Chapter 5

Production Scheduling Approaches for Operations Management

Marcello Fera, Fabio Fruggiero, Alfredo Lambiase,
Giada Martino and Maria Elena Nenni

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/55431

1. Introduction

Scheduling is essentially the short-term execution plan of a production planning model. Production scheduling consists of the activities performed in a manufacturing company in order to manage and control the execution of a production process. A schedule is an assignment problem that describes into details (in terms of minutes or seconds) which activities must be performed and how the factory's resources should be utilized to satisfy the plan. Detailed scheduling is essentially the problem of allocating machines to competing jobs over time, subject to the constraints. Each work center can process one job at a time and each machine can handle at most one task at a time. A scheduling problem, typically, assumes a fixed number of jobs and each job has its own parameters (i.e., tasks, the necessary sequential constraints, the time estimates for each operation and the required resources, no cancellations). All scheduling approaches require some estimate of how long it takes to perform the work. Scheduling affects, and is affected by, the shop floor organization. All scheduling changes can be projected over time enabling the identification and analysis of starting time, completion times, idle time of resources, lateness, etc….

A right scheduling plan can drive the forecast to anticipate completion date for each released part and to provide data for deciding what to work on next. Questions about "Can we do it?" and/or "How are we doing?" presume the existence of approaches for optimisation. The aim of a scheduling study is, in general, to perform the tasks in order to comply with priority rules and to respond to strategy. An optimal short-term production planning model aims at gaining time and saving opportunities. It starts from the execution orders and it tries to allocate, in the best possible way, the production of the different items to the facilities. A good schedule starts from planning and springs from respecting resource conflicts, managing the release of jobs to

INTECH
open science | open minds

© 2013 Fera et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

a shop and optimizing completion time of all jobs. It defines the starting time of each task and determines whatever and how delivery promises can be met. The minimization of one or more objectives has to be accomplished (e.g., the number of jobs that are shipped late, the minimization set up costs, the maximum completion time of jobs, maximization of throughput, etc.). Criteria could be ranked from applying simple rules to determine which job has to be processed next at which work-centre (i.e., dispatching) or to the use of advanced optimizing methods that try to maximize the performance of the given environment. Fortunately many of these objectives are mutually supportive (e.g., reducing manufacturing lead time reduces work in process and increases probability to meeting due dates). To identify the exact sequence among a plethora of possible combinations, the final schedule needs to apply rules in order to quantify urgency of each order (e.g., assigned order's due date - defined as global exploited strategy; amount of processing that each order requires - generally the basis of a local visibility strategy). It's up to operations management to optimize the use of limited resources. Rules combined into *heuristic*[1] approaches and, more in general, in upper level multi-objective methodologies (i.e., *meta-heuristics*[2]), become the only methods for scheduling when dimension and/or complexity of the problem is outstanding [1]. In the past few years, metaheuristics have received much attention from the hard optimization community as a powerful tool, since they have been demonstrating very promising results from experimentation and practices in many engineering areas. Therefore, many recent researches on scheduling problems focused on these techniques. Mathematical analyses of metaheuristics have been presented in literature [2, 3].

This research examines the main characteristics of the most promising meta-heuristic approaches for the general process of a Job Shop Scheduling Problems (i.e., JSSP). Being a NP complete and highly constrained problem, the resolution of the JSSP is recognized as a key point for the factory optimization process [4]. The chapter examines the soundness and key contributions of the 7 meta-heuristics (i.e., Genetics Approaches, Ants Colony Optimization, Bees Algorithm, Electromagnetic Like Algorithm, Simulating Annealing, Tabu Search and Neural Networks), those that improved the production scheduling vision. It reviews their accomplishments and it discusses the perspectives of each meta approach. The work represents a practitioner guide to the implementation of these meta-heuristics in scheduling job shop processes. It focuses on the logic, the parameters, representation schemata and operators they need.

2. The job shop scheduling problem

The two key problems in production scheduling are „priorities" and „capacity". Wight (1974) described s*cheduling* as „establishing the timing for performing a task" and observes that, in

1 The etymology of the word heuristic derives from a Greek word *heurìsco (εὑρισκω)* - it means „to find"- and is considered the art of discovering new strategy rules to solve problems. Heuristics aims at a solution that is „good enough" in a computing time that is „small enough".

2 The term metaheuristc originates from union of prefix *meta (μετα)* - it means „behind, in the sense upper level methodology" – and word *heuristic* - it means „to find". Metaheuristcs' search methods can be defined as upper level general methodologies guiding strategies in designing heuristics to obtain optimisation in problems.

http://dx.doi.org/10.5772/55431

manufacturing firms, there are multiple types of scheduling, including the detailed scheduling of a shop order that shows when each operation must start and be completed [5]. Baker (1974) defined scheduling as „a plan than usually tells us when things are supposed to happen" [6]. Cox *et al.* (1992) defined *detailed scheduling* as „the actual assignment of starting and/or completion dates to operations or groups of operations to show when these must be done if the manufacturing order is to be completed on time"[7]. Pinedo (1995) listed a number of important surveys on production scheduling [8]. For Hopp and Spearman (1996) „scheduling is the allocation of shared resources over time to competing activities" [9]. Makowitz and Wein (2001) classified production scheduling problems based on attributes: the presence of setups, the presence of due dates, the type of products.

Practical scheduling problems, although more highly constrained, are high difficult to solve due to the number and variety of jobs, tasks and potentially conflicting goals. Recently, a lot of Advanced Production Scheduling tools arose into the market (e.g., Aspen PlantTM Scheduler family, Asprova, R2T – Resourse To Time, DS APS – DemandSolutions APS, DMS – Dynafact Manufacturing System, i68Group, ICRON-APS, JobPack, iFRP, Infor SCM, SchedulePro, Optiflow-Le, Production One APS, MQM – Machine Queue Management, MOM4, JDA software, Rob-ex, Schedlyzer, OMP Plus, MLS and MLP, Oracle Advanced Scheduling, Ortec Schedule, ORTEMS Productionscheduler, Outperform, AIMMS, Planet Together, Preactor, Quintiq, FactoryTalk Scheduler, SAP APO-PP/DS, and others). Each of these automatically reports graphs. Their goal is to drive the scheduling for assigned manufacturing processes. They implement rules and optimise an isolated sub-problem but none of the them will optimise a multi stage resource assignment and sequencing problem.

In a Job Shop (i.e., JS) problem a classic and most general factory environment, different tasks or operations must be performed to complete a job [10]; moreover, priorities and capacity problems are faced for different jobs, multiple tasks and different routes. In this contest, each job has its own individual flow pattern through assigned machines, each machine can process only one operation at a time and each operation can be processed by only one machine at a time. The purpose of the procedure is to obtain a schedule which aims to complete all jobs and, at the same time, to minimize (or maximize) the objective function. Mathematically, the JS Scheduling Problem (i.e., JSSP) can be characterized as a combinatorial optimization problem. It has been generally shown to be NP-hard[3] belonging to the most intractable problems considered [4, 11, 12]. This means that the computation effort may grow too fast and there are not universal methods making it possible to solve all the cases effectively. Just to understand what the technical term means, consider the single-machine sequencing problem with three jobs. How many ways of sequencing three jobs do exist? Only one of the three jobs could be in the first position, which leaves two candidates for the second position and only one for the last position. Therefore the no. of permutations is 3!. Thus, if we want to optimize, we need to consider six alternatives. This means that as the no. of jobs to be sequenced becomes larger (i.e., n>80), the no. of possible sequences become quite ominous and an exponