184	1084	199	0.184	0.00	0.00	0.00	1.000 [b]	1,263,085	97,160.4	264,519	2.722
JRG (1997)	16,792	1292	245	0.190	1084	199	0.184	249.12	19.163	4.46	0.233	9.69	0.75	0.27	0.367
SB (1997)	15,351	1181	225	0.191	1084	199	0.184	114.39	8.799	1.81	0.205	1744.56	134.20	75.40	0.562
BLV (1995)	14,621	1125	210	0.187	1084	199	0.184	48.19	3.707	1.85	0.500	1080.28	83.10	48.44	0.583
BV (1998)	14,095	1084	199	0.184	1084	199	0.184	0.60	0.05	0.12	2.613	12,989.5	999.19	1228.20	1.229
NLP (1998)	14,106	1085	199	0.184	1084	199	0.184	1.61	0.124	0.30	2.452	381,360	29,335.4	21,145.3	0.721
SG (1996)	2437	812	118	0.146	809	115	0.142	1.08	0.36	0.51	1.414	58,080	19,360	11,087	0.573
ALLU (1994)	16,263	1251	258	0.206	1084	199	0.184	193.01	14.85	4.66	0.314	3479.56	267.66	155.29	0.580
ALLU (1994)	15,149	1165	209	0.179	1084	199	0.184	98.94	7.61	2.09	0.275	3479.56	267.66	155.29	0.580
Lo (1993)Lo (1995)	4065	1016	148	0.145	1013	146	0.144	1.33	0.33	0.30	0.908	739,662	184,916	177,234	0.958
WW (1995)	14,751	1135	219	0.193	1084	199	0.184	57.97	4.46	2.31	0.517	782.55	60.20	40.57	0.674
YN (1996a)	10,200	1020	179	0.176	1019	178	0.175	0.83	0.08	0.15	1.823	1,063,687	132,961	82,785	0.623
B (1995)	9936	1242	132	0.106	1204	123	0.102	24.81	3.101	1.87	0.602	91,563	11,445.4	3222.17	0.282
Mf (1996)	14,145	1088	201	0.185	1084	199	0.184	4.84	0.37	0.63	1.681	21,252	1634.77	669.62	0.410
YN (1996b)YN (1996c)	8709	1089	123	0.113	1087	121	0.112	1.38	0.17	0.40	2.308	1,295,817	161,977	105,027	0.648
NS (1996)	14,119	1086	201	0.185	1084	199	0.184	2.59	0.20	0.28	1.398	1389.60	106.89	92.86	0.869
T (1997)	14,096	1084	199	0.184	1084	199	0.184	0.69	0.05	0.12	2.251	99,451.4	7650.10	23,047.0	3.013
Opt	14,088	1084	199	0.184	1084	199	0.184	0.00	0.00	0.00	1.000 [b]	NV	NV	NV	NV

[a] Mean relative error (MRE) = (Makespan achieved (Optimum))/(Optimum) × 100.
$\sum$ = sum; μ = mean; σ = standard deviation; CV = coefficient of variation = σ/μ; NV The result is not viable. μ_{Opt} – This is the mean value of the optimum solution on the instances attempted by the particular algorithm.
σ_{Opt} – This is the standard deviation of the optimum solution on the instances attempted by the particular algorithm.
CV_{Opt} – This is the coefficient of variation of the optimum solution on the instances attempted by the particular algorithm.
[b] As μ and σ tend to 0, CV tends to 1.

Other approximation techniques analysed such as priority dispatch rules, threshold accepting and iterative improvement have produced extremely poor results while some of the relatively newer approaches such as problem-space methods, large step optimisation and neural networks have the potential to be promising techniques. However further analysis is required especially concerning the testing and tuning of parameter values and the application of more problem specific information in order to make these methods more suitable for larger and harder problems. In the case of neural networks several authors conclude it is too early to make an assessment of their application to Π_J. Currently results have only been provided by Sabuncuoglu and Gurgun (1996) which are solely applied to small problems. In addition their method requires excessive effort even for these small instances where many other approaches can solve similar problems in seconds on personal computers.

Although branch and bound approaches are able to produce good solutions, the test bed of problems used in the analysis is just at the limit of the problems which can be solved by these methods. Hence for instances of slightly higher dimensionality the BB algorithm is unable to cope with the size of the search tree generated and invariably memory overflow occurs. Even on this test set Perregaard and Clausen (1995) are unable to solve LA27 and 29 due to these limitations. Martin (1996) is able to cope better with this test set and is able to solve all 13 instances, the only method to do so, in a less average time than Perregaard and Clausen (1995). Thereby suggesting that a time orientation representation merits more consideration. Neverthless for instances of greater dimensionality Martin (1996) invariably suffers with the same problems. In order to avoid such difficulties we suggest incorporating heuristics into the BB methods (cf. Nuijten and Le Pape (1998) and Thomsen (1997). These methods are able to achieve an average MRE of 0.12% and 0.05% in 29,335 and 7,650 CI-CPU seconds, respectively).

In general it is evident that the best results (with respect to time and solution quality) are achieved from hybrid approximation algorithms, as they combine several methods.

7. Hybrid techniques using local search

From a general perspective, the solution to Π_J can be considered as a collection of local decisions concerning which operation to schedule next. Dorndorf and Pesch (1995) suggest that a framework should be constructed which navigates these local decisions through the search domain in order to determine a high quality global solution in a reasonable amount of time. In the framework local decisions made by myopic problem specific heuristics are guided beyond local optimality by an underlying metastrategy. Such hybrid methods are known as meta-heuristics or iterated local search algorithms. They cover a broad spectrum of techniques varying from those which are rather simple, such as the iterative improvement algorithm of Aarts et al. (1994), to others which are extremely elaborate, such as the variable depth guided local search approach of Balas and Vazacopoulos (1998). Vaessens et al. (1996) show that local search methods currently dominate all other techniques and Glover and Greenberg (1989) attribute their success to the fact that they are carefully tailored to take advantage of domain specific requirements.

Π_J meta-heuristics are based on the neighbourhood strategies developed by Grabowski et al. (1986) and Matsuo et al. (1988), Van Laarhoven et al. (1988) and Nowicki and Smutnicki (1996). Each neighbourhood structure permutes critical operations which require processing on the same machine. In essence these strategies differ in which operations in the neighbourhood are swapped and how this exchange is performed. The strength of meta-heuristics comes from the fact that they allow perturbations to non improving schedules and hence are able to transcend local optima. Not surprisingly such methods are considered to be far superior to current BB and AI methodologies (Glover, 1990; Nowicki and Smutnicki, 1996). The prominence of iterated local search techniques is clearly brought out by the works of Laguna et al. (1991, 1993) in which a TS technique is compared with a BB approach. While solution quality is comparable, in some instances the TS technique can achieve results 80 times faster even though the BB approach is implemented on a computer that is

twice as powerful. This time discrepancy is further emphasised as the system of Vaessens (1996) requires 219,570 CI-CPU seconds to solve LA 21 optimally to a makespan of 1046 while Nowicki and Smutnicki require 10.08 CI-CPU seconds to achieve a makespan of 1047. Hence it is evident that the computing requirements for these hybrid local search methods are substantially lower than BB methods.

