# 1

# Exact, Heuristic and Meta-heuristic Algorithms for Solving Shop Scheduling Problems

G.I. Zobolas[1], C.D. Tarantilis[2], and G. Ioannou[3]

[1] Athens University of Economics and Business, Department of Management Science & Technology Management Science Laboratory Evelpidon 47A & Leukados 33, 11369, Athens, Greece gzobolas@aueb.gr
[2] Athens University of Economics and Business, Department of Management Science & Technology Management Science Laboratory Evelpidon 47A & Leukados 33, 11369, Athens, Greece tarantil@aueb.gr
[3] Athens University of Economics and Business, Department of Management Science & Technology Management Science Laboratory Evelpidon 47A & Leukados 33, 11369, Athens, Greece ioannou@aueb.gr

**Summary.** This chapter sets out to present a very important class of production scheduling problems and the main methods employed to solve them. More specifically, after a brief description of single and parallel machines scheduling problems, which constitute the basis of production scheduling research, the main shop scheduling problems are presented (flow shop, job shop, open shop, group shop and mixed shop) followed by an analysis of their computational complexity. Thereafter, the most important exact, heuristic and meta-heuristic methods are presented and classified. Finally, a thorough review for each shop scheduling problem is conducted where the most important methods proposed in the literature, specifically for each problem, are presented.

**Key words:** Production Scheduling, Single and Parallel Machines Scheduling, Shop Scheduling, Flow Shop, Job Shop, Open Shop, Group Shop, Mixed Shop, Meta-heuristics.

## 1.1 Introduction

The production process of manufacturing enterprises has always been a key factor for overall business success. Production scheduling problems are faced by thousands of companies worldwide that are engaged in the production of tangible goods. Thus, it is not without reason that efficiently and effectively solving production scheduling problems has attracted the interest of many practitioners and researchers from both fields of production management and

G.I. Zobolas et al.: *Exact, Heuristic and Meta-heuristic Algorithms for Solving Shop Scheduling Problems*, Studies in Computational Intelligence (SCI) **128**, 1–40 (2008)
www.springerlink.com 

combinatorial optimization. This interest is further amplified by the similarity of production scheduling problems with problems arising in other scientific areas (e.g. packet scheduling in telecommunication networks, PCB design, routing, timetabling etc) [97] and, therefore, the applicability of the developed methods in these areas as well. It should be also mentioned that due to the complex nature of scheduling problems, many new computational methods for their solution have emerged which can also be applied to a wide range of combinatorial optimization problems.

Due to the virtually unlimited number of different production environments, many variations of production scheduling problems exist. However, academic research and solution methodology development have focused mainly on a limited number of classical problems which, most of the times, cannot be directly applied to complex manufacturing structures. Thus, the development of flexible solution methodologies, which can be modified and applied to several different cases, is of critical importance for production management practice.

Production scheduling problems have detained thousands of researchers worldwide during the $20^{th}$ century. Initially, research focused on simplified and generic problems with limited applicability in real cases [78]. Most such problems dealt with optimizing a single objective in one-machine environments and featured many simplified assumptions. Since then, a wide variety of scheduling problems (and their variants) has been identified and an even wider range of solution methodologies has been proposed. At the very beginning, research focused on exact methods, i.e. methods that guaranteed the optimum solution of a given problem. Due to the lack of computational resources and the need to solve large scale scheduling problems, it was soon realized that exact methods were impractical and thus research focused on problem specific heuristics. On the other hand, the need for more robust solution methodologies led to the development and application of the first meta-heuristic methods (the term meta-heuristic was proposed by Glover in 1986 [57]) whose performance is continuously improving. Contemporary methodologies usually combine several heuristic and meta-heuristic algorithms (hybrid algorithms) in an effort to overcome the inherent limitations of single meta-heuristic components.

This chapter focuses on the most important shop production scheduling problems and the solution methodologies employed to solve them. The remainder is organized as follows: Section 1.2 is devoted to the presentation of the most classical production scheduling problems focusing on shop scheduling. Section 1.3 describes the main categories of solution methodologies emphasizing the latest meta-heuristic algorithms. Section 1.4 addresses the Flow Shop Scheduling Problem (FSSP) and relative research on solution methodologies. Similarly, Section 1.5 considers the Job Shop Scheduling Problem (JSSP) and finally, Section 1.6 covers the Open Shop Scheduling Problem (OSSP).

## 1.2 Production Scheduling Problems and their Classification

Although production scheduling problems have overlapping characteristics, they can be classified based on several facets [30]. Among them, the most widely used are the job arrival process, the inventory policy, various shop and job attributes and shop configuration.

Based on the job arrival process, production scheduling problems can be either ***static*** if all jobs arrive at the same time or ***dynamic*** if jobs arrive intermittently. Based on the inventory policy, a problem might be ***open*** if all products are made-to-order or ***closed*** if all products are made-to-stock. Hybrids of open and closed systems are very frequently observed in real cases. Production scheduling problems can also be classified as ***deterministic*** if job processing times and machine availability is known a priori or, conversely, ***probabilistic***. Among other job attributes based on which such problems can be classified is whether the production environment is single or multi stage. In a single stage environment, each job goes through one machine, whereas in multi stage environments, each job consists of several operations that might be processed in different machines. Finally, the number of jobs and machines and the flow pattern of jobs among machines also constitute a major classification facet.

It should be mentioned that during the last decades, academic research has focused mainly on static, deterministic, multi stage shop scheduling problems. Therefore, this chapter also focuses on these problems and their solution methodologies.

### 1.2.1 Single Machine Scheduling Problem

The simplest production environment is the one machine or single machine environment where all operations go through the same resource. The single machine scheduling problem was the first to be addressed academically and its characteristics and findings have been applied to more complex problems. They are also very useful for studying more complex serial structures where one machine is the bottleneck of the whole process and thus, generating a good schedule for the bottleneck machine is essential for the overall schedule performance. Generally, researchers focus on a specific performance measure and they try to develop methods to schedule operations in order to optimize the specific performance criterion. Finding the optimum solution with respect to the specific performance measure is not always feasible (see Section 1.2.4 about computational complexity). Analytically presenting all findings on single machine scheduling is out of the scope of this chapter and therefore research findings are summarized in Table 1.1 [144].

The Flowtime of a job is calculated as its completion time minus the release time, in other words it is the total time the job remains in the system. In case all jobs are not equally important, weights can be introduced so as the

Table 1.1: Single machine scheduling.

| Performance Measure | Optimum solution |
|---|---|
| Min Flowtime | Shortest processing time dispatching rule |
| Min weighted flowtime | Weighted shortest processing time dispatching rule |
| Min Total Lateness | Shortest processing time dispatching rule |
| Min Max Tardiness | Earliest due date dispatching rule |
| Min number of tardy jobs | Hodgson's algorithm |
| Min Tardiness | Heuristically optimized |
| Min weighted number of tardy jobs | Heuristically optimized |

scheduler can pay special attention to 'important' jobs (weighted Flowtime criterion). The Lateness of a job is calculated as its completion time minus its due date. If this value is positive then the Lateness is also called Tardiness while if this value is negative, Lateness is also denoted as Earliness. If there are specific due dates for all jobs, a very important performance measure is the 'number of tardy jobs' optimized by a special procedure proposed by Hodgson [144]. Also, see Table 1.2 for a list of the most widely used priority dispatching rules.

A very important variant of the single machine problem is the single machine scheduling problem with sequence dependent setups. This problem is equivalent to the notorious travelling salesman problem (TSP) which is NP-Hard. Small instances of these problems can be solved efficiently but in general, both problems are considered computationally intractable and medium-large instances can only be approximated with heuristic or metaheuristic methodologies.

### 1.2.2 Parallel Machine Scheduling Problem

Most of the times, real life scheduling problems consider multiple machines. Multiple machines may occur in parallel or in series or both. In parallel machine scheduling we consider that each job can be processed on any of the machines and processing times are independent to the machine (identical machines). The scheduling decisions involved are: a) which machine processes each job, b) in what order [144]. Similarly to the single machine case, some problems can be solved optimally while others are approximated. The minimum flowtime problem can be solved with the shortest processing time list dispatching rule (job with the shortest processing time assigned to the least loaded machine). On the other hand, large instances of makespan minimization problem cannot be solved with exact algorithms in practical computational times. Further details on parallel machine scheduling are also beyond of the scope of this chapter.

### 1.2.3 Major Shop Scheduling Problems

As the single and parallel machine problems rarely represent actual production environments, academic research soon focused on more complex problems with multiple jobs and resources to be considered. Based on the flow pattern of jobs towards the resources, one can distinguish among several types of shops. The most studied problems in the literature are the flow shop scheduling problem (FSSP), the job shop scheduling problem (JSSP) and the open shop scheduling problem (OSSP), which are analyzed in the following subsections. Apart from these main three shop scheduling problem cases, the mixed shop (MSSP) and group shop (GSSP) scheduling problems have recently been proposed but have attracted much less academic interest and their solution methodologies are usually variants of the meta-heuristic methods proposed for the three main problems.

#### Flow Shop Scheduling Problem

The general flow shop scheduling problem (FSSP) consists of a set of $N$ jobs $(1, 2, \ldots, n)$ to be processed on a set of $M$ machines $(1, 2, \ldots, m)$. In the FSSP, all jobs are processed sequentially on multiple machines in the same order. Additionally, each job can be processed on one and only machine at a time and each machine can process only one job at a time respectively. Additionally, all operations are assumed non-preemptable and setup times are included in the processing times and are independent to sequencing decisions. The scheduling problem lies in finding a sequence of jobs that optimizes a specific performance criterion (usually makespan, number of tardy jobs, total tardiness or total flowtime). According to Conway's [33] notation the FSSP with makespan criterion can be shown by $n/m/F/C_{max}$ and according to Graham et al [65], it can be shown by $F//C_{max}$.

Many variants of the general FSSP exist, like the zero-buffer flow shops or flow shop with blocking, the no-wait flow shops and the hybrid flow shops [74]. In the zero-buffer flow shop, a job $i$ having been processed on machine $j$ cannot advance to machine $j + 1$ if this machine is still processing the predecessor of job $i$. In this case, job $i$ must remain at machine $j$, also delaying job $i$'s successor to be processed on machine $j$. Generally, in the flow shop with blocking, there is no intermediate buffer and, therefore, a job cannot proceed to the next machine until this machine is free. In a no-wait flow shop, once a job is started on the first machine it has to be continuously processed through completion at the last machine without interruptions. If this is not possible, the start of a job on a given machine must be delayed so that the completion of the operation coincides the starting of the operation on the next machine. Finally, in the hybrid flow shop, there are $K$ serial workstations and there are one or more identical parallel machines at each workstation.

Although the variants of the FSSP have extensive industrial applications, academic research has focused mainly on a reduced version of the general

FSSP, the permutation flow shop scheduling problem (PFSP) with the added assumption that jobs must be processed in the same sequence by each of the $M$ machines. Due to the extensive academic research on the PSFP, the problem will be analyzed in the rest of this chapter.

**Job Shop Scheduling Problem**

The general job shop scheduling problem (JSSP) consists of a set of $N$ jobs $(1, 2, \ldots, n)$ to be processed on a finite set of $M$ machines $(1, 2, \ldots, m)$. In the general JSSP, each job must be processed on every machine and consists of a series of $m_i$ operations which have to be scheduled in a predetermined order, different for each job. These precedence constraints differentiate the JSSP from FSSP. Similarly to the FSSP, each job can be processed on one and only machine at a time and each machine can process only one job at a time respectively. Additionally, all operations are assumed non-preemptable and setup times are included in the processing times and are independent to sequencing decisions. The scheduling problem lies in finding a sequence of jobs for each machine that optimizes a specific performance criterion (usually the total makespan) while, at the same time, ensuring the observance of all problem constraints. According to Conway's [33] notation the JSSP with makespan criterion can be shown by $n/m/J/C_{max}$ and according to Graham et al [65], it can be shown by $J//C_{max}$.

At first, a solution to a job shop scheduling problem was represented with the use of Gantt charts. Although the Gantt chart is an excellent monitoring tool that displays operation processing throughout the time horizon, its limitations concerning the problem representation itself led to the development of other representation methods. Among them, the one that prevailed is the disjunctive graph representation proposed by Roy and Sussmann [138]. However, it should be mentioned that Gantt charts are still widely used in user interfaces to represent a solution. Fig. 1.1 displays a disjunctive graph for a 4x3 job-shop problem.

In the node-weighted disjunctive graph of Fig. 1.1, a vertex corresponds to each operation. The set of nodes $(N)$ represents operations to be processed on the set of machines $M$. The fictitious initial node $S$ is called the source and the final fictitious node $F$ the sink respectively. Each operation's processing time $t_{O_{kl}}$ is represented by the positive weight of each node $j$ (thus $t_S = t_F = 0$). $O_{kl}$ denotes that the specific operation belongs to job $k$ and is processed on machine $l$. Let $A$ be the set of directed conjunctive arcs (shown by complete lines) representing each job's precedence constraints, such that $(O_{kl},O_{km}) \in A$ indicates that operation $O_{kl}$ is an immediate predecessor of operation $O_{km}$ within the subset job's $k$ operations. Capacity constraints are represented by the set $E$ of uni-directed orientable edges (shown by dotted lines - one colour for each machine) where each member of $E$ is linked with a pair of disjunctive arcs sharing a common machine. Thus, two operations $O_{kl}$ and $O_{il}$ to be processed by the same machine $l$ cannot be executed simultaneously.

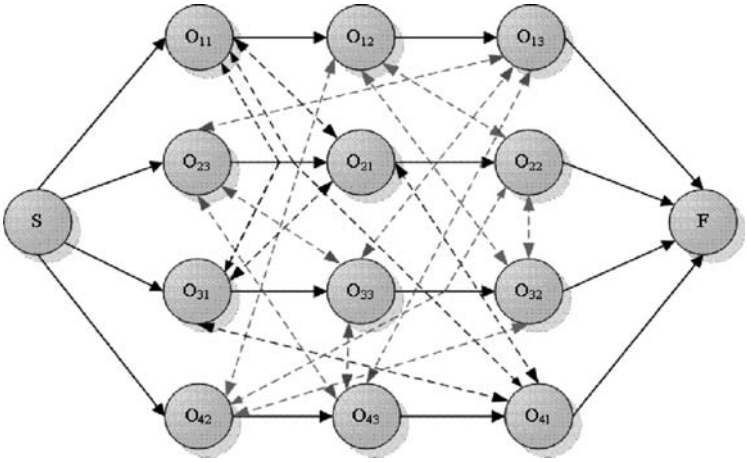


Fig. 1.1: The disjunctive graph representation for a 4x3 job shop problem.

### Open Shop Scheduling Problem

A special type of the JSSP is the open shop scheduling problem (OSSP). In the OSSP there is no predefined sequence of operations among jobs and, therefore, the OSSP has a considerably larger solution space than a JSSP with similar dimensions ($nxm$). Probably, the best example of open shop is a car repair shop where the operation/repair sequence is not strictly defined. Just like the FSSP and JSSP, the OSSP consists of a set of $N$ jobs $(1, 2, \ldots, n)$ to be processed on a finite set of $M$ machines $(1, 2, \ldots, m)$. In the general OSSP, each job must be processed on every machine and consists of a series of $m_i$ operations which have to be scheduled in any order. Similarly to the general FSSP and JSSP, each job can be processed on one and only machine at a time and each machine can process only one job at a time respectively. Additionally, all operations are assumed non-preemptable and setup times are included in the processing times and are independent to sequencing decisions. The scheduling problem lies in finding a sequence of jobs for each machine that optimizes a specific performance criterion (usually the total makespan) while, at the same time, ensuring the observance of all problem constraints. According to Conway's [33] notation, the JSSP with makespan criterion can be shown by $n/m/O/C_{max}$ and according to Graham et al [65], it can be shown by $O//C_{max}$.

### Mixed Shop and Group Shop Scheduling Problems

In 1985, the mixed shop scheduling problem (MSSP) was introduced by Masuda et al [108] and some years later, in 1997, the group shop scheduling

problem (GSSP) was proposed in the context of a mathematical competition [80]. In the MSSP, the machine routes of jobs can either be fixed or unrestricted [103]. The problem can also be regarded as a mix of the three aforementioned main shop scheduling problems. Similarly, the GSSP shares many characteristics of the three main shop scheduling problems. More specifically, let $O$ be a set of operations that is partitioned into subsets $J = \{J_1, ., J_n\}$, and subsets $M = \{M_1, ., M_m\}$, where $n$ is the number of jobs, $m$ is the number of machines. Let $J_i$ be the set of operations which belong to job $i$ and $M_k$ the set of operations which have to be processed on machine $k$. Each job's $i$ operations belong to $g$ groups $G = \{G_1, ..., G_g\}$. Within each group, operations are not restricted while, on the other hand, operations that belong to different groups must satisfy some precedence relationship between the groups imposed by the problem. In the special case where each operation constitutes a group, the GSSP is equivalent to the JSSP. Correspondingly, if for all jobs, all operations of jobs $i$ belong to the same group, the GSSP is equivalent to the OSSP.

Considering that the mixed shop and group shop scheduling are special cases of the three main shop scheduling problems and that the meta-heuristic solution methodologies used to solve them share many characteristics and in essence are simple variations of the ones proposed for the main shop scheduling problems, the MSSP and GSSP are not further analyzed in this chapter.

### 1.2.4 Computational Complexity of Shop Scheduling Problems

All shop scheduling problems belong to the NP class [95]. Although some special cases of these problems might be solved by specific algorithms in polynomial time, in the general case, they are computationally intractable when the number of machines is greater than three, which means that they can only be solved with deterministic algorithms with exponential behaviour. More specifically, the time required to solve shop scheduling problems increases exponentially with the size of the input [85]. In the following subsections, the computational complexity of the three major shop scheduling problems is analyzed.

#### Complexity of the FSSP

Small instances of flow shops can be solved optimally. For example, the minimum makespan model for two machines can be solved with Johnson's algorithm [86] which also solves special cases with three machines. More specifically, Johnson's algorithm may be applied to a flow shop scheduling problem when the middle machine (machine 2) is dominated by the other two (a machine $l$ dominates a machine $k$ if for each job $i$ the processing time of job $i$ on machine $l$ is greater than or equal to the processing time of job $i$ on machine $k$). However, many researchers [63, 96] have proved that the $n$-job $m$-machine flow shop sequencing problems belong to the class of NP-Hard problems, and

therefore, computational time for obtaining an optimal solution increases exponentially with increasing problem size. As a result, academic research has focused on the development of heuristic and meta-heuristic methods.

For an $n$-job, $m$-machine general flow shop scheduling instance, the maximum number of possible solutions is $(n!)^m$. Thus, even for a relatively small instance, the number of possible solutions is considerably large (e.g., for a 10x10 instance the maximum number of possible solutions is $(10!)^{10} = 3.96 \times 10^{65}$. Concerning the PFSP, a reduced version of the general flow shop, the maximum number of possible solutions is considerably smaller, i.e. $n!$. However, Garey et al [55] proved that the $F_3/prmu/C_{max}$ is also strongly NP-Hard ($F_3$ indicates the three-machine case and $prmu$ denotes the permutation flow shop variant).

### Complexity of the JSSP

The reason for the computational intractability of the JSSP is the fact that many different and conflicting factors must be taken into account. Such factors are due date requirements, cost restrictions, production levels, machine capacity, alternative production processes, order characteristics, resource characteristics and availability etc. However, it is the combinatorial nature of the job shop problem which determines its computational complexity [30]. Most job shop scheduling problems belong to the NP class [33]. Lenstra and Rinnooy Kan [95] proved that the general JSSP is NP-Hard for shops when the number of machines is greater than three. The only efficiently solvable cases of the JSSP are [17]:

- The number of jobs is equal to two [22].
- The two-machine JSSP where each job consists of at most two operations [83].
- The two-machine JSSP with unit processing times [72].
- The two-machine JSSP with a fixed number of jobs (repetitious processing of jobs on the machines). This case was solved by Brucker [23].

For an $n$-job, $m$-machine job shop scheduling instance, the complexity is equal to the FSSP case ($(n!)^m$ possible solutions).

### Complexity of the OSSP

Various polynomial algorithms have been proposed for some cases of OSSP. For the two-machine problem, a polynomial time algorithm has been proposed by Gonzalez and Sahni [62] while Adiri and Aizikowitz [4] presented a linear time algorithm for the three-machine OSSP, provided that one machine dominates one of the other two. Although these problems with special structures are polynomially solvable, Gonzalez and Sahni [62] proved that in the general case and if the number of machines is greater than three, the OSSP is

NP-Complete. For an $n$-job, $m$-machine open shop scheduling instance, the maximum number of possible solutions is $(nm)!$. Thus, even for a relatively small instance, the number of possible solutions is considerably large (e.g., even for a small 10x10 instance the maximum number of possible solutions is $100! = 9.33 \times 10^{157}$). As a result, research on OSSP solution methodology focused on heuristic and meta-heuristic methodologies.

### 1.2.5 Optimization of Production Scheduling Problems

Shop scheduling problems are typical representatives of the Combinatorial Optimization class of problems where solutions are encoded with discrete variables. Generally, combinatorial optimization problems are optimization problems where the set of feasible solutions is or can be reduced to a discrete one, and the goal is to find the best possible solution. According to Blum and Roli [19], a combinatorial optimization problem $P = (S, f)$ can be defined as:

- A set of variables $X = \{x_1, \ldots, x_n\}$.
- Variable domains $D_1, \ldots, D_n$.
- Constraints among variables.
- An objective function $f$ to be minimized/maximized where $f : D1 * \ldots * Dn \rightarrow \Re^+$.

In a combinatorial optimization problem, the set of all possible and feasible assignments is:

$$S = \{s = \{(x_1, u_1), \ldots, (x_n, u_n)\}\} \mid u_i \in D_i, s. \tag{1.1}$$

$S$ is called a search or solution space and consists of every possible solution to the specific problem. In order to find an optimum solution, the solution space has to be explored effectively and efficiently. This optimum solution minimizes or maximizes (depending on the problem) the objective function while satisfying all constraints of the specific problem. The optimum solution is also called global optimum.

Apart from the solution space, the coding space represents the set of possible assignments of a problem's variables after being encoded so that a specific method can be used. For example, Cheng et al [32] have conducted an extensive review of possible encoding methods when using evolutionary algorithms in JSSP. Among the representations mentioned, the operation-based one encodes each possible solution with a vector that corresponds to the operation sequence. In this scheme, the coding space consists of all possible operation permutations. On the other hand, the solution space of JSSP consists of detailed operation schedules for each machine and may also include non feasible schedules, i.e. schedules that do not satisfy all constraints of the problem. Other possible representations for such problems are the job-based, the preference-list based, the priority rule-based, the completion time-based, the machine-based and the random keys. In the job-based, the solution to a

given problem is given by a simple job permutation. More specifically, following this representation, the operations that belong to the first job of the sequence are scheduled in the earliest possible position and so on, so forth. The preference-list based representation is a priority rule permutation and at each position, the operation that complies with the given priority rule is scheduled. The completion time-based representation is based on a vector of completion times for all operations while the machine-based representation generates $m$ vectors with job permutations where $m$ is the number of machines. Finally, the random keys representation is mainly used with continuous optimization methods where an operation permutation is extracted from a series of real values with a heuristic procedure (usually, the position with the lowest real value is scheduled first and so on, so forth).

Ideally, a representation of a problem should link a single encoded solution (coding space) to a single feasible solution (solution space) and additionally, all possible and feasible solutions should be able to be encoded (full coverage of the solution space). However, in some cases, various encoding schemes not only direct to infeasible solutions but to illegal ones as well, i.e. solutions outside the solution space. Therefore, each problem's representation should be carefully chosen and always in conjunction with the solution methodology to be developed.

## 1.3 Solution Methodologies for Shop Scheduling Problems

Combinatorial optimization problems can be solved/optimized using two different approaches that are presented in the rest of this Section. The first approach is by using complete or exact algorithms. The second approach is applied to more complex or larger problems, leads to near-optimum solutions and is characterized by the use of approximate algorithms. The second approach can be further divided to heuristic and meta-heuristic methods. The scope of this chapter is to present the latest meta-heuristic trends on solving shop scheduling problems and thus, the analysis focuses on meta-heuristic methods.