Dorndorf and Pesch (1995) state that the outcome of an approach depends heavily on the underlying metastrategy. GLS and SA choose moves from a probabilistic learning scheme derived from natural selection and physical science respectively while SBP and TS adopt a deterministic learning method which embraces the aggressive orientation of selecting the best move within the limits of current restrictions (tabu moves for TS and evaluations and guideposts for the variable depth SBP). Hence the computational study of Vaessens et al. (1996) as well as the results documented here suggest TS and SBP provide better strategies to navigate the myopic heuristic through the search domain, thereby generating improved and in general faster solutions. Another reason for the superiority of TS and SBP, implies Glover (1995), is that a bad strategic decision is able to provide more information than a good random choice (such selections are made in SA and GAS). If one uses a strategic decision process rather than a random selection process to arrive at the solution then all the steps taken are known. Hence from a poor strategic decision one can determine the steps at which deficiencies occur and appropriate actions can be taken. However as a random decision provides no indication of the procedure applied to arrive at a particular solution such modifications cannot be made. Although BB and AI methods also use a strategic decision process Glover (1995) believes that, unlike TS which permits responsive exploration, they are too rigid and are therefore unable to provide the required flexibility.

7.1. Limitations of local search

The primary deficiency of local search methods is that they are highly problem dependent requiring multiple runs of the algorithm, in order to obtain meaningful results, and several parameters have to be carefully selected to achieve good solutions. Johnson et al. (1989) suggest that applying these techniques is more of an art as many choices have to be made concerning parameter values which are not trivial and are quite often very poorly made by trial and error. In many situations determination of these values cannot even be realised after several experiments as successful implementation requires much experience.

Many local search methods are also dependent on the initial solutions (Applegate and Cook, 1991; Lourenço, 1993, 1995; Nowicki and Smutnicki, 1996). Poor initialisation can lead to weak final schedules or excessive computing times. In many methods the termination criterion is the number of iterations and as a result these techniques do not stop when optimality is achieved. Hence a futile search of the domain space continues. The suggestion here is a termination criterion should be bilevel: LB and iteration based such that when either one is achieved the algorithm stops.

8. Future work

Although a classification of hard and easy problems has been provided in this paper, scope for future work exists in understanding the reason a 20×10 instance is hard while a 30×10 instance is easy. Also problems of dimensionality $m > n$ and $m \gg n$ need to be investigated because as yet there has been little analysis on instances of this structure.

Local search techniques have made considerable progress in the last 20 years and have quickly reached a certain degree of maturity. Even though extensive computational analysis of these techniques has been performed their overall powers and limitations are less than completely known. There is also no formal method to suggest effective ways of combining these techniques. For example Yamada and Nakano (1995a, 1996a) believe that the most appropriate location for SBP in their SA algorithm is during the acceptance/rejection phase however other researchers might decide that an alternative arrangement is more suitable. Hence

there is a need for looking into how to assemble the components together and also conversely how to dismantle a hybrid model and recombine it in such a way that it becomes a new and more powerful search tool.

Due to the combinatorial nature of Π_J problems current methods have extreme difficulty in effectively searching the solution space for instances of dimensionality greater than 15. To improve existing techniques parallel approaches can be applied to simultaneously perform multiple searches of the neighbourhood. Unfortunately the most recent works applying such approaches have met with limited success (Taillard, 1994; Perregaard and Clausen, 1995; Boyd and Burlingame, 1996). Hence techniques which are more naturally suited to such formulations, such as neural networks should be tried.

We consider that AI methods such as neural networks have yet to show their true potential. Current NN approaches to Π_J (Sabuncuoglu and Gurgun, 1996) are only suitable for small instances and require excessive computing time. Nevertheless if a suitable technique is to be created it will be of hybrid, possibly NN based, construction. While the system developed should be robust emphasis will also be placed on flexibility. By exploiting its inherent parallel distributed processing capability the NN will quickly prune the vast search space and permit an embedded metastrategy to find the optimal schedule in the reduced solution domain.

Even though the above said goals are realisable in the long term there is much work which can be performed in the short term. One area that merits exploration is the creation of local search approaches that consist of three or more search levels. Another area of interest will be combining local search and exact enumerative techniques. This will provide suitable integration of optimisation and approximation methods such that convergence to strong LBS and proof of optimality are achieved from the exact part of the algorithm using a strong UB attained by the heuristic part of the algorithm. Results obtained by Caseau and Laburthe (1995), Thomsen (1997) and Nuijten and Le Pape (1998) indicate their potential. Glover (1994) also suggests the tremendous scope in using both LBS and UBS in order to find good solutions and he refers to this process as "ghost imaging".

It is important to realise that strict adherence to the constraints and boundaries of a problem can prevent the investigation of surrogate approaches which do not precisely correspond to these rules. This is clearly highlighted by the poor performance of many approaches, especially those of an exact nature where extremely tight constraints are specified. However within the domain of TS the technique of strategic oscillation encourages passing through non feasible solutions in order to converge to better minima. The advantage of such an approach is that it executes moves that are less complex than might otherwise be. Moving outside the feasible boundary and returning from different directions also uncovers opportunities for improvement that are not readily attainable when a search is more narrowly confined.

9. Conclusions

Up to the boom period it is evident that research within Π_J has followed distinct phases. Originally much of the work was focused on the creation of heuristics, then the emphasis shifted to exact algorithms and complexity analysis before returning to the study of heuristics however from an enriched perspective. While some attention has returned to the issue of complexity and the derivation of efficient algorithms much of the focus has remained with these elegant heuristics. Hence current research is progressing in both aspects.

This paper has presented a concise overview of job shop scheduling techniques over the last 40 years and has not only led to an improvement in the understanding of Π_J, highlighting the main people, techniques and benchmark instances but it also has detailed the structure of difficult problems. In addition pointers for future work and directions for new applications and research are given where it is suggested that hybrid systems which incorporate metastrategies of the form of TS, SBP as the driving mechanism for NNS and other such AI paradigms, in the development of new applications and improved methodologies, are most suitable. Such models would therefore be

appropriate not just for Π_J but also to solve production scheduling, network planning and other complex combinatorial problems. Although considerable progress has been made in recent years where FT 10 can now be solved in seconds there are still instances of just slightly larger dimensionality which are still open, emphasising the shear intractability of this problem. As a result within the domain of constraint optimisation problems, Π_J remains a prime challenge and one which will stimulate interest well into the next century.

10. For further reading

Nowicki and Smutnicki, 1993; Van Laarhoven and Aarts, 1989

References

Aarts, B.J.M., 1996. A parallel local search algorithm for the job shop scheduling problem. Master's Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

Aarts, E.H.L., Lenstra, J.K. (Eds.), 1997. Local Search in Combinatorial Optimization. Wiley, Chichester, UK.

Aarts, E.H.L., Van Laarhooven, P.J.M., Lenstra, J.K., Ulder, N.L.J., 1994. A computational study of local search algorithms for job-shop scheduling. ORSA Journal on Computing 6 (2), 118–125.

Aarts, E.H.L., Van Laarhooven, P.J.M., Ulder, N.L.J., 1991. Local search based algorithms for job-shop scheduling. Working Paper, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job-shop scheduling. Management Science 34 (3), 391–401.

Akers, S.B., Jr., 1956. A graphical approach to production scheduling problems. Operations Research 4, 244–245.

Akers, S.B., Jr., Friedman, J., 1955. A non numerical approach to scheduling problems. Operations Research 3, 429–442.

Alexander, S.M., 1987. Expert system for the selection of scheduling rules in a job-shop. Computers and Industrial Engineering 12 (3), 167–171.

Alvehus, M., 1997. The shifting bottleneck procedure: A survey. Graduate Course Report, Linköping University, Linköping, Sweden.

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

Applegate, D., Cook, W., 1991. A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3 (2), 149–156.

Ashour, S., 1967. A decomposition approach for the machine scheduling problem. International Journal of Production Research 6 (2), 109–122.

Ashour, S., Hiremath, S.R., 1973. A branch-and-bound approach to the job-shop scheduling problem. International Journal of Production Research 11 (1), 47–58.

Ashour, S., Moore, T.E., Chin, K.-Y., 1974. An implicit enumeration algorithm for nonpreemptive shop scheduling problem. AIIE Transactions 6 (1), 62–72.

Ashour, S., Parker, R.G., 1973. An out-of-kilter approach for machine sequencing problems. Lecture Notes in Economics and Mathematical Systems 86, 206–223.

Balas, E., 1965. An additive algorithm for solving linear programs with zero-one variables. Operations Research 13, 517–546.

Balas, E., 1969. Machine scheduling via disjunctive graphs: An implicit enumeration algorithm. Operations Research 17, 941–957.

Balas, E., 1979. Disjunctive programming. In: Hammer, P.L., Johnson, E.L., Korte, B. (Eds.), Discrete Optimisation II. North-Holland, Amsterdam, pp. 3–52.

Balas, E., 1985. On the facial structure of scheduling polyhedra. Mathematical Programming Study 24, 179–218.

Balas, E., Lancia, G., Serafini, P., Vazacopoulos, A., 1998. Job-shop scheduling with deadlines. Journal of Combinatorial Optimization 1 (4), 324–353.

Balas, E., Lenstra, J.K., Vazacopoulos, A., 1995. The one-machine problem with delayed precedence constraints and its use in job shop scheduling. Management Science 41 (1), 94–109.

Balas, E., Vazacopoulos, A., 1998. Guided local search with shifting bottleneck for job-shop scheduling. Management Science 44 (2), 262–275.

Baptiste, P., Le Pape, C., 1995. A theoretical and experimental comparison of constraint propagation techniques for disjunctive scheduling. In: The Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'14), Montreal, Quebec, Canada.

Baptiste, P., Le Pape, C., Nuijten, W.P.M., 1995. Constraint-based optimization and approximation for job-shop scheduling. In: The Proceedings of the AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems (IJCAI-95), Montreal, Canada.

Barker, J.R., McMahon, G.B., 1985. Scheduling the general job-shop. Management Science 31 (5), 594–598.

Barnes, J.W., Chambers, J.B., 1995. Solving the job shop scheduling problem using tabu search. IIE Transactions 27, 257–263.

Bartusch, M., Möhring, R.H., Radermacher, F.J., 1988. Scheduling project networks with resource constraints and time windows. Ann. Oper. Res. 16, 201–240.

Beasley, J.E., 1990. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 41 (11), 1069–1072.

Beasley, J.E., 1996. Obtaining test problems via internet. Journal of Global Optimization 8 (4), 429–433.

Biegel, J.E., Wink, L.J., 1989. Expert systems can do job-shop scheduling: An exploration and a proposal. Computers and Industrial Engineering 17 (1-4), 347–352.

Bierwirth, C., 1995. A generalized permutation approach to job-shop scheduling with genetic algorithms. OR Spektrum 17 (2-3), 87–92.

Bierwirth, C., Mattfeld, D.C., Kopfer, H., 1996. On permutation representations for scheduling problems. In: Voigt, H.M. et al. (Eds.), PPSN'IV Parallel Problem Solving from Nature, Springer, Berlin, pp. 310–318.

Blackstone, J.H., Jr., Phillips, D.T., Hogg, G.L., 1982. A state of the art survey of dispatching rules for manufacturing job-shop operations. International Journal of Production Research 20, 27–45.

Blazewicz, J., Domschke, W., Pesch, E., 1996. The job-shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93 (1), 1–33.

Blazewicz, J., Dror, M., Weglarz, J., 1991. Mathematical programming formulations for machine scheduling: A survey. European Journal of Operational Research 51 (3), 283–300.

Boyd, E.A., Burlingame, R., 1996. A parallel algorithm for solving difficult job-shop scheduling problems. Operations Research Working Paper, Department of Industrial Engineering, Texas A & M University, College Station, TX, USA.

Bowman, E.H., 1959. The schedule-sequencing problem. Operations Research 7, 621–624.

Bratley, P., Florian, M., Robillard, P., 1973. On sequencing with earliest starts and due dates with application to computing bounds for the ($n/m/G/F_{\max}$) problem. Naval Research Logistics Quarterly 20, 57–67.

Brooks, G.H., White, C.R., 1965. An algorithm for finding optimal or near optimal solutions to the production scheduling problem. Journal of Industrial Engineering 16 (1), 34–40.

Brucker, P., 1988. An efficient algorithm for the job-shop problem with two jobs. Computing 40, 353–359.

Brucker, P., 1994. A polynomial algorithm for the two machine job-shop scheduling problem with a fixed number of jobs. OR Spektrum 16, 5–7.

Brucker, P., Jurisch, B., Sievers, B., 1994. A branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics 49, 109–127.

Brucker, P., Hurink, J., Werner, F., 1996a. Improving local search heuristics for some scheduling problems part I. Discrete Applied Mathematics 65 (1–3), 97–122.

Brucker, P., Kravchenko, S.A., Sotskov, Y.N., 1996a. Preemptive scheduling of the job-shop problems with the fixed number of jobs. Osnabrück. Schrift. Math. Reihe P: Preprints, Heft 184.

Brucker, P., Hurink, J., Werner, F., 1997a. Improving local search heuristics for some scheduling problems part II. Discrete Applied Mathematics 72 (1/2), 47–69.

Brucker, P., Kravchenko, S.A., Sotskov, Y.N., 1997b. On the complexity of two machine job shop scheduling with regular objective functions. OR-Spektrum 19, 5–10.

Carlier, J., 1982. The one-machine sequencing problem. European Journal of Operational Research 11, 42–47.

Carlier, J., Pinson, E., 1989. An algorithm for solving the job shop problem. Management Science 35 (2), 164–176.

Carlier, J., Pinson, E., 1990. A practical use of Jackson's preemptive schedule for solving the job-shop problem. Annals of Operations Research 26, 269–287.