### 1.3.1 Exact Algorithms

Exact or complete algorithms are guaranteed to find for every finite size instance of a combinatorial optimization problem an optimal solution in bounded time. However, for the typical combinatorial optimization problems, like the shop scheduling problems which are usually NP-Hard, no algorithms exist to solve these problems in polynomial time. Therefore exact algorithms need exponential computation time in most cases which leads to impractical computational burden for real large scale applications. The family of exact

methods is considerably large but the most common exact/complete methods for scheduling problems are branch and bound algorithms, mixed integer programming and decomposition methods.

### 1.3.2 Heuristic Algorithms

As seen, the use of complete/exact methods to solve complex combinatorial optimization problems often leads to impractical computational times. This phenomenon has led the vast majority of researchers on such problems to approximation methods. In approximation methods, the guarantee of finding optimal solutions is sacrificed in order to get near-optimum solutions in reasonable and practical computational times. The basic form of approximation algorithms is called 'heuristics', a name derived from the Greek verb '$\epsilon\upsilon\rho\iota\sigma\kappa\epsilon\iota\nu$' which means 'to find'. The usual classification of heuristic methods is constructive and local search methods [19].

#### Constructive

Starting from scratch, constructive algorithms generate solutions by gradually adding parts of the solution to the initially empty partial solution. In typical shop scheduling problems for example, these parts are usually operations. Constructive heuristics are generally the fastest approximate algorithms although some special implementations may induce high computational load. Their advantage in computational time requirements is counterbalanced by generally inferior quality solutions when compared to local search techniques. Among the most widely used constructive heuristics for shop scheduling problems are the various 'Dispatching Rules'. Table 1.2 summarizes the most common dispatching rules for shop scheduling problems [17].

Dispatching (or Priority) Rules are the most common heuristics for shop scheduling problems due to their easy implementation and low requirements in computational power. Although they perform very well in certain cases, no rule exists that can be applied to all shop problems and perform satisfactorily. Even worse, there is no way to estimate the performance of a dispatching rule for a specific instance a priori. It should be mentioned that some priority rules generate the optimum solution in certain simple problems (e.g. the minimization of flowtime in single-machine scheduling where the SPT priority rule generates the global optimum solution).

#### Local Search

According to [19], a neighborhood of a solution $s$ may be defined as a function $N : S \rightarrow 2^S$ where each $s$ is assigned a set of neighboring solutions $N(s) \subseteq S$. $N(s)$ represents all neighboring solutions of $s$. For every defined neighborhood of solutions, the solution or solutions of highest quality (i.e. the best objective

Table 1.2: Dispatching rules.

| Rule | Description |
|---|---|
| SOT | An operation with the shortest processing time on the machine considered |
| LOT | An operation with longest processing time on the machine considered |
| LRPT | An operation with longest remaining job processing times |
| SRPT | An operation with shortest remaining job processing times |
| LORPT | An operation with highest sum of tail and operation processing time |
| Random | The operation for the considered machine is randomly chosen |
| FCFS | The first operation in the queue of jobs waiting for the same machine |
| SPT | A job with smallest total processing time |
| LPT | A job with longest total processing time |
| LOS | An operation with longest subsequent operation processing time |
| SNRO | An operation with smallest number of subsequent operations |
| LNRO | An operation with largest number of subsequent operations |

function value) are called locally optimum solutions within the defined neighborhood. In the case where there is only one solution with the best objective function value, this local optimum is called strict locally optimum solution.

Local search algorithms start from an initial solution (most of the times generated by a constructive heuristic or randomly) and iteratively try to replace part or the whole solution with a better one in an appropriately defined set of neighboring solutions. In order to replace parts of an initial solution, local search methods perform a series of moves leading to the formation of new solutions in the same neighborhood. The most common moves are the $2-Opt$, the $1-1$ *exchange* and the $1-0$ *exchange* moves. The $2-Opt$ move reverses a set of tasks of random length in a machine while the $1-1$ *Exchange* move swaps two tasks from the same machine. Finally, the $1-0$ *Exchange* move transfers a task from its position in one machine to another position in the same machine [161]. Of course, the number of possible moves and the corresponding neighborhoods are virtually unlimited.

The main drawback of basic local search methods is that they get easily trapped in local optima as they are myopic in nature. More specifically, local search with appropriate moves can be very effective in exploring a neighborhood of an initial solution but no mechanism exists that can lead to other distant neighborhoods of the solution space where the global optimum may exist. To remedy this weakness, new modern local search methods (explorative local search) have been developed with embedded meta-strategies to guide the search process. Such methods are presented in Section 1.3.3.

### 1.3.3 Meta-heuristic Algorithms

During the last decades, a new family of approximate algorithms has emerged and has dominated the combinatorial optimization problem solution research.

This new type of algorithm basically combines heuristic methods in higher level frameworks. The aim of the new methodology is to efficiently and effectively explore the search space driven by logical moves and knowledge of the effect of a move [19] facilitating the escape from locally optimum solutions. These methods are nowadays called meta-heuristics, a term that was first introduced by Glover [57]. Meta-heuristic methods have an advantage over simpler heuristics in terms of solution robustness; however they are usually more difficult to implement and tune as they need special information about the problem to be solved to obtain good results. Due to the computational complexity of combinatorial optimization problems, the moderate results acquired by heuristic methods and the time limitations for application of exact algorithms, the application of meta-heuristic methods to solve such problems is a well established field of research.

**Definition**

There are many definitions of meta-heuristic algorithms and methods. Perhaps, the most thorough definition was given by Stützle in 1999 and is cited in [19] (p. 270):

> *"Meta-heuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance. The main goal is to avoid disadvantages of iterative improvement and, in particular, multiple descent by allowing the local search to escape from local optima. This is achieved by either allowing worsening moves or generating new starting solutions for the local search in a more intelligent way than just providing random initial solutions. Many of the methods can be interpreted as introducing bias such that high quality solutions are produced quickly. This bias can be of various forms and can be cast as descent bias (based on the objective function), memory bias (based on previously made decisions) or experience bias (based on prior performance). Many of the meta-heuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in meta-heuristic algorithms randomness is not used blindly but in an intelligent, biased form."*

Based on the above definition, it could be said that meta-heuristics are an intelligent way to explore the solution space. Exact algorithms are too slow for even small to medium search spaces and simple heuristics blindly or myopically program the next move based on rules that cannot be characterized robust and logically defined. Based on recent findings, the best performing methods for optimizing shop scheduling problem are those encompassing hybrid systems such as local search techniques embedded within metastrategies that overcome local optimality by accepting non improving moves and, thus, inferior solutions. Although allowing non-improving moves seems contradicting at first, such strategies help meta-heuristic algorithms to escape local

optimality as the new inferior solutions might be in the neighborhood of the global optimum and therefore, a simple local search may thereafter lead to the global optimum solution.

Concerning the search direction strategy, the diversification and intensification strategies can be defined [19]. When a meta-heuristic method uses the diversification strategy, the main aim is the effective exploration of all possible neighborhoods of the solutions space. On the other hand, the intensification strategy focuses on the use of the gathered search knowledge and the exploration of a narrower solution subspace.

### Classification of Meta-heuristic Methods

Several meta-heuristic algorithm classification schemes exist based on various properties of these methods. Among them, the most meaningful and the most widely used is based on the quantity of solutions these algorithms deal with. More specifically, meta-heuristic algorithms can be divided in population-based and single point search. Population based meta-heuristic methods combine a number of solutions in an effort to generate new solutions that share good merits of the old ones and are expected to have better fitness. Such methods are iterative procedures that gradually replace solutions with better found ones. On the other hand, single point search methods improve upon a specific solution by exploring its neighborhood with a set of moves.

Another important and widely used classification scheme is based on the memory used during the search process. Concerning memory usage, it has nowadays become implicit in modern meta-heuristic methods. Memory usage constitutes a main characteristic of effective meta-heuristic methods and is the indication of the intelligence employed during the search process. Memory usage may be further divided in short-term and long-term. Short-term memory keeps track of recent moves and helps avoiding cycling around specific solutions. On the other hand, in the long-term memory, information is gradually stored and employed at specifically defined stages of the algorithm, depending on the implementation. Memory-less algorithms are nowadays rarely employed for complex combinatorial optimization problems.

### Meta-heuristic Algorithms used for Shop Scheduling Problems

Almost all meta-heuristic algorithms proposed in the literature have been employed to solve shop scheduling problems. For a thorough description of these methods, the reader may refer to Blum and Roli's work [19]. Epigrammatically, the most popular meta-heuristic methods for solving shop scheduling problems are:

- **Evolutionary Computation** algorithms that fall mainly into three main categories: Genetic Algorithms [77], Evolutionary Strategies [133] and Evolutionary Programming [49]. One of the newest algorithms of this category

is the Differential Evolution Algorithm (DEA) introduced by Storn and Price [146].
- **Particle Swarm Optimization** (PSO) [42].
- **Ant Colony Optimization** (ACO) introduced by Dorigo [40].
- **Scatter Search and Path Relinking** proposed by Glover et al [60].
- **Neural Networks** (NN) that constitute an advanced artificial intelligence technology that mimics the brain's learning and decision making process [51, 52].
- **Basic Local Search** where a neighborhood of a solution is explored with a set of moves and the local optimum is returned.
- **Explorative Local Search** mainly represented by the *Greedy Randomized Adaptive Search Procedure* (GRASP) proposed by Feo and Resende [45], *Variable Neighborhood Search* (VNS) proposed by Hansen and Mladenović [70] and *Iterated Local Search* (ILS) proposed by Stützle [149].
- **Simulated Annealing** (SA) proposed by Kirkpatrick et al [87].
- **Tabu Search** (TS) proposed by Glover [59].
- **Threshold Accepting** proposed by Dueck and Scheuer [41].

### Hybrid Methods

The development and application of hybrid meta-heuristic algorithms is increasingly attracting academic interest. Hybrid meta-heuristic algorithms combine different concepts or components from various meta-heuristics [19] and towards this end, they attempt to merge the strengths and eliminate the weaknesses of different meta-heuristic concepts. Therefore, the effectiveness of the solution space search may be further enhanced and new opportunities emerge which may lead to even more powerful and flexible search methods. Talbi [158] has proposed a taxonomy for hybrid meta-heuristics. Generally, we can distinguish three main forms of hybridization [19]. The first form is called *component exchange* among meta-heuristic methods and its most typical representative is the hybridization of population-based methods with local search methods. The second form is called *cooperative search* and typically involves the exchange of information between two or more different meta-heuristic algorithms. The information exchange level can vary from implementation to implementation as well as during the same implementation. The third form is called *integrating meta-heuristic algorithms and systematic methods* and has produced very promising results in real-world cases. Among the successful implementations of this form is the combination of meta-heuristic algorithms and constraint programming [128].

Most hybrid methods for shop scheduling methods belong to the first form of hybridization where various components and solution characteristics are shared among two or more heuristic and meta-heuristic methodologies. The power of hybrid meta-heuristic methods is indicated by the fact that, as we will see in the following sections of this chapter, current state-of-the-art methods for shop scheduling problems are all hybrid meta-heuristic algorithms.

**Adaptive Memory Programming - The Unified View**

Judging from the brief review of meta-heuristic algorithms in the previous section, it is obvious that the family of meta-heuristic methods is large and keeps growing as new forms and hybrids are still presented. Taillard et al [157] however mentioned that meta-heuristic methods at their present form share many common characteristics and could be unified under the term adaptive memory programming (AMP), firstly proposed by Glover [58]. More specifically, Taillard et al [157] mentioned that most meta-heuristic components follow a generalized procedure: at first, they memorize parts, whole solutions or characteristics of solutions during the search process. In a second stage, this stored information is used to generate new solutions and at the third stage, this new solution's neighborhood is explored by means of a local search method.

Thus, whether a meta-heuristic is a simple population-based genetic algorithm or a sophisticated hybrid incorporating various meta-heuristic components, the essence is that it usually follows the same three-step procedure. Therefore, under the unified view of adaptive memory programming, the various meta-heuristic methodologies might not be very different from each other after all. On the other hand, some methods (like simulated annealing and threshold accepting) cannot be generalized under the AMP scheme as their basic versions are virtually memory-less. However, their core methodology can be used in the improvement stage of the AMP framework (step three of AMP).

## 1.4 The Flow Shop Scheduling Problem

The flow shop scheduling problem is, as mentioned, NP-Hard. Due to the complexity of the problem, exact algorithms developed for the general FSSP failed to achieve high quality solutions for problems of increased size in reasonable time and thus academic research focused on heuristic and meta-heuristic methods. Regarding solution methodologies, Johnson [86] proposed an $O(nlogn)$ complexity algorithm which optimally solves the $F_2//C_{max}$ problem. Under the special circumstance where the middle machine is dominated by the other two, Johnson's algorithm may solve to optimality the $F_3//C_{max}$ problem. Among exact methods proposed for the general FSSP, we can distinguish among dynamic programming [75], branch and bound [37, 81, 90, 104] elimination rules [6] and, of course, complete enumeration which is the most time consuming exact method. Considering that these methods may be applied in small instances of FSSP, it is beyond the scope of this chapter to further analyze them.

### 1.4.1 Heuristics for the FSSP

A large number of heuristics has been proposed for the FSSP. The most important constructive ones are the Palmer [126], CDS [26], RA [38], Gupta [68],

NEH [118] and HFC [88]. Palmer [126] and Gupta [68] construction heuristics are based on the utilization of a criterion to generate the sequence of jobs. This criterion is a slope index that is calculated for each job based on the processing times and their pattern in terms of machine sequence. The CDS [26] algorithm on the other hand, generates $m-1$ artificial two-machine schedules (where $m$ is the number of machines) and solves them with Johnson's rule. The best $m-1$ solution is then chosen as the best sequence for the $m$-machine problem. According to the RA heuristic [38], a virtual two machine problem is defined (just like the CDS heuristic) but instead of directly applying Johnson's algorithm over the processing times, weighting schemes for each machine are calculated before.

The NEH heuristic [118] is much more complex than the aforementioned heuristics. Initially, jobs are arranged in a descending order of their total processing time. Then, based on this mentioned order, an increasingly larger partial sequence is generated at each step by introducing one job from the unscheduled order into the partial sequence (until all jobs are scheduled). At each step, a new job is scheduled in all possible positions ($k+1$ positions where $k$ is the size of last step's partial sequence) and after choosing the best place for this job, regarding the obtained makespan, this new partial sequence is fixed for the remaining procedure. The HFC heuristic [88] can only be applied to FSSP and not the reduced version of it, the PFSP. It is a two-stage method where Johnson's rule is used extensively in the first stage, while in the second stage, improvement of the initial schedule is performed by allowing job passing between machines (non permutation schedules).

In addition to the aforementioned constructive heuristics, a few improvement heuristics have been proposed by Dannenbring [38], Ho and Chang [76] and Suliman [150]. Contrary to constructive heuristics, improvement heuristics start from an already built schedule and try to improve it by some given procedure. Dannenbring [38] proposed two simple improvement heuristics based on the constructive heuristic RA proposed by the same author for initial solution generation: Rapid Access with Close Order Search (RACS) and Rapid Access with Extensive Search (RAES). RACS works by swapping every adjacent pair of jobs in a sequence. The best schedule among the $n-1$ generated is then given as a result. In RAES heuristic, RACS is repeatedly applied while improvements are found. Both RACS and RAES heuristics start from a schedule generated with the RA constructive heuristic.

Ho and Chang [76] developed a method that aims at the minimization of the elapsed times between the end of the processing of a job in a machine and the beginning of the processing of the same job in the following machine in the sequence. The authors refer to this time as 'gap' which is calculated for every possible pair of jobs and machines. Starting from the CDS heuristic solution, a series of calculations is conducted and then, the heuristic swaps jobs based on the corresponding gap value. Suliman [150] developed a two-phase improvement heuristic for the FSSP. In the first phase, similarly to Ho and Chang's improvement heuristic, a schedule is generated with the CDS heuristic

method. In the second phase, the schedule generated is improved with a job pair exchange mechanism. In an effort to reduce the computational load of an exhaustive job pair exchange, the number of possible moves is reduced by introducing a directionality constraint. More specifically, if a better schedule is acquired by moving a specific job forward, then this job is not allowed to move backward in the job sequence string.

Although FSSP heuristics perform very fast in most cases, they generally fail to produce high quality solutions. Even the NEH heuristic, which is considered the most powerful construction heuristic [154], fails to reach solutions even 5-7% worse than the optimum in some difficult instances due to Taillard [155]. Therefore, it is not without reason that most of the academic effort has been put towards the development of meta-heuristic and most recently, powerful hybrid meta-heuristic methods.

### 1.4.2 Meta-heuristics for the FSSP

A very large number of meta-heuristic algorithms, hybrid or not, have been proposed for the FSSP and PSFP. The application list for the FSSP includes most known meta-heuristic algorithms including evolutionary algorithms, particle swarm optimization, ant colony optimization, scatter search, neural networks, simulated annealing, tabu search and many forms of explorative local search. Most of these methods use simpler heuristic algorithms to generate an initial population of solutions.

Concerning the implementation of ***Genetic Algorithms*** (GA) in FSSP and PFSP, some early noteworthy research was performed by Reeves [134]. In his implementation, the offsprings generated at each step of the algorithm do not replace their parents but solutions with a fitness value below average. Among the innovations presented in this paper, Reeves used an equivalent to the One Point Order Crossover operator, called C1. Moreover the algorithm uses an adaptive mutation rate where the position of one job is simply changed by the shift mutation operator. A common feature of many meta-heuristic methods for the FSSP that is also found in this implementation is the use of the NEH heuristic to seed the initial population with a good sequence among randomly generated ones. Finally, another innovative feature of Reeves' implementation is that during selection, parent one is selected using a fitness rank distribution while parent two is chosen using a uniform distribution. During the same year, Chen et al [31] presented their own GA for the PFSP. Their implementation was basically a simple genetic algorithm with some added features and enhancements. More specifically, in this implementation the initial population is generated with the CDS and RA heuristics and also from simple job exchanges of some individuals. The crossover operator used is the Partially Mapped Crossover (PMX) and it is noteworthy that no mutation is applied.

Murata et al [115] presented their own version of GA for the PSFP where the two-point crossover operator and a shift mutation along with an

elitist strategy to obtain good solutions are used. Based on initial results, the algorithm failed to achieve results competitive to other meta-heuristic methodologies and therefore, the authors developed two hybrid versions; genetic simulated annealing and genetic local search. As their name implies, in these algorithms an improvement is conducted before the selection and crossover phases by means of a simulated annealing and local search algorithms respectively. These two hybrid algorithms performed better than the non-hybrid genetic algorithm and the simple implementations of tabu search, simulated annealing and local search. Reeves and Yamada [135] presented another hybrid genetic algorithm. The innovation in this algorithm was the use of a Multi-Step Crossover Fusion (MSXF) operator which combined crossover with a local search procedure. This operator uses one parent as a reference to conduct a biased local search to the other. The calculation of new upper bounds for some Taillard's [155] benchmark instances is indicative of the algorithm's high performance.

Another genetic algorithm was presented by Ponnambalam et al [130]. Their algorithm features the Generalised Position Crossover (GPX), shift mutation and a randomized initial solution. A hybrid implementation based on genetic algorithms and simulated annealing was presented by Wang and Zheng [167]. The powerful NEH heuristic is also used for population initialization and multi-crossover operators are used for solution recombination. However, the mutation operator is replaced by a simulated annealing component. Finally, one of the most recent genetic algorithms for the PFSP and a hybrid version of it were proposed by Ruiz et al [140]. Ruiz et al used a modified NEH heuristic to generate a diversified initial population, proposed four new crossover operators, used shift mutation and implemented specialized restart techniques for population renewal. Finally, they hybridized their genetic algorithm with a local search scheme.

Similarly to the Genetic Algorithm, the ***Differential Evolution Algorithm*** (DEA) has also been proposed, in a hybrid form, to solve the PFSP. More specifically, Tasgetiren et al [162] have proposed a DEA coupled with local search to optimize the PFSP. The authors borrowed the random key representation by Bean [14] to convert the real variables to discrete ones and used a repairing technique so a basic local search could be applied to the resulting solutions. Another attempt to apply the DEA to PSFP for minimizing makespan, total flowtime and tardiness has been conducted by Onwubolu and Davendra [124]. The authors compared the proposed method with the classic genetic algorithm and concluded that DEA demonstrates competitive performance and very easy implementation for the specific problem. Considering that the DEA in its canonical form operates with real variables, the authors proposed a transformation scheme to convert real-coded solutions to discrete ones and vice versa.

***Particle Swarm Optimization*** (PSO) has only recent and thus limited number of applications. The two major papers in this field have been presented by Lian et al [98] and Tasgetiren et al [163]. Lian et al [98] proposed a

conversion technique to apply the PSO algorithm in discrete problems like the FSSP and compared the PSO with a traditional GA. The authors concluded that the PSO performed better than the traditional GA. Similarly, Tasgetiren et al [163] proposed a hybrid PSO algorithm for the PFSP with both makespan and total flowtime minimization criteria. More specifically, they hybridized the basic PSO algorithm with an explorative local search technique, the Variable Neighborhood Search (VNS). They also proposed a scheme to allow the PSO to operate in the discrete variable environment of PFSP. This heuristic scheme is named Shortest Position Value (SPV) and it is borrowed from the random key representation of Bean [14]. Computational tests conducted by the author revealed the strength of the proposed method for both performance criteria and in both Taillard and Watson benchmark instances.

The ***Ant Colony Optimization*** (ACO) algorithm has also been used for solving the FSSP/PFSP. One of the first attempts to apply ACO to PFSP was done by Stützle [147] and some years later by Ying and Liao [177]. Ying and Liao's computational experiments on Taillard benchmark instances revealed that the ACO approach is a very effective meta-heuristic for the PFSP. Among other presented implementations, T'kindt et al [152] proposed a version of ACO to solve the two-machine flow shop with two criteria. More recently, Rajendran and Ziegler [132] proposed two versions of ACO to solve the PFSP with the minimum makespan and minimum total flowtime objectives. The first algorithm presented is an extension of Stützle's [147] ideas of the ant colony algorithm (MMAS algorithm) by including the summation rule suggested by Merkle and Middendorf [112] and a local search technique.

Nowicki and Smutnicki [121] presented a ***Scatter Search and Path Relinking*** version (algorithm MSSA) for the PFSP and have conducted extensive research on the properties on the problem's solutions space. The strength of their implementation is attested by some new upper bounds found in Taillard's benchmark instances for minimizing the total makespan. More specifically, the authors have used their TSAB algorithm and extended it using a modified scatter search algorithm (MSSA). Finally, one of the most important characteristics of their algorithm is that its good properties remain scalable with increasing instance size.

There are quite a few works about ***Neural Networks*** (NN) applied to FSSP, although most of them are hybrid implementations with other meta-heuristics. One of the first works in this field was conducted by Lee and Shaw [94]. The authors developed a hybrid neural network-genetic algorithm for the FSSP showing good performance. A few years later, Solimanpur et al [145] developed a hybrid tabu search-neural network approach called EXFS. In their implementation, the NEH heuristic is initially used to construct a solution. Thereafter, tabu search is used to improve the solution and special neurons are employed to penalize some moves. A more recent implementation was developed by El-bouri et al [44] for the PFSP. More specifically, they used solved instances of PFSP instances to train some neurons. Afterwards,

the neurons were employed to guide a local search procedure to the most promising job assignments.

Concerning ***Basic Local Search*** and ***Explorative Local Search*** methods, there are very few implementations where such methods are used alone (i.e. not as a part of a hybrid algorithm). It should be mentioned though that, when hybridized, such methods greatly improved the performance of other methods, especially of evolutionary methods. One of the few implementations was the iterated local search (ILS) procedure proposed by Stützle [148] which was later recoded by Ruiz and Maroto [139]. The latter proved that ILS performed very satisfactorily in the PFSP.