Carlier, J., Pinson, E., 1994. Adjustment of heads and tails for the job-shop problem. European Journal of Operational Research 78 (2), 146–161.

Caseau, Y., Laburthe, F., 1994. Improved CLP scheduling with task intervals. In: Van Hentenryck, P. (Ed.), Proceedings of the 11th International Conference on Logic Programming (ICLP'94), MIT Press, Cambridge, MA.

Caseau, Y., Laburthe, F., 1995. Disjunctive scheduling with task intervals. LIENS Technical Report No 95-25, Laboratoire d'Informatique de l'École Normale Supérieure Département de Mathématiques et d'Informatique, 45 rue d'Ulm, 75230 Paris, France.

Cedimoglu, I.H., 1993. Neural networks in shop floor scheduling. Ph.D. Thesis, School of Industrial and Manufacturing Science, Cranfield University, UK.

Chang, Y., L, Sueyoshi, T., Sullivan, R.S., 1996. Ranking dispatching rules by data envelopment analysis in a job-shop environment. IIE Transactions 28 (8), 631–642.

Charalambous, O., Hindi, K.S., 1991. Review of artificial intelligence based job-shop scheduling systems. Inf. Decis. Technol. 17 (3), 189–202.

Charlton, J.M., Death, C.C., 1970a. A generalized machine scheduling algorithm. Operations Research Quarterly 21 (1), 127–134.

Charlton, J.M., Death, C.C., 1970b. A method of solution for general machine scheduling problems. Operations Research 18 (4), 689–707.

Cheng, R., Gen, M., Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms-I representation. Computers & Industrial Engineering 30 (4), 983–997.

Cheng, C.-C., Smith, S.F., 1997. Applying constraint satisfaction techniques to job shop scheduling. Annals of Operations Research 70, 327–357.

Cherkassky, V., Gehring, D., Mulier, F., 1996. Comparison of adaptive methods for function estimation from samples. IEEE Transactions on Neural Networks 7 (4), 969–984.

Chu, C., Portmann, M.C., Proth, J.M., 1992. A splitting-up approach to simplify job-shop scheduling problems. International Journal of Production Research 30 (4), 859–870.

Clark, W., 1922. The Gantt Chart: A Working Tool of Management, 3rd edition. Ronald, New York.

Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M., 1994. Ant-system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science 34 (1), 39–54.

Committee On the Next Decade of Operations Research (CONDOR), 1988. Operations research: The next decade. Operations Research 36 (4), 619–637.

Conway, R.W., Maxwell, W.L., Miller, L.W., 1967. Theory of Scheduling. Addison–Wesley, Reading, MA.

Cook, S.A., 1971. The complexity of theorem proving procedures. In: The Proceedings of the Third Annual ACM Symposium on the Theory of Computing, Association of Computing Machinery, New York, pp. 151–158.

Crowston, W.B., Glover, F., Thompson, G.L., Trawick, J.D., 1963. Probabilistic and parametric learning combinations of local job-shop scheduling rules. ONR Research Memorandum No. 117, Carnegie Institute of Technology, Pittsburgh, PA, USA.

Dagli, C.H., Lammers, S., Vellanki, M., 1991. Intelligent scheduling in manufacturing using neural networks. Journal of Neural Network Computing Technology Design and Applications 2 (4), 4–10.

Dagli, C.H., Sittisathanchai, S., 1995. Genetic neuro-scheduler – A new approach for job-shop scheduling. International Journal of Production Economics 41 (1-3), 135–145.

Dauzère-Pérès, S., 1995. A procedure for the one machine sequencing problem with dependent jobs. European Journal of Operational Research 81, 579–589.

Dauzère-Pérès, S., Lasserre, J.B., 1993. A modified shifting bottleneck procedure for job-shop scheduling. International Journal of Production Research 31 (4), 923–932.

Davidor, Y., Yamada, T., Nakano R., 1993. The ecological framework II: Improving GA performance at virtually zero cost. In: 55th International Conference on Genetic Algorithms (ICGA), pp. 171–176.

Davis, L., 1985. Job-shop scheduling with genetic algorithm. In: J.J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms and their Applications. Lawrence Erlbaum, Pittsburg, PA, USA, pp. 136–140.

Della Croce, D.F., Tadei, R., Rolando, R., 1994. Solving a real world project scheduling problem with a genetic approach. Belgian Journal of Operations Research, Statistics and Computer Science 33 (1/2), 65–78.

Della Croce, F., Tadei, R., Volta, G., 1995. A genetic algorithm for the job shop problem. Computers and Operations Research 22 (1), 15–24.

Dell'Amico, M., Trubian, M., 1993. Applying tabu search to the job-shop scheduling problem. Annals of Operations Research 41, 231–252.

Demirkol, E., Mehta, S., Uzsoy, R., 1997. A computational study of shifting bottleneck procedures for shop scheduling problems. Journal of Heuristics 3 (2), 111–137.

Demirkol, E., Mehta, S., Uzsoy, R., to appear. Benchmarking for shop scheduling problems. European Journal of Operational Research.

Dongarra, J.J., 1998. Performance of various computers using standard linear equations software. Technical Report CS-89-85, Computer Science Department, University of Tennessee, Knoxville, TN 37996-1301 and Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831, Tennessee, USA.

Dorndorf, U., Pesch, E., 1995. Evolution based learning in a job-shop scheduling environment. Computers and Operations Research 22 (1), 25–40.

Dyer, M.E., Wolsey, L.A., 1990. Formulating the single machine sequencing problem with release dates as a mixed integer program. Discrete Applied Mathematics 26, 255–270.

Erschler, J., Roubellat, F., Vernhes, J.P., 1976. Finding some essential characteristics of the feasible solutions for a scheduling problem (Technical Note). Operations Research 24 (4), 774–783.

Evans, J.R., 1987. Structural analysis of local heuristics in combinatorial optimization. Computers and Operations Research 14 (6), 465–477.

Falkenauer, E., Bouffouix, S., 1991. A genetic algorithm for the job-shop. In: The Proceedings of the IEEE International Conference on Robotics and Automation, Sacremento, California, USA.

Fisher, M.L., 1973a. Optimal solution of scheduling problems using Lagrange multipliers: Part I. Operations Research 21, 1114–1127.

Fisher, M.L., 1973b. Optimal solution of scheduling problems using Lagrange multipliers: Part II. In: Symposium on the Theory of Scheduling and its Applications. Springer, Berlin.

Fisher, M.L., 1976. A dual algorithm for the one-machine scheduling problem. Math. Programming 11, 229–251.

Fisher, M.L., Lageweg, B.J., Lenstra, J.K., Rinnooy Kan, A.H.G., 1983. Surrogate duality relaxation for job-shop scheduling. Discrete Applied Mathematics 5 (1), 65–75.

Fisher, M.L., Northup, W.D., Shapiro, J.F., 1975. Using duality to solve discrete optimization problems: Theory and computational experience. Math. Programming Stud. 3, 56–94.