***Simulated Annealing*** (SA) was one of the first meta-heuristic methods proposed [87] and therefore it is not without reason that a very large number of SA algorithms have been proposed for the FSSP/PFSP. One of the first works in this area has been conducted by Osman and Potts [125]. The authors proposed a simple SA algorithm using a shift neighborhood and a random neighborhood search. Ogbu and Smith [123] proposed an SA algorithm for the PFSP which involved an initialization with the Palmer [126] and Dannenbring's [38] RA heuristics. In a later work by the same authors [122], their initial approach was compared to that of Osman and Potts [125] where Ogbu and Smith's [122] algorithm was found slightly more efficient. Gangadharan and Rajendran [54] also applied the SA method to solve the FSSP. Two SA algorithms comparable in performance with Osman and Potts's [125] approach were developed by Ishibuchi et al [82].

Zegordi et al [179] developed a hybrid simulated annealing algorithm called SAMDJ with the incorporation of problem domain knowledge in the basic SA scheme. More specifically, they used a specially formulated table with several rules concerning the biased movement of jobs towards specific positions or directions in the sequence. It was empirically proven that this table facilitated the annealing process and lessened the SA control parameters. Performance wise, SAMDJ proved to be slightly inferior in terms of solution quality compared to Osman and Potts's [125] SA algorithm but considerably faster. Murata et al [115], as already mentioned in the genetic algorithm section, developed a hybrid genetic-simulated annealing algorithm for the PFSP which, according to their experiments, outperformed the single meta-heuristic components when used individually. The same procedure was followed by Moccellin and dos Santos [114] who presented a hybrid tabu search - simulated annealing heuristic. The hybrid algorithm was then compared to simple tabu search and simple simulated annealing algorithms, showing the superiority of the hybrid approach. Wodecki and Božejko [170] proposed an SA algorithm that was run in a parallel computing environment. The algorithm was tested against the classic NEH heuristic and superior results were recorded. Finally, Hasija and Rajendran [71] proposed an algorithm based on simulated annealing for the PFSP with total tardiness of jobs criterion and two new solution perturbation schemes. The authors concluded that their algorithm performed better than existing tabu search and simulated annealing techniques.

***Tabu Search*** (TS) has also been extensively used to solve the FSSP/PFSP in single or hybrid forms. One of the first attempts towards this direction was conducted by Widmer and Hertz [169] who presented the 'SPIRIT' method, a two-stage heuristic. In the first stage, an initial solution is calculated with an insertion method in direct relation to the Open Travelling Salesman Problem (OTSP). In the second stage, a standard TS meta-heuristic with exchange neighborhood is used to improve the initial solution. Taillard [154] also presented a similar procedure to that of Widmer and Hertz [169]. In Taillard's implementation, an improved version of the classic NEH heuristic is used to obtain good initial schedules which are then improved by a tabu search procedure. After having tested various types of neighborhoods, the one-job change of position proved to be superior for the specific implementation.

Another meta-heuristic algorithm based on Widmer and Hertz's [169] SPIRIT heuristic is the TS method of Moccellin et al [113]. Their implementation resembles Widmer and Hertz's procedure with the main difference being the calculation of the initial solution. One of the most important works in this field was conducted by Nowicki and Smutnicki [120]. The authors proposed a TS meta-heuristic where only reduced parts of the possible neighborhoods are evaluated along with a fast method for obtaining the makespan after performing moves. Although this method is only suitable for the minimization of makespan criterion (and not the also widely used total flowtime of jobs) it quickly became the benchmark for other methods due to its high performance in terms of solution quality and computational speed. This fact is also demonstrated by the new best upper bounds found in Taillard's benchmark instances. The innovation in Nowicki and Smutnicki's work is that instead of moving single jobs during the search procedure, whole blocks of jobs are moved.

A few years later, Ben-Daya and Al-Fawzan [15] implemented a TS algorithm with extra intensification and diversification schemes that enabled better moves in the TS process. The algorithm proposed provides similar results to the TS algorithm of Taillard [154], indicating a successful implementation. Finally, as already mentioned in the SA section, a successful hybrid TS-SA method was proposed by Moccellin and dos Santos [114]. More specifically, a classic TS was hybridized with an SA procedure and the hybrid algorithm outperformed the single meta-heuristic components. One of the most recent implementations for the PFSP with tabu search was proposed by Solimanpur et al [145]. The authors developed a hybrid tabu search - neural network approach and they implemented block properties like Taillard [154]. Finally, Grabowski and Wodecki [64] proposed a fast TS approach.

***Threshold Accepting*** has also been used to solve PFSP by Gupta et al [69]. More specifically, the authors developed and compared various local search schemes for the two-machine PFSP with two criteria: the main criterion being the optimization of makespan and a secondary criterion being either the total flow time, total weighted flow time, or total weighted tardiness. Among the methods tested are simulated annealing, genetic algorithms,

threshold accepting, tabu search, and multi-level search algorithms. The authors conclude that the multi-level search heuristics were most appropriate for the flow time related secondary objective, while simulated annealing proved to be superior best for the due date related objective. The authors also concluded that the genetic algorithm performed poorly if not coupled with some other local search methods.

## 1.5 The Job Shop Scheduling Problem

The history of the job shop scheduling problem, starting more than 40 years ago, is closely linked to the history of the most known benchmark instance introduced by Fisher and Thompson [46]. This particular 10-job, 10-machine instance (also known as MT10 or FT10) detained researchers for over 25 years and signaled a continuous competition among researchers for the most powerful solution procedure. The JSSP solution history is characterized by circles concerning researchers' preferences. More specifically, at first, researchers proposed some exact methods followed by an era of heuristic methodologies development. Thereafter, researchers focused again on exact methods and JSSP complexity until the new era of sophisticated heuristic and meta-heuristic algorithms began.

Branch and bound algorithms are the most efficient exact methods for JSSP and they explore specific knowledge about the critical path of the problem. The main principle behind them is the enumeration of all possible feasible solutions of the problem so that properties or attributes not shared by any optimal solution are detected as early as possible. A branch of the enumeration tree defines a subset of the feasible solutions where each element of the subset satisfies this branch. For each subset, the objective value of its best solution is estimated by a lower bound. In case this lower bound is greater than the best known upper bound the subset can be dropped from further consideration. The best known upper bound can be a heuristic solution of the original problem [17]. Although branch and bound algorithms guarantee the optimal solving of the problem, they tend to be very slow in combinatorial optimization problems and rather impractical to use. Their main drawback is the lack of strong lower bounds in order to cut branches of the tree as early as possible and reduce the computational time required. Researchers are focused on finding ways to improve lower bounds.

One of the earliest works in the field of exact methods was performed by Brooks and White [21] and Greenberg [66]. These methods were based on Manne's [106] integer programming formulation. Other papers in this field are presented by Balas [7], Florian et al [48] and Fisher [47]. A very efficient method was proposed by McMahon and Florian [111] which turned out to be the best performing exact method for several years. The authors combined the bounds for the single-machine scheduling problem with the objective function and operation release dates in order to minimize maximum lateness

with the enumeration of active schedules. They were based on the fact that among active schedules, the optimal ones existed. In more recent methods, the neighborhood structure used in some recent local search implementations was employed as a branching structure [12]. The optimum solution of the infamous FT10 instance was first found by Lageweg in 1984 as reported by Lawler et al [91] although no proof of optimality was given. In this work, multi-machine lower bounds were used. Optimality of the found solution to FT10 (with an optimum makespan of 930) was first proven by Carlier and Pinson [28]. The latter were based on one-machine bounds and several simple inference rules on operation subsets. Current research status concerning branch and bound schemes is characterized by the very well performing methods of Applegate and Cook [5] and Martin and Shmoys [107]. Moreover, the use of advanced inference rules characterize the very well performing branch and bound methods of Baptiste et al [10, 11], Carlier and Pinson [29] and Brucker et al [25]. Finally, it should be mentioned that an emerging field of research concerning branch and bound techniques is their coupling with meta-heuristic strategies to form advanced hybrid algorithms.

### 1.5.1 Heuristics for the JSSP

Early works for the JSSP can be found as early as 1956 when Jackson [83] generalized Johnson's [86] rule for the FSSP and applied it to the two-machine JSSP. Since then, a large number of heuristic methodologies have been proposed. Perhaps the most frequently applied heuristics for JSSP are the various dispatching rules, presented in Section 1.3.2. One of the most important implementations of priority dispatching rules is Giffler and Thompson's algorithm [56] which can be generally considered as a common basis of all priority rule based heuristics. Their algorithm is a constructive heuristic where at each step, an unscheduled operation is chosen randomly from a conflict set of operations that 'compete' for the same machine. This machine corresponds to the machine that the operation with the lowest processing time among all unscheduled operations is to be processed at.

Apart from the priority dispatching rules, one of the most powerful heuristics for the JSSP is the shifting bottleneck heuristic originally proposed by Adams et al [3] and later improved by Balas et al [8]. The main concept of this algorithm is the solution, for each machine, of a one-machine scheduling problem to optimality, under the assumption that a lot of arc directions in the optimal one-machine schedules coincide with the optimal job shop schedule. The algorithm optimizes the schedule of the bottleneck machine first. The bottleneck machine is defined as the machine whose one machine optimum schedule has the longest makespan. The algorithm continues until all machines are scheduled. The power of this method is that although one-machine schedules are also NP-Hard, algorithms exist to efficiently solve them [27]. However, the assumption that the lot of arc directions in the optimal one machine schedules coincide with the optimal job shop schedule is not true and

thus this method does not lead to the global optimum solution. However, the shifting bottleneck algorithm has proven to be a very effective heuristic with many successful implementations in JSSP. This heuristic is also widely used nowadays for efficient hybrid algorithm implementations.

### 1.5.2 Meta-heuristics for the JSSP

Similarly to the FSSP, a very large number of hybrid and non hybrid meta-heuristic algorithms have been proposed for the JSSP. The application list for the FSSP includes most known meta-heuristic algorithms including evolutionary algorithms, particle swarm optimization, ant colony optimization, scatter search, neural networks, simulated annealing, tabu search and many forms of explorative local search. Most of these methods use simpler heuristic algorithms to generate an initial population of solutions.

Concerning the implementation of ***Genetic Algorithms*** (GA) in JSSP, many researchers have concluded that GAs are not suitable for fine-tuning of solutions close to optimal ones. In other words, genetic algorithms fail to intensify the search to the most promising regions of a neighborhood. This is why successful implementations of genetic algorithms usually incorporate a local search procedure for search intensification. A large number of early genetic algorithm applications may be traced in the literature. Among, the most well-known methods are those of Davis [39], Nakano and Yamada [117], Yamada and Nakano [172], Tamaki and Nishikawa [159], Mattfeld et al [110], Croce et al [34], Bierwirth et al [16] and Cheng et al [32]. Among more recent works in this field, Zhou et al [184] proposed a hybrid genetic algorithm with a neighborhood search procedure which also used a series of priority dispatching rules in the genetic evolution process. A very successful implementation of hybrid genetic algorithms can be found in the work of Wang and Zheng [166]. The authors also mention the drawback of genetic algorithm operators to disrupt the search in areas close to local or global optima and thus, they hybridized their GA with a simulated annealing algorithm which enhanced the intensification operation of their hybrid method. Park et al [127] also proposed a genetic algorithm and used Giffler and Thompson's algorithm [56] for the initial chromosomes generation. Murovec and Šuhel [116] presented a hybrid genetic-tabu search implementation with exceptional results in all benchmarks tested. This is also indicated by the three new upper bounds found for the ABZ9, YN1 and YN2 benchmark instances. Gonçalves et al [61] have presented an efficient hybrid GA implementation. The authors utilized the random keys representation [14] and hybridized their genetic algorithm with a local search scheme. Finally, Zobolas et al [185] developed a three-component hybrid algorithm where a Genetic Algorithm, a Differential Evolution and a Variable Neighborhood Search procedure were employed.

The ***Particle Swarm Optimization*** (PSO) method applications in JSSP are relatively very recent, indicating the great interest of the academic community in this optimization method. Although PSO can be generally

used with continuous variables and, therefore, is unsuitable for discrete variables, most researchers have presented special transformation methods or variations of the original PSO so that this method can be also applied to JSSP. Lian et al [99] presented a similar PSO algorithm with the added feature of converting a continuous domain to a discrete domain. Xia and Wu [171] hybridized the PSO algorithm with a classic simulated algorithm and formed their HPSO algorithm. Zhao and Zhang [181] also presented a hybrid particle swarm - simulated annealing algorithm and compared the hybrid version with the single algorithm implementations. Their hybrid method was found to be more effective and more efficient than the single meta-heuristic components. Finally, Sha and Hsu [143] also used a transformation routine but unlike most researchers, utilized the preference list-based representation and hybridized their PSO implementation with a tabu search procedure to improve the solutions acquired.

The use of ***Ant Colony Optimization*** (ACO) algorithm in solving the JSSP is rather limited. Just like the PSO algorithm, the implementations found in literature are very recent. Huang and Liao [79] proposed a hybrid ant colony - tabu search algorithm. The authors employed a novel decomposition method inspired by the shifting bottleneck procedure and a mechanism of occasional re-optimizations of partial schedules instead of using the more conventional feasible solutions construction approach. Heinonen and Pettersson [73] presented a hybrid ant colony - local search algorithm and focused on the infamous FT10 benchmark instance so that the ACO performance can be evaluated.

The application of ***Neural Networks*** (NN) to the JSSP can be characterized quite extensive. However, the last few years there seems to be a decreasing interest towards neural network implementations for JSSP. Among early works conducted in this field are the works of Foo and Takefuji [51–53], Zhou et al [182, 183] and Dagli et al [35]. However, mediocre results have been obtained and only the work of Sabuncuoglu and Gurgun [141] produced good results in benchmark tests. Jain and Meeran [84] also presented a powerful neural network scheme for the JSSP. There are quite a few implementations of hybrid neural networks with genetic algorithms including the algorithms of Dagli and Sittisathanchai [36], Lee and Dagli [93] and Yu and Liang [178]. Among more recent neural network hybrid methods is the work of Luh et al [105]. The authors combined recurrent neural network optimization ideas with Lagrangian relaxation for constraint handling to solve the JSSP. According to their tests, they managed to outperform most neural network implementations presented up to that date. Yang and Wang [176] have presented an adaptive neural network and heuristics hybrid approach for JSSP. More specifically they implemented two heuristics that operate in the neural network framework and are used to accelerate the solving process of the neural network and guarantee its convergence and to obtain non-delay schedules from the feasible solutions gained by the neural network. Finally, one of the most recent works in the domain of neural networks for JSSP is the work of

Fonseca and Navaresse [50] concerning job shop simulation. The proposed scheme managed to satisfactorily estimate the manufacturing lead times for orders simultaneously processed in a four-machine job shop. The authors compared their findings with data generated from three well-known simulation packages (Arena, SIMAN and ProModel) and found out that the manufacturing lead times produced by their scheme turned out to be equally valid.

Concerning ***Basic Local Search*** and ***Explorative Local Search*** methods, similarly to the FSSP/PFSP, there are very few implementations where such methods are used alone (i.e. not as a part of a hybrid algorithm). It should be mentioned though that, when hybridized, such methods greatly improve the performance of other methods, especially of evolutionary methods. One of the few implementations was the Greedy Randomized Adaptive Search Procedure (GRASP) of Resende [136] and the Guided Local Search - Shifting Bottleneck hybrid of Balas and Vazacopoulos [9]. Both algorithms and especially the one of Balas and Vazacopoulos, have demonstrated very good performance in the JSSP.

***Simulated Annealing***, as mentioned in the FSSP section, is one of the earliest proposed meta-heuristic [87]. The most known implementations of simulated annealing have been proposed by Matsuo et al [109], Van Laarhoven et al. [164,165], Aarts et al [1,2], Yamada et al. [175], Sadeh and Nakakuki [142] and Yamada and Nakano [173, 174]. However, as Jain and Meeran [85] observed, simulated annealing is unable to quickly achieve good solutions to JSSP problems, perhaps because it is is a generic and memory-less technique. As a result, academic research focused on hybrid versions of simulated annealing some of which are already mentioned in the previous subsections of meta-heuristics for the JSSP (e.g. the work of Wang and Zheng [166]). Zhang et al [180] recently proposed a hybrid simulated annealing - tabu search approach whose main principle is using simulated annealing to find the elite solutions inside big valleys so that tabu search can re-intensify search from the promising solutions. The effectiveness of this method is demonstrated by the 17 new upper bounds found in benchmark test instances. Finally, a very recent work on a hybrid simulated annealing algorithm has been conducted by El-Bouri et al [43]. More specifically, the authors developed a hybrid algorithm that apart from simulated annealing, it consists of a tabu search and adaptive memory programming framework. The proposed framework uses two distinct memories modules: the first memory temporarily prevents further changes in the configuration of a solution in an effort to maintain the presence of good solution elements while the second memory aims at tracking good solutions found during an iteration, so that the best ones can be used as the starting point in a subsequent iteration. The authors compared their algorithm with the traditional simulated annealing procedure and realized substantial performance improvements.

***Tabu Search*** (TS) is one of the most powerful meta-heuristics for the JSSP. This is affirmed by the fact that most state-of-the-art algorithms for the JSSP include some sort of tabu search functionality. The main strength

of tabu search is the use of memory that speeds up the solution space search process. Early implementations of tabu search in JSSP have been conducted by Taillard [153,156], Barnes and Chambers [13] and Sun et al [151]. One of the most powerful implementations was proposed by Nowicki and Smutnicki [119]. The computational effectiveness of their TSAB method is demonstrated by the fact that the notorious FT10 instance was solved in just 30 seconds using a PC of that era. A later work conducted by Watson et al [168] is based on Nowicki and Smutnicki's TSAB algorithm and aims at developing an understanding of why tabu search is so effective on JSSP. Later hybrid implementations include Pezzella and Merelli's [129] hybrid tabu search-shifting bottleneck procedure where the shifting bottleneck heuristic is initially used to generate schedules and then to refine solutions in subsequent iterations. Finally, as mentioned in the simulated annealing for JSSP section, a very recent hybrid tabu search - simulated annealing method was proposed by Zhang et al [180] which managed to find new upper bounds for many benchmark instances.

***Threshold Accepting*** algorithms have also been used for JSSP optimization but their application is rather limited. One of the first implementations was proposed by Aarts et al [1] in their computational study of local search algorithms for JSSP. However, poor results in comparison to other local search methods were recorded. A variant of the classic threshold accepting algorithm was proposed by Tarantilis and Kiranoudis [161]. The variant used is named 'List-Based Threshold Accepting method' (LBTA) and the authors demonstrated competitive results. A very important characteristic of this algorithm is that tuning of only one parameter is needed (namely the list size), thus making this implementation very easy to tune for a wide variety of problems. Based on the LBTA method of Tarantilis and Kiranoudis [161], Lee et al [92] proposed a new version and adopted the probabilistic steepest neighbor selection strategy to the original LBTA method to compromise between searching time and neighborhood quality, a strategy aiming at speeding up the search process.

## 1.6 The Open Shop Scheduling Problem

The open shop scheduling problem has attracted considerably less attention from academic researchers than flow shop and job shop scheduling problems, a fact outlined by the noticeably smaller number of papers in this field. Concerning exact methods for the OSSP, a polynomial time algorithm has been proposed by Gonzalez and Sahni [63] for the two-machine problem. Brucker et al [24] developed a branch and bound algorithm for the general $m$-machine problem based on the disjunctive graph representation of the problem. The method was very efficient and some benchmark instances were solved to optimality for the first time. Adiri and Aizikowitz [4] presented a linear time algorithm for the three-machine OSSP, provided that one machine dominates one of the other two. Algorithms for arbitrary $m$-machine problems with one

or two dominating machines are analyzed in Tanaev et al [160]. More recently, Kyparissis and Koulamas [89] presented a polynomial time algorithm for the two-machine OSSP with the objective of minimizing the total completion time subject to minimum makespan. They also extended their algorithm to the three-machine OSSP.

### 1.6.1 Heuristics for the OSSP

Röck and Schmidt [137] introduced a machine aggregation algorithm for the general $m$-machine OSSP based on the fact that the two-machine cases is polynomially solvable. Some more promising methods focus on 'dense' schedules where no machine is idle unless there is no operation available to be processed on that machine. Bräsel et al [20] developed some efficient constructive heuristic algorithms based on a structure analysis of the feasible combinations of job and machine orders. Guéret and Prins [67] presented list scheduling heuristics with two priorities for each operation, and matching heuristics which are followed by a local search improvement procedure. Finally, Liaw [100] introduced an iterative improvement approach based on a decomposition technique.

### 1.6.2 Meta-heuristics for the OSSP

Although there are quite a few exact and heuristic methods for the OSSP, the great difference between the OSSP and the JSSP/FSSP is observed in the number of meta-heuristic methodologies proposed. Liaw [101] proposed a robust tabu search algorithm for the OSSP. In this work, Liaw also developed a dispatching rule for initial solution generation. One year later, a hybrid genetic algorithm - tabu search algorithm (HGA) for the OSSP was proposed by the same author [102]. This work was based on the previous work of Liaw on tabu search for the OSSP and the author managed to improve the results of the first paper to the point that some benchmark instances were solved to optimality for the first time. Liaw also compared the results acquired by HGA to those obtained with list scheduling heuristic, insertion heuristic (IH), simulated annealing and pure TS algorithms and HGA outperformed them all. The same year, Prins [131] also proposed a genetic algorithm for the OSSP with slightly worse results than Liaw's implementation in Taillard's benchmark instances, except for some 20x20 problems. A more recent meta-heuristic implementation for the OSSP was proposed by Blum [18]. More specifically, Blum proposed a hybrid ant colony optimization with beam search algorithm which outperformed both genetic algorithm implementations of Liaw [102] and Prins [131], rendering the beam-ACO method a state-of-the-art method for OSSP.

## Conclusions

In this chapter, the main shop scheduling problems were presented along with a review of exact, heuristic and meta-heuristic methodologies used to solve

them. Exact methods fail to solve large instances of these problems in reasonable computational time while heuristic methods lack the robustness required, although they induce much less computational effort. Thus, it is not without reason that late research has mainly focused on meta-heuristic methods for these hard combinatorial optimization problems.

Concerning current and future research on this field, there are mainly two trends: the hybridization of single meta-heuristic components to form more powerful search methods and the use of advanced computational equipment (i.e. parallel processing). These two trends can be also unified. However, most importantly, future research should also concentrate on the applicability of any proposed method in real life scheduling problems. Considering the expansion of Enterprise Resource Planning (ERP) systems and their applicability in most production environments, the incorporation of advanced shop scheduling problem solving methods in ERP systems is of great importance for both academic research and industrial practice.

## References

1. 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 J Comput 6:118–125
2. 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
3. Adams, J., Balas, E., Zawack, D. (1988) The shifting bottleneck procedure for the job-shop scheduling. Manage Sci 34:391–401
4. Adiri, I., Aizikowitz, N. (1989) Open shop scheduling problems with dominated machines. Nav Res Logist Q 36:273–281
5. Applegate, D., Cook, W. (1991) A computational study of the job-shop scheduling problem. ORSA J Comput 3:149–156
6. Baker, K.R. (1975) A comparative survey of flowshop algorithms. Oper Res 23:62–73
7. Balas, E. (1969) Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. Oper Res17:941–957
8. Balas, E., Lenstra, J.K., Vazacopoulos, A. (1995) The one machine problem with delayed precedence constraints and its use in job shop scheduling. Manage Sci 41:94–109
9. Balas, E., Vazacopoulos, A. (1998) Guided local search with shifting bottleneck for job-shop scheduling. Manage Sci 44:262–275
10. Baptise, P., Le Pape, C., Nuijten, W. (1995a). Constrained based optimization and approximation for job-shop scheduling. In Proc. AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, 14th Int. Joint Conference on Artificial Intelligence, IJCAD, Montreal, Canada
11. Baptiste, P., Le Pape, C., Nuijten, W. (1995b) Incorporating efficient operations research algorithms in constraint-based scheduling. In Proc. 1st Joint Workshop on Artificial Intelligence and Operations Research, Timberline Lodge, OR