Fisher, M.L., Rinnooy Kan, A.H.G., (Eds.), 1988. The design, analysis and implementation of heuristics (special issue). Management Science 34 (3), 263–401.

Fisher, H., Thompson, G.L., 1963. Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J.F., Thompson, G.L. (Eds.), Industrial Scheduling. Prentice-Hall, Englewood Cliffs, NJ, pp. 225–251.

Florian, M., Trépant, P., McMahon, G., 1971. An implicit enumeration algorithm for the machine sequencing problem. Management Science Application Series 17 (12), B782–B792.

Foo, S.Y., Takefuji, Y., 1988a. Stochastic neural networks for solving job-shop scheduling: Part 1. Problem representation. In: Kosko, B. (Ed.), IEEE International Conference on Neural Networks, San Diego, CA, USA, pp. 275–282.

Foo, S.Y., Takefuji, Y., 1988b. Stochastic neural networks for solving job-shop scheduling: Part 2. Architecture and simulations. In: Kosko, B. (Ed.), IEEE International Conference on Neural Networks, San Diego, CA, pp. 283–290.

Foo, S.Y., Takefuji, Y., 1988c. Integer linear programming neural networks for job-shop scheduling. In: Kosko, B. (Ed.), IEEE International Conference on Neural Networks, San Diego, CA, pp. 341–348.

Foo, S.Y., Takefuji, Y., Szu, H., 1994. Job-shop scheduling based on modified Tank-Hopfield linear programming networks. Eng. Appl. Artificial Intelligence 7 (3), 321–327.

Foo, S.Y., Takefuji, Y., Szu, H., 1995. Scaling properties of neural networks for job-shop scheduling. Neurocomputing 8 (1), 79–91.

Fox, M.S., 1987. Constraint – Directed Search: A Case Study of Job-Shop Scheduling, Research Notes in Artificial Intelligence. Pitman, London.

Gantt, H.L., 1919. Efficiency and democracy. Trans. ASME 40, 799–808.

Garey, M.R., Johnson, D.S., 1979. Computers and Intertractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flow shop and job-shop scheduling. Mathematics of Operations Research 1 (2), 117–129.

Gere, W.S., Jr., 1966. Heuristics in job-shop scheduling. Management Science 13, 167–190.

Giffler, B., Thompson, G.L., 1960. Algorithms for solving production scheduling problems. Operations Research 8 (4), 487–503.

Glover, F., 1968. Surrogate constraints. Operations Research 16, 741–749.

Glover, F., 1975. Surrogate constraint duality in mathematical programming. Operations Research 23, 434–451.

Glover, F., 1977. Heuristics for integer programming using surrogate constraints. Decision Sciences 8 (1), 156–166.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13 (5), 533–549.

Glover, F., 1989. Tabu search – Part I. ORSA Journal on Computing 1 (3), 190–206.

Glover, F., 1990. Tabu search – Part II. ORSA Journal on Computing 2 (1), 4–32.

Glover, F., 1994. Optimization by ghost image processes in neural networks. Computers and Operations Research 21 (8), 801–822.

Glover, F., 1995. Tabu search fundamentals and uses. Working Paper, College of Business and Administration and Graduate School of Business Administration, University of Colorado, Boulder, Colorado, USA.

Glover, F., Greenberg, H.J., 1989. New approaches for heuristic search: A bilateral linkage with artificial intelligence. European Journal of Operations Research 39 (2), 119–130.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer, Norwell, MA.

Gonzalez, T., Sahni, S., 1978. Flowshop and jobshop schedules: Complexity and approximation. Operations Research 20, 36–52.

Grabot, B., Geneste, L., 1994. Dispatching rules in scheduling: A fuzzy approach. International Journal of Production Research 32 (4), 903–915.

Grabowski, J., Nowicki, E., Smutnicki, C., 1988. Block algorithm for scheduling operations in a job-shop system (in Polish). Przeglad Statystyczny XXXV (1), 67–80.

Grabowski, J., Nowicki, E., Zdrzalka, S., 1986. A block approach for single machine scheduling with release dates and due dates. European Journal of Operational Research 26, 278–285.

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

Greenberg, H., 1968. A branch and bound solution to the general scheduling problem. Operations Research 16 (2), 353–361.

Grefenstette, J.J., 1987. Incorporating problem specific knowledge into genetic algorithms. In: Davis, L. (Ed.), Genetic Algorithms and Simulated Annealing. Pitman, London, pp. 42–60.

Hara, H., 1995. Job-shop scheduling by minimal conflict search. Japanese Artificial Intelligence Society 10 (1), 88–93.

Hardgrave, W.H., Nemhauser, G.L., 1963. A geometric model and graphical algorithm for a sequencing problem. Operations Research 11, 889–900.

Harvey, W.D., 1995. Nonsystematic backtracking search. Ph.D. Thesis, Department of Computer Science, Stanford University, CA.

Harvey, W.D., Ginsberg, M.L., 1995. Limited discrepancy search. In: International Joint Conference on Artificial Intelligence (IJCAI'95), Montréal, Québec, Canada.

Haupt, R., 1989. A survey of priority rule based scheduling. OR Spektrum 11, 3–16.

Hefetz, N., Adiri, I., 1982. An efficient optimal algorithm for the two-machines unit-time job-shop schedule-length problem. Mathematics of Operations Research 7, 354–360.

Hilbert, D., 1900. Mathematische probleme. Nachrichten der Akademie der Wissenschaften Göttingen, pp. 290–329.

Hoitomt, D.J., Luh, P.B., Pattipati, K.R., 1993. Practical approach to job-shop scheduling problems. IEEE Trans. Rob. Autom. 9 (1), 1–13.

Holtsclaw, H.H., Uzsoy, R., 1996. Machine criticality measures and subproblem solution procedures in shifting bottleneck methods: A computational study. Journal of the Operational Research Society 47, 666–677.

Hoogeveen, J.A., Van De Velde, S.L., 1995. Stronger Lagrangian bounds by the use of slack variables: Applications to machine scheduling problems. Mathematical Programming 70 (2), 173–190.

Ivens, P., Lambrecht, M., 1996. Extending the shifting bottleneck procedure to real-life applications. European Journal of Operational Research 90, 252–268.

Jackson, J.R., 1955. Scheduling a production line to minimise maximum tardiness. Research Report 43, Management Science Research Projects, University of California, Los Angeles, USA.

Jackson, J.R., 1956. An extension of Johnson's result on job lot scheduling. Naval Research Logistics Quarterly 3 (3), 201–203.

Jain, A.S., Meeran, S., 1998. Job-shop scheduling using neural networks. International Journal of Production Research 36 (5), 1249–1272.

Jain, A.S., Rangaswamy, B., Glover, F., 1997. New and "stronger" job-shop neighbourhoods: Are they as good as they seem?. Technical Report, Graduate School of Business and Administration, University of Colorado, Boulder, CO, USA.

Jeremiah, B., Lalchandani, A., Schrage, L., 1964. Heuristic rules toward optimal scheduling. Research Report, Department of Industrial Engineering, Cornell University.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with set-up times included. Naval Research Logistics Quarterly 1, 61–68.