12. Barker, J.R., McMahon, G.B. (1985) Scheduling the general job-shop. Manage Sci 31:594–598
13. Barnes, J.W., Chambers, J.B. (1995) Solving the job shop scheduling problem using tabu search. IIE Trans 27:257–263
14. Bean, J.C. (1994) Genetic algorithm and random keys for sequencing and optimization, ORSA J Comput 6:154–160
15. Ben-Daya, M., Al-Fawzan, M. (1998) A tabu search approach for the flow shop scheduling problem. Eur J Oper Res 109:88–95
16. 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 310–318
17. Blazewicz, J., Domschke, W., Pesch, E. (1996) The job-shop scheduling problem: Conventional and new solution techniques. Eur J Oper Res 93:1–33
18. Blum, C. (2005) Beam-ACO-hybridizing ant colony optimization with beam search: an application to open shop scheduling. Comput Oper Res 32: 1565–1591
19. Blum, C., Roli, A. (2003) Meta-heuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Comput Surv 35:268–308
20. Bräsel, H., Tautenhan, T., Werner, F. (1993) Constructive Heuristic Algorithms for the Open Shop Problem. Computing 51:95–110
21. Brooks, G.H., White, C.R. (1965) An algorithm for finding optimal or near-optimal solutions to the production scheduling problem. J Ind Eng 16:34–40
22. Brucker, P. (1988) A polynomial algorithm for the two machine job-shop scheduling problem with a fixed number of jobs. OR Spektrum 16:5–7
23. Brucker, P. (1994) An efficient algorithm for the job-shop problem with two jobs. Computing 40:353–359
24. Brucker, P., Hurink, J., Jurish, B., Wostmann, B. (1997) A branch and bound algorithm for the open-shop problem, Discrete Appl Math 76: 43–59
25. Brucker, P., Jurisch, B., Sievers, B. (1994) A branch and bound algorithm for the job-shop scheduling problem. Discrete Appl Math 49:107–127
26. Campbell, H.G., Dudek, R.A., Smith, M.L. (1970) A heuristic algorithm for the n-job, m-machine problem. Manage Sci 16:B630–B637.
27. Carlier, J. (1982) The one-machine sequencing problem. Eur J Oper Res 11: 42–47
28. Carlier, J., Pinson, E. (1989) An algorithm for solving the job-shop problem. Manage Sci 35:164–176
29. Carlier, J., Pinson, E. (1994) Adjustments of heads and tails for the job-shop problem. Eur J Oper Res 78:146–161
30. Charalambous, O. (1991) Knowledge based job-shop scheduling. PhD thesis, University of Manchester Institute of Science and Technology, Manchester
31. Chen, C.L., Vempati, V.S., Aljaber, N. (1995) An application of genetic algorithms for flow shop problems. Eur J Oper Res 80:389–396
32. Cheng, R., Gen, M., Tsujimura, Y. (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms, part I: representation. Comput Ind Eng 30:983–997
33. Conway, R.W., Maxwell, W.L., Miller, L.W. (1967) Theory of scheduling, Addison-Wesley
34. Croce, F., Tadei, R., Volta, G. (1995) A genetic algorithm for the job shop problem. Comp Oper Res 22: 15–24

35. Dagli, C.H., Lammers, S., Vellanki, M. (1991) Intelligent scheduling in manufacturing using neural networks. J Neural Networ Comput Tech Design Applic 2:4–10
36. Dagli, S.H., Sittisathanchai, S. (1995) Genetic neuro-scheduler: A new approach for job shop scheduling. Int J Prod Econ 41:135–145
37. Daniels, R.L., Mazzola, J.B. (1994) Flow shop scheduling with resource flexibility. Oper Res 42:1174–1182
38. Dannenbring, D.G. (1977) An evaluation of flow shop sequencing heuristics. Manage Sci 23:1174–1182
39. Davis, L. (1985) Job-shop scheduling with genetic algorithm. In: Grefenstette J J (Ed.), Proceedings of the First International Conference on Genetic Algorithms and their Applications. Lawrence Erlbaum, Pittsburg, PA, USA, 136–140
40. Dorigo, M. (1992) Optimization, Learning and Natural Algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy
41. Dueck, G., Scheuer, T. (1990) Threshold accepting. A general purpose optimization algorithm appearing superior to simulated annealing, J Comput Phys 90:161–175
42. Eberhart, R., Kennedy, J. (1995) A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39–43
43. El-Bouri, A., Azizi, N., Zolfaghari, S. (2007) A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: The case of job shop scheduling. Eur J Oper Res 177:1894–1910
44. El-Bouri, A., Balakrishnan, Popplewell, N. (2005) A neural network to enhance local search in the permutation flowshop. Comput Ind Eng 49:182–196
45. Feo, T.A., Resende, M.G.C. (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
46. 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
47. Fisher, M.L. (1973) Optimal solution of scheduling problems using Lagrange multipliers: Part I. Oper Res 21:1114–1127
48. Florian, M., Trepant, P., McMahon, G. (1971) An implicit enumeration algorithm for the machine sequencing problem. Manage Sci 17:B782–B792
49. Fogel, L.J. (1962) Toward inductive inference automata. In Proceedings of the International Federation for Information Processing Congress. Munich, 395–399
50. Fonseca, D.J., Navaresse, D. (2002) Artificial neural networks for job shop simulation. Adv Eng Inform 16:241–246
51. 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 275–282
52. 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, USA 283–290
53. 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, USA 341–348
54. Gangadharan, R., Rajendran, C. (1994) A simulated annealing heuristic for scheduling in a flowshop with bicriteria. Comput Ind Eng 27:473–476

55. Garey, M.R.D., Johnson, D.S., Sethi, R. (1976) The complexity of flowshop and job shop scheduling. Math Oper Res 1:117–129
56. Giffler, B., Thompson, G.L. (1960) Algorithms for solving production scheduling problems. Oper Res 8:487–503
57. Glover, F. (1986) Future paths for integer programming and links to artificial intelligence, Comput Oper Res 13:533–549
58. Glover, F. (1997) Tabu search and adaptive memory programming - advances, applications and challenges. In: Barr, Helgason Kennington (Eds.), Advances in Meta-heuristics, Optimization and Stochastic Modelling Technologies, Kluwer Academic Publishers, Boston, MA 1–75
59. Glover, F., Greenberg, H.J. (1989) New approaches for heuristic search: A bilateral linkage with artificial intelligence. Eur J Oper Res 39:119–130
60. Glover, F., Laguna, M., Marti, R. (2000) Fundamentals of scatter search and path relinking. Control 29:653–684
61. Gonçalves, J.F., Mendes, J.J.d.M., Resende, M.G.C. (2005) A hybrid genetic algorithm for the job shop scheduling problem. Eur J Oper Res 167:77–95
62. Gonzalez, T., Sahni, S. (1976) Open shop scheduling to minimize finish time, J Assoc Comput Mach 23:665–679
63. Gonzalez, T., Sahni, S. (1978) Flowshop and Jobshop Schedules: Complexity and approximation, Oper Res 20:36–52
64. Grabowski, J., Wodecki, M. (2004) A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Comput Oper Res 31:1891–1909
65. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math 5:287–326
66. Greenberg, H.H. (1968) A branch and bound solution to the general scheduling problem. Oper Res 16:353–361
67. Guéret, C., Prins, C. (1998) Classical and new heuristics for the open-shop problem: A computational evaluation. Eur J Oper Res 107:306–314
68. Gupta, J.N.D. (1971) A functional heuristic algorithm for the flow-shop scheduling problem. Oper Res 22:39–47
69. Gupta, J., Hennig, K., Werner, F. (2002) Local search heuristics for two-stage flow shop problems with secondary criterion. Comput Oper Res 29:123–149
70. Hansen, P., Mladenović, N. (2001) Variable neighborhood search: Principles and Applications. Eur J Oper Res 130:449–467
71. Hasija, S., Rajendran, C. (2004) Scheduling in flowshops to minimize total tardiness of jobs. Int J Prod Res 42:2289–2301
72. Hefetz, N., Adiri, I. (1982) An efficient optimal algorithm for the two-machine, unit-time, jobshop, schedule-length, problem. Math Oper Res 7:354–360
73. Heinonen, J., Pettersson, F. (2007) Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. Appl Math Comput 187:989–998
74. Hejazi, S.R., Saghafian, S. (2005) Flowshop-scheduling problems with makespan criterion: a review. Int J Prod Res 43:2895–2929
75. Held, M., Karp, R.M. (1962) A Dynamic Programming Approach to Sequencing Problems. J Soc Ind and Appl Math 10:196–210
76. Ho, J.C., Chang, Y-L. (1991) A new heuristic for the n-job, m-machine flowshop problem. Eur J Oper Res 52:194–202

77. Holland, J.H. (1975) Adaption in natural and artificial systems. The University of Michigan Press, Ann Harbor, MI
78. Hopp, W.J., Spearman, M.L. (1996) Factory Physics: Foundations of Manufacturing Management, Irwin
79. Huang, K-L., Liao, C-J. (2007) Ant colony optimization combined with taboo search for the job shop scheduling problem. Comput Oper Res, in press
80. Hurkens C, http://www.win.tue.nl/whizzkids/1997, accessed April 2007
81. Ignall, E., Schrage, L. (1965) Application of the branch and bound technique to some flow-shop scheduling problems. Oper Res 11:400–412
82. Ishibuchi, H., Misaki, S., Tanaka, H. (1995) Modified simulated annealing algorithms for the flow shop sequencing problem. Eur J Oper Res 81:388–398
83. Jackson, J.R. (1956) An extension of Johnson's result on job lot scheduling. Int J Prod Res 36:1249–1272
84. Jain, A.S., Meeran, S. (1998) Job-shop scheduling using neural networks. Int J Prod Res 36:1249–1272
85. Jain, A.S., Meeran, S. (1999) Deterministic job-shop scheduling: Past, present and future. Eur J Oper Res 113:390–434
86. Johnson, S.M. (1954) Optimal two- and three-stage production schedules with set-up times included. Nav Res Logist Q 1:61–68
87. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983) Optimization by simulated annealing. Science 4598:671–680
88. Koulamas C (1998) A new constructive heuristic for the flowshop scheduling problem. Eur J Oper Res 105:66–71
89. Kyparisis, G.J., Koulamas, C. (2000) Open shop scheduling with makespan and total completion time criteria. Comput Oper Res 27:15–27
90. Lageweg, B.J., Lenstra, J.K., Rinnooy Kan, A.H.G. (1978) A general bounding scheme for the permutation flow-shop problem. Oper Res 26:53–67
91. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (1993) Sequencing and scheduling: algorithms and complexity. In: Graves S C, Rinnooy Kan A H G, Zipkin P H (Eds.). Handbooks in Operations Research and Management Science, 4, Logistics of Production and Inventory, Elsevier, Amsterdam
92. Lee, D.S., Vassiliadis, V.S., Park, J.M. (2004) A novel threshold accepting meta-heuristic for the job-shop scheduling problem. Comput Oper Res 31: 2199–2213
93. Lee, H.C., Dagli, C.H. (1997) A parallel genetic-neuro scheduler for job-shop scheduling problems. Int J Prod Econ 51:115–122
94. Lee, I., Shaw, M.J. (2000) A neural-net approach to real-time flow-shop sequencing. Comput Ind Eng 38:125–147
95. Lenstra, J.K., Rinnooy Kan, A.H.G. (1979) Computational complexity of discrete optimization problems. Ann Discrete Math 4:121–140
96. Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P. (1977) Complexity of machine scheduling problems. Ann Discrete Math 7:343–362
97. Leung, J.Y.-T. (2004) Handbook of scheduling, Chapman & Hall/CRC, Boca Raton 78
98. Lian, Z., Gu, X., Jiao, B. (2006) A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan. Appl Math and Computation 175:773–785
99. Lian, Z., Jiao, B., Gu, X. (2006) A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. Appl Math Comput 183:1008–1017