Johnson, D.S., Papadimitriou, C.H., Yannakakis, M., 1988. How easy is local search? Journal of Computer and System Sciences 37 (1), 79–100.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., 1989. Optimization by simulated annealing: An experimental evaluation. Part I: Graph partitioning. Operations Research 37 (6), 865–892.

Kanzedal, S.A., 1983. A decomposition approach to solve large scale scheduling problems (in Russian). Avtomat. i Telemekh. 10, 144–151.

Kim, S.Y., Lee, Y.H., Agnihotri, D., 1995. A hybrid approach for sequencing jobs using heuristic rules and neural networks. Production Planning and Control 6 (5), 445–454.

Kobayashi, S., Ono, I., Yamamura, M., 1995. An efficient genetic algorithm for job shop scheduling problems. In: Proceedings of the Sixth International Conference on Genetic Algorithms. Morgan Kaufmann, San Francisco, CA, pp. 506–511.

Kolmogorov, A.N., 1957. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition (in Russian, American Mathematical Society Translation 28 (1963) 55–59). Doklady Akademiia Nauk SSR 14 (5), 953–956.

Kolonko, M., to appear. Some new results on simulated annealing applied to the job shop scheduling problem. The European Journal of Operational Research.

Kravchenko, S.A., 1995. Optimal scheduling for two-machine unit-time job-shop problems (in Russian). Institute of Engineering Cybernetics of the Academy of Sciences of Belarus, Preprint 7, Minsk, Belarus.

Kravchenko, S.A., Sotskov, Y.N., 1996. Optimal makespan schedule for three jobs on two machines. Z. Oper. Res. 43 (2), 233–238.

Krüger, K., Shakhlevich, N.V., Sotskov, Y.N., Werner, F., 1995. A heuristic decomposition algorithm for scheduling problems on mixed graphs. Journal of the Operational Research Society 46, 1481–1497.

Kubiak, W., Timkovsky, V.G., 1996. Total completion time minimization in two-machine job shops with unit time operations. European Journal of Operational Research 94 (2), 310–320.

Kusiak, A., Chen, M., 1988. Expert systems for planning and scheduling manufacturing systems. European Journal of Operational Research 34, 113–130.

Lageweg, B.J., 1982. Combinatorial planning models. Ph.D. Thesis, Mathematisch Centrum, Amsterdam, The Netherlands.

Lageweg, B.J., 1984. Private Communication with Van Laarhooven, P.J.M.; Aarts, E.H.L., and Lenstra, J.K., discussing the achievement of a makespan of 930 for FT 10.

Lageweg, B.J., Lenstra, K., Rinnooy Kan, A.H.G., 1977. Job-shop scheduling by implicit enumeration. Management Science 24 (4), 441–450.

Laguna, M., Barnes, J.W., Glover, F.W., 1991. Tabu search methods for a single machine scheduling problem. Journal of Intelligent Manufacturing 2, 63–74.

Laguna, M., Barnes, J.W., Glover, F.W., 1993. Intelligent scheduling with tabu search: An application to jobs with linear delay penalties and sequence-dependent setup costs and times. Journal of Applied Intelligence 3, 159–172.

Lawler, E.L., 1983. Recent results in the theory of machine scheduling. In: Bachem, A., Grötschel, M., Korte, B. (Eds.), Mathematical Programming: The State of the Art. Springer, Berlin, pp. 202–234.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1982. Recent developments in deterministic sequencing and scheduling: A survey. In: Dempster, M.A.H. et al. (Eds.), Deterministic and Stochastic Scheduling. Reidel, Dordrecht, pp. 35–73.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., 1993. Sequencing and scheduling: Algorithms and complexity. In: Handbook in Operations Research and Management Science 4: Logistics of Production and Inventory.

Lawrence, S., 1984. Supplement to resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

Lenstra, J.K., 1976. Sequencing by enumerative methods. Ph.D. Thesis, Mathematisch Centrum Tract 69, Amsterdam, The Netherlands.

Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Computational complexity of discrete optimization problems. Annals of Discrete Mathematics 4, 121–140.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P., 1977. Complexity of machine scheduling problems. Ann. Discrete Math. 7, 343–362.

Lo, Z.-P., Bavarian, B., 1993. Multiple job-shop scheduling with artificial neural networks. Comput. Electr. Eng. 19 (2), 87–101.

Lourenço, H.R.D., 1993. A computational study of the job-shop and the flow-shop scheduling problems. Ph.D. Thesis TR – 1060, School of Operations Research and Industrial Engineering, Cornell University, NY.

Lourenço, H.R.D., 1995. Job-shop scheduling: Computational study of local search and large-step optimization methods. European Journal of Operational Research 83, 347–364.

Lourenço, H.R.D., Zwijnenburg, M., 1996. Combining the large-step optimization with tabu-search: Application to the job-shop scheduling problem. In: Osman, I.H., Kelly, J.P.

(Eds.), Meta-heuristics: Theory and Applications. Kluwer Academic Publishers, Boston, MA, pp. 219–236.

MacCarthy, B.L., Liu, J., 1993. Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. International Journal of Production Research 31 (1), 59–79.

Martin, P.D., 1996. A time-oriented approach to computing optimal schedules for the job-shop scheduling problem. Ph.D. Thesis, School of Operations Research and Industrial Engineering, Cornell University, NY.

Martin, O., Otto, S.W., Felten, E.W., 1989. Large-step Markov chains for traveling salesman problem. Complex Systems 5, 299–326.

Martin, O., Otto, S.W., Felten, E.W., 1992. Large-step Markov chains for TSP incorporating local search heuristics. Operations Research Letters 11, 219–224.

Matsuo, H., Suh, C.J., Sullivan, R.S., 1988. A controlled search simulated annealing method for the general job-shop scheduling problem. Working Paper, 03-04-88, Graduate School of Business, University of Texas at Austin, Austin, Texas, USA.

Mattfeld, D.C., 1996. Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling. Physica, Heidelberg.

Mattfeld, D.C., Bierwirth, C., Kopfer, H., to appear. A search space analysis of the job shop scheduling problems. Annals of Operations Research.

Mattfeld, D.C., Kopfer, H., Bierwirth, C., 1994. Control of parallel population dynamics by social-like behaviour of GA-individuals. In: Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN3). Springer, Berlin, pp. 15–25.

McMahon, G.B., Florian, M., 1975. On scheduling with ready times and due dates to minimize maximum lateness. Operations Research 23 (3), 475–482.

Moore, L.J., 1968. An n-job, one-machine sequence algorithm for minimizing the number of late jobs. Management Science 15, 102–109.

Morton, T.E., 1990. Shifting bottleneck methods in job shop and project scheduling: Tutorial and research directions. Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA, USA.

Morton, T.E., Pentico, D.W., 1993. Heuristic Scheduling Systems, Wiley Series in Engineering and Technology Management. Wiley, New York.

Moscato, P., 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. C3P Report 826, Caltech Concurrent Computation Program, Caltech, California, USA.