100. Liaw, C.-F. (1998) An iterative improvement approach for the nonpreemptive open shop scheduling problem. Eur J Oper Res 111:509–517
101. Liaw, C.-F. (1999) A tabu search algorithm for the open shop scheduling problem. Comput Oper Res, 26:109–126
102. Liaw, C.-F. (2000) A hybrid genetic algorithm for the open shop scheduling problem. Eur J Oper Res 124:28–42
103. Liu, S.Q., Ong, H.L., Ng, K.M. (2005) A fast tabu search algorithm for the group shop scheduling problem. Adv Eng Softw 36:533–539
104. Lomnicki, Z.A. (1965) A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem. Oper Res Q 16:89–100
105. Luh, P.B., Zhao, X., Wang, Y., Thakur, L.S. (2000) Lagrangian Relaxation Neural Networks for Job Shop Scheduling. IEEE T Robotic Autom 16:78–88
106. Manne, A.S. (1960) On the job shop scheduling problem. Oper Res 8:219–223
107. Martin, P., Shmoys, D. (1995) A new approach to computing optimal schedules for the job shop scheduling problem. Extended Abstract, Comell University, Ithaca
108. Masuda, T., Ishii, H., Nishida, T. (1985) The mixed shop scheduling problem. Discrete Appl Math, 11:175–86
109. 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
110. 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 15–25
111. McMahon, G.B., Florian, M. (1975) On scheduling with ready times and due dates to minimize maximum lateness. Oper Res 23:475–482
112. Merkle, D., Middendorf, M. (2000) An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In Proceedings of the EvoWorkshops, Springer, Berlin, 287–296
113. Moccellin, J.a.V. (1995) A new heuristic method for the permutation flow shop scheduling problem. J Oper Res Soc 46:883–886
114. Moccellin, J.a.V., dos Santos, M.O. (2000) An adaptive hybrid meta-heuristic for permutation flowshop scheduling. Control Cybern 29:761–771
115. Murata, T., Ishibuchi, H., Tanaka, H. (1996) Genetic algorithms for flowshop scheduling problems. Comput Ind Eng 30:1061–1071
116. Murovec, B., Šuhel, P. (2004) A repairing technique for the local search of the job-shop problem. Eur J Oper Res 153:220–238
117. 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, 474–479
118. Nawaz, M., Enscore, Jr E., Ham, I. (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega-Int Journal Manage S 11:91–95
119. Nowicki, E., Smutnicki, C. (1996) A fast taboo search algorithm for the job-shop problem. Manage Sci 42:797–813
120. Nowicki, E., Smutnicki, C. (1996) A fast tabu search algorithm for the permutation flow-shop problem. Eur J Oper Res 91:160–175

121. Nowicki, E., Smutnicki, C. (2006) Some aspects of scatter search in the flow-shop problem. Eur J Oper Res 169:654–666
122. Ogbu, F., Smith, D. (1990b) Simulated annealing for the permutation flowshop problem. OMEGA-Int J Manage S 19:64–67
123. Ogbu, F., Smith, D. (1990a) The application of the simulated annealing algorithms to the solution of the n=m=Cmax flowshop problem. Comput Oper Res 17:243–253
124. Onwubolu, G.C., Davendra, D. (2006) Scheduling flow-shops using differential evolution algorithm, Eur J Oper Res 171:674–692
125. Osman, I., Potts, C. (1989) Simulated annealing for permutation flow-shop scheduling. OMEGA-Int J Manage S 17:551–557
126. Palmer, D.S. (1965) Sequencing jobs through a multistage process in the minimum total time: a quick method of obtaining a near optimum. Oper Res Q 16:101–107
127. Park, B.J., Choi, H.R., Kim, H.S. (2003) A hybrid genetic algorithm for the job shop scheduling problems. Comput Ind Eng 45:597–613
128. Pesant, G., Gendreau, M. (1996) A view of local search in Constraint Programming. In Principles and Practice of Constraint Programming- CP'96. Lect Notes in Computer Science 1118:353–366, Springer-Verlag
129. Pezzella, F., Merelli, E. (2000) A tabu search method guided by shifting bottleneck for the job shop scheduling problem. Eur J Oper Res 120:297–310
130. Ponnambalam, S.G., Aravindan, P., Chandrasekaran, S. (2001) Constructive and improvement flow shop scheduling heuristics: An extensive evaluation. Prod Plan Control 12:335–344
131. Prins, C. (2000) Competitive genetic algorithms for the open-shop scheduling problem. Math Method Oper Res 52:389–411
132. Rajendran, C., Ziegler, H. (2004) Ant-colony algorithms for permutation flowshop-scheduling to minimize makespan/total flowtime of jobs. Eur J Oper Res 155:426–438
133. Rechenberg, I. (1973) Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog
134. Reeves, C.R. (1995) A genetic algorithm for flowshop sequencing. Comput Oper Res 22:5–13
135. Reeves, C., Yamada, T. (1998) Genetic algorithms, path relinking, and the flowshop sequencing problem. Evol Comput 6:45–60
136. Resende, M.G.C. (1997) A GRASP for job shop scheduling. INFORMS Spring Meeting, San Diego, CA
137. Röck, H., Schmidt, G. (1983) Machine aggregation heuristics in shop-scheduling. Math Oper Res 45:303–314
138. Roy, B., Sussmann, B. (1964) Les problèmes d'ordonnancement avec constraintes disjonctives. Note D.S. no. 9 bis, SEMA, Paris
139. Ruiz, R., Maroto, C. (2005) A comprehensive review and evaluation of permutation flowshop heuristics, Eur J Oper Res 165:479–494
140. Ruiz, R., Maroto, C., Alcaraz, J. (2006) Two newrobust genetic algorithms for the flowshop scheduling problem, Omega-Int Journal Manage S 34:461–476
141. Sabuncuoglu, I., Gurgun, B. (1996) A neural network model for scheduling problems. Eur J Oper Res 93:288–299
142. Sadeh, N., Nakakuki, Y. (1996) Focused simulated annealing search an application to job-shop scheduling. Ann Oper Res 63:77–103

143. Sha, D.Y., Hsu, C-Y. (2006) A hybrid particle swarm optimization for job shop scheduling problem. Comput Ind Eng 51:791–808
144. Sipper, D., Bulfin, R.L. (1997) Production Planning, Control and Integration, McGraw Hill
145. Solimanpur, M., Vrat, P., Shankar, R. (2004) A neuro-tabu search heuristic for the flow shop scheduling problem. Comput Oper Res 31:2151–2164
146. Storn, R., Price, K. (1997) Differential Evolution - A simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
147. Stützle, T. (1998a) An ant approach for the flow shop problem. In EUFIT'98, Aaeban, Germany, 1560–1564
148. Stützle, T. (1998b) Applying iterated local search to the permutation flow shop problem. Technical Report, AIDA-98-04, FG Intellektik, TU Darmstadt
149. Stützle, T. (1999) Iterated local search for the quadratic assignment problem. Technical report, aida-99-03, FG Intellektik, TU Darmstadt.
150. Suliman, S. (2000) A two-phase heuristic approach to the permutation flow-shop scheduling problem. Int J Prod Econ 64:143–152
151. Sun, D.K., Batta, R., Lin, L. (1995) Effective job-shop scheduling through active chain manipulation. Comput Oper Res 22:159–172
152. T'kindt, V., Monmarch, N., Tercinet, F., Laugt, D. (2002) An Ant Colony optimization algorithm to solve a (2) machine bicriteria flowshop-scheduling problem. Eur J Oper Res 142:250–257
153. Taillard, E. (1989) Parallel taboo search technique for the jobshop scheduling problem. Internal Research Report ORWP89/11, Départment de Mathématiques (DMA), École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
154. Taillard, E. (1990) Some efficient heuristic methods for the flow shop sequencing problem. Eur J Oper Res 47:65–74
155. Taillard, E. (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285
156. Taillard, E. (1994) Parallel taboo search techniques for the jobshop scheduling problem. ORSA J Comput 16:108-117
157. Taillard, E.D., Gambardella, L.M., Gendreau, M., Potvin, J.-Y. (2001) Adaptive memory programming: A unified view of meta-heuristics, Eur J Oper Res 135:1–16
158. Talbi, E-G. (2002) A Taxonomy of Hybrid Meta-heuristics. J Heuristics 8: 541–564
159. Tamaki, H., Nishikawa, Y. (1992) A paralleled genetic algorithm based on a neighbourhood model and its application to the job-shop scheduling. In: M?nner R, Manderick B (Eds.), Proceedings of the Second International Workshop on Parallel Problem Solving from Nature (PPSN'2), Brussels, Belgium, 573–582
160. Tanaev, V.S., Sotskov, Y.N., Strusevich, V.A. (1994) Scheduling Theory: Multi-Stage Systems, Kluwer Academic Publishers, Printed in Dordrecht
161. Tarantilis, C.D., Kiranoudis, C.T. (2002) A list-based threshold accepting method for the job-shop scheduling problems. Int J Prod Econ 77:159–171
162. Tasgetiren, M.F., Sevkli, M., Liang, Y.C., Gencyilmaz, G. (2004) Differential Evolution Algorithm for Permutation Flowshop Sequencing Problem with Makespan Criterion. In Proceedings of the 4th International Symposium on Intelligent Manufacturing Systems (IMS 2004) Sakarya, Turkey, 442–452

163. Tasgetiren, M.F., Liang, Y,C., Sevkli, M., Gencyilmaz, G. (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 177: 1930–1947
164. Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K. (1988) Job-shop scheduling by simulated annealing. Report OSR8809. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands
165. Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K. (1992) Jobshop scheduling by simulated annealing. Oper Res 40: 113-125
166. Wang, L., Zheng, D.Z. (2001) An effective hybrid optimization strategy for job-shop scheduling problems. Comput Oper Res 28:585–596
167. Wang, L., Zheng, D.Z. (2003) An Effective Hybrid Heuristic for Flow Shop Scheduling. The Int J Adv Manuf Tech 21:38–44
168. Watson, J-P., Howe, A.E., Whitley, L.B. (2006) Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. Comput Oper Res 33:2623–2644
169. Widmer, M., Hertz, A. (1989) A new heuristic method for the flow shop sequencing problem. Eur J Oper Res 41:186–193
170. Wodecki, M., Bożejko, W. (2002) Solving the flow shop problem by parallel simulated annealing. In: Wyrzykowski R, Dongarra J, Paprzycki M, Waàsniewski J (Eds.), Parallel Processing and Applied Mathematics, 4th International Conference, PPAM 2001. In: Lect Notes Comput Sc 2328:236–244 Springer-Verlag, Berlin
171. Xia, W-j., Wu, Z-m. (2006) A hybrid particle swarm optimization approach for the job-shop scheduling problem. Int J Adv Manuf Tech 29:360–366
172. Yamada, T., Nakano, R. (1992) A genetic algorithm applicable to large-scale job-shop problems. In: M?nner R, Manderick B (Eds.), Proceedings of the Second International Workshop on Parallel Problem Solving from Nature (PPSN'2), Brussels, Belgium 281–290
173. Yamada, T., Nakano, R. (1995) Job-shop scheduling by simulated annealing combined with deterministic local search. In: Meta-heuristics International Conference (MIC'95), Hilton, Breckenridge, Colorado, USA 344–349
174. Yamada, T., Nakano, R. (1996) Job-shop scheduling by simulated annealing combined with deterministic local search. Meta-heuristics: Theory and Applications. Kluwer Academic Publishers, Hingham, MA 237–248
175. 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 4687–4692
176. Yang, S., Wangm D, (2001) A new adaptive neural network and heuristics hybrid approach for job-shop scheduling. Comput Oper Res 28:955–971
177. Ying, K.C., Liao, C.J. (2004) An ant colony system for permutation flow-shop sequencing. Comput Oper Res 31:791–801
178. Yu, H., Liang, W. (2001) Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. Comput Ind Eng 39:337–356
179. Zegordi, S.H., Itoh, K., Enkawa, T. (1995) Minimizing makespan for flowshop scheduling by combining simulated annealing with sequencing knowledge. Eur J Oper Res 85:515–531
180. Zhang, C., Li, P., Rao, Y., Guan, Z. (2007) A very fast TS/SA algorithm for the job shop scheduling problem. Comput Oper Res, in press

181. Zhao, F., Zhang, Q. (2006) An Improved Particle Swarm Optimization-Based Approach for Production Scheduling Problems. In Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation June 25 - 28, 2006, Luoyang, China
182. 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 889–894
183. Zhou, D.N., Cherkassky, V., Baldwin, T.R., Olson, D.E. (1991) A neural network approach to job-shop scheduling. IEEE T Neural Networ 2: 175–179
184. Zhou, H., Feng, H., Han, L. (2001) The hybrid heuristic genetic algorithm for job shop scheduling. Comput Ind Eng 40:191–200
185. Zobolas, G., Tarantilis, C.D., Ioannou, G. (2007) A Hybrid Evolutionary Algorithm for the Job Shop Scheduling Problem. J Oper Res Soc, doi: 10.1057/palgrave.jors.2602534