Nabeshima, I., 1971. General scheduling algorithms with applications to parallel scheduling and multiprogramming scheduling. Journal of Operations Research Society of Japan 14 (2), 72–99.

Nakano, R., Yamada, T., 1991. Conventional genetic algorithm for job-shop problems. In: Kenneth, M.K., Booker, L.B. (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications, San Diego, California, USA, pp. 474–479.

Norman, B., Bean, J., 1995. Random keys genetic algorithm for job-shop scheduling: Unabridged version. Technical Report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, USA.

Nowicki, E., Smutnicki, C., 1993. A fast taboo search algorithm for the job-shop problem. Preprint No. 8/93, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw.

Nowicki, E., Smutnicki, C., 1996. A fast taboo search algorithm for the job-shop problem. Management Science 42 (6), 797–813.

Nuijten, W.P.M., Aarts, E.H.L., 1994. Constraint satisfaction for multiple capacitated job shop scheduling. In: Cohn, A.G. (Ed.), Proceedings of the 11th International Conference on Artificial Intelligence (ECAI'94), pp. 635–639.

Nuijten, W.P.M., Aarts, E.H.L., 1996. A computational study of constraint satisfaction for multiple capacitated job-shop scheduling. European Journal of Operations Research 90 (2), 269–284.

Nuijten, W.P.M., Le Pape, C., 1998. Constraint-based job-shop scheduling with ILOG SCHEDULER. Journal of Heuristics 3 (4), 271–286.

Ovacik, I.M., Uzsoy, R., 1992. A shifting bottleneck algorithm for scheduling semiconductor testing operations. Journal of Electronics Manufacturing 2, 119–134.

Ovacik, I.M., Uzsoy, R., 1996. Decomposition methods for scheduling semiconductor testing facilities. International Journal of Flexible Manufacturing Systems 8, 357–388.

Panwalkar, S.S., Iskander, W., 1977. A survey of scheduling rules. Operations Research 25 (1), 45–61.

Parker, R.G., 1995. Deterministic Scheduling. Chapman and Hall, London.

Perregaard, M., Clausen, J., 1995. Parallel branch-and-bound methods for the job-shop scheduling problem. Working Paper, University of Copenhagen, Copenhagen.

Pesch, E., 1993. Machine Learning by Schedule Decomposition. Working Paper, Faculty of Economics and Business Administration, University of Limburg, Maastricht, The Netherlands.

Pesch, E., Tetzlaff, U.A.W., 1996. Constraint propagation based scheduling of job shops. INFORMS Journal on Computing 8 (2), 144–157.

Pinson, E., 1995. The job-shop scheduling problem: A concise survey and some recent developments. In: Chrétienne, P., Coffman, Jr., E.G., Lenstra, J.K., Liu, Z. (Eds.), Scheduling Theory and its Applications. Wiley, New York, pp. 277–293.

Porter, D.B., 1968. The Gantt chart as applied to production scheduling and control. Naval Research Logistics Quarterly 15, 311–317.

Radermacher, F.J., 1985. Scheduling of project networks. Annals of Operations Research 4 (6), 227–252.

Ramudhin, A., Marier, P., 1996. The generalised shifting bottleneck procedure. European Journal of Operational Research 93 (1), 34–48.

Resende, M.G.C., 1997. A GRASP for job shop scheduling. INFORMS Spring Meeting, San Diego, CA.

Rinnooy Kan, A.H.G., 1976. Machine scheduling problems: Classification, complexity and computations. In: Stenfert Kroese, H.E., Leiden, B.V. (Eds.), Martinus Nijhoff, The Hague, The Netherlands.

Rodammer, F.A., White, K.P., Jr., 1988. A recent survey of production scheduling. IEEE Transactions of Systems, Man and Cybernetics 18 (6), 841–851.

Ross, P., Fang, H.L., Corne, D., 1993. A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problems. In: Fifth International Conference on Genetic Algorithms (ICGA'5), pp. 375–382.

Roy, B., Sussmann, B., 1964. Les problèmes d'ordonnancement avec contraintes disjonctives. Note D.S. no. 9 bis, SEMA, Paris, France.

Sabuncuoglu, I., Bayiz, M., 1997. A beam search based algorithm for the job shop scheduling problem. Research Report IEOR-9705, Department of Industrial Engineering, Bilkent University, Turkey.

Sabuncuoglu, I., Gurgun, B., 1996. A neural network model for scheduling problems. European Journal of Operations Research 93 (2), 288–299.

Sadeh, N., 1991. Look-ahead techniques for micro-opportunistic job shop scheduling. Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA.

Sadeh, N., Fox, M.S., 1996. Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. Artificial Intelligence 86 (1), 1–41.

Sadeh, N., Nakakuki, Y., 1996. Focused simulated annealing search – an application to job-shop scheduling. Annals of Operations Research 63, 77–103.

Sadeh, N., Sycara, K., Xiong, Y.L., 1995. Backtracking techniques for the job-shop scheduling constraint satisfaction problem. Artificial Intelligence 76 (1/2), 455–480.

Salvesen, M.E., Anderson, S.L., 1951. Some notes on the problem of programming industrial production. George Washington University Logistics papers.

Satake, T., Morikawa, K., Nakamura, N., 1994. Neural Network Approach for Minimizing the Makespan of the General Job-Shop. International Journal of Production Economics 33 (1-3), 67–74.

Schrage, L., 1970. Solving resource-constrained network problems by implicit enumeration: Non pre-emptive case. Operations Research 18, 263–278.

Shakhlevich, N.V., Sotskov, Y.N., Werner, F., 1996. An adaptive approach in production scheduling based on the mixed graph model. IEE Proc.-Control Theory Appl. 143 (1), 9–16.

Shi, G., 1997. A genetic algorithm applied to a classic job-shop scheduling problem. International Journal of Systems Science 28 (1), 25–32.

Shmoys, D.B., Stein, C., Wein, J., 1994. Improved approximation algorithms for shop scheduling problems. SIAM Journal on Computing 23 (3), 617–632.

Sim, S.K., Yeo, K.T., Lee, W.H., 1994. An expert neural network system for dynamic job-shop scheduling. International Journal of Production Research 32 (8), 1759–1773.

Smith, W.E., 1956. Various optimizers for single stage production. Naval Research Logistics Quarterly 3, 59–66.

Sotskov, Y.N., 1985. Optimal scheduling two jobs with regular criterion (in Russian). In: Design Processes Automating. Institute of Engineering Cybernetics of the Academy of Sciences of Belarus, Minsk, Belarus, pp. 86–95.

Sotskov, Y.N., 1991. The complexity of shop-scheduling problems with two or three jobs. European Journal of Operational Research 53, 326–336.

Sotskov, Y.N., 1996. Software for production scheduling based on the mixed (multi)graph approach. Computing and Control Engineering Journal 7 (5), 240–246.

Sotskov, Y.N., 1997. Mixed multi-graph approach to scheduling jobs on machines of different types. Optimization, pp. 1–36.

Sotskov, Y.N., Shakhlevich, N.V., 1995. NP-hardness of shop scheduling problems with three jobs. Discrete Appl. Math. 59 (3), 237–266.

Storer, R.H., Wu, S.D., Vaccari, R., 1992. New search spaces for sequencing problems with applications to job-shop scheduling. Management Science 38 (10), 1495–1509.

Storer, R.H., Wu, S.D., Vaccari, R., 1995. Problem and heuristic space search strategies for job shop scheduling. ORSA Journal on Computing 7 (4), 453–467.

Sun, D.K., Batta, R., Lin, L., 1995. Effective job-shop scheduling through active chain manipulation. Computers and Operations Research 22 (2), 159–172.

Sussmann, B., 1972. Scheduling problems with interval disjunctions. Z. Oper. Res. 16, 165–178.

Szwarc, W., 1960. Solution of the Akers–Friedman scheduling problem. Operations Research 8, 782–788.

Taillard, É., 1989. Parallel taboo search technique for the job-shop scheduling problem. Internal Research Report ORWP89/11, Département de Mathématiques (DMA), École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland.

Taillard, , É.1993. Benchmarks for basic scheduling problems. European Journal of Operational Research 64 (2), 278–285.

Taillard, , É.1994. Parallel taboo search techniques for the job-shop scheduling problem. ORSA Journal on Computing 16 (2), 108–117.

Tamaki, H., Nishikawa, Y., 1992. A paralleled genetic algorithm based on a neighbourhood model and its application to the job-shop scheduling. In: Männer, R., Manderick, B. (Eds.), Proceedings of the Second International Workshop on Parallel Problem Solving from Nature (PPSN'2), Brussels, Belgium, pp. 573–582.

Thomsen, S., 1997. Metaheuristics combined with branch and bound (in Danish). Technical Report, Copenhagen Business School, Copenhagen, Denmark.

Timkovsky, V.G., 1995. The complexity of unit time job shop scheduling problems. Working Paper, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada.

Ulder, N.L.J., Aarts, E.H.L., Bandelt, H.-J., Van Laarhooven, P.J.M., Pesch, E., 1990. Improving TSP exchange heuristics by population genetics. First International Workshop on Parallel Problem solving from Nature (PPSN'1), Dortmund, Germany.

Ulder, N.L.J., Aarts, E.H.L., Bandelt, H.-J., Van Laarhooven, P.J.M., Pesch, E., 1991. Genetic local search algorithm for the travelling salesman problem. Lecture Notes in Computer Science 496, 109–116.

Vaessens, R.J.M., 1995. Generalised job-shop scheduling: Complexity and local search. Ph.D. Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

Vaessens, R.J.M., 1996. Operations research library of problems. Management School, Imperial College, London.

Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K., 1995. A local search template (revised version). Memorandum Computer Science and Operations Research (COSOR), Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K., 1996. Job-shop scheduling by local search. INFORMS Journal on Computing 8, 302–317.

Van den Akker, J.M., 1994. LP-based solution methods for single machine scheduling problems. Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

Van De Velde, S., 1991. Machine scheduling and Lagrangian relaxation. Ph.D. Thesis. CWI, Amsterdam, The Netherlands.

Van Hulle, M.M., 1991. A goal programming network for mixed integer linear programming: A case study for the job-shop scheduling problem. International Journal of Neural Systems 2 (3), 201–209.

Van Laarhoven, P.J.M., Aarts, E.H.L., 1989. Simulated annealing: Theory and applications, Mathematics and its Applications Series. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K., 1988. Job-shop scheduling by simulated annealing. Report OS-R8809. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.

Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K., 1992. Job shop scheduling by simulated annealing. Operations Research 40 (1), 113–125.

Wagner, H.M., 1959. An integer programming model for machine scheduling. Naval Research Logistics Quarterly 6, 131–140.

Wang, W., Brunn, P., 1995. Production scheduling neural networks. In: Derigs, U., Bachem, A., Drexl, A. (Eds.), Operations Research Proceedings 1994. Springer, Berlin, pp. 173–178.

Watanabe, T., Tokumaru, H., Hashimoto, Y., 1993. Job-shop scheduling using neural networks. Control Eng. Prac. 1 (6), 957–961.

Wennink, M., 1995. Operations research library of problems. Management School, Imperial College, London (Anonymous FTP site at ftp://mscmga.ms.ic.ac.uk/pub/job-shop1.txt).

Werner, F., Winkler, A., 1995. Insertion techniques for the heuristic solution of the job-shop problem. Discrete Appl. Math. 58 (2), 191–211.

White, K.P., Rogers, R.V., 1990. Job-shop scheduling: Limits of the binary disjunctive formulation. International Journal of Production Research 28 (12), 2187–2200.

Willems, T.M., Rooda, J.E., 1994. Neural networks for job-shop scheduling. Control Eng. Prac. 2 (1), 31–39.

Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevast'janov, S.V., Shmoys, D.B., 1997. Short shop schedules. Operations Research 45 (2), 288–294.

Yamada, T., Nakano, R., 1992. A genetic algorithm applicable to large-scale job-shop problems. In: Männer, R., Manderick, B. (Eds.), Proceedings of the Second International Workshop on Parallel Problem Solving from Nature (PPSN'2), Brussels, Belgium, pp. 281–290.

Yamada, T., Rosen, B.E., Nakano, R., 1994. A simulated annealing approach to job-shop scheduling using critical block transition operators. In: IEEE International Conference on Neural Networks (ICNN'94), Orlando, Florida, USA, pp. 4687–4692.

Yamada, T., Nakano, R., 1995a. Job-shop scheduling by simulated annealing combined with deterministic local search. In: Metaheuristics International Conference (MIC'95), Hilton, Breckenridge, Colorado, USA, pp. 344–349.

Yamada, T., Nakano, R., 1995b. A genetic algorithm with multi-step crossover for job-shop scheduling problems. In: Proceedings of the Int. Conf. on GAs in Eng. Sys. (GALESIA'95), pp. 146–151.

Yamada, T., Nakano, R., 1996a. Job-shop scheduling by simulated annealing combined with deterministic local search. Meta-heuristics: Theory and Applications. Kluwer Academic Publishers, Hingham, MA, pp. 237–248.

Yamada, T., Nakano, R., 1996b. Scheduling by genetic local search with multi-step crossover. In: Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV), Berlin, Germany, 22–26 September, pp. 960–969.

Yamada, T., Nakano, R., 1996c. A fusion of crossover and local search. In: IEEE International Conference on Industrial Technology (ICIT'96), Shanghai, China, pp. 426–430.

Yannakakis, M., 1990. The analysis of local search problems and their heuristics. In: Lecture Notes in Computer Science 415. Springer, Berlin, pp. 298–311.

Zhou, D.N., Cherkassky, V., Baldwin, T.R., Hong, D.W., 1990. Scaling neural networks for job-shop scheduling. In: International Joint Conference on Neural Networks (IJCNN'90), San Diego, California, pp. 889–894.

Zhou, D.N., Cherkassky, V., Baldwin, T.R., Olson, D.E., 1991. A neural network approach to job-shop scheduling. IEEE Transactions on Neural Network 2 (1), 175–179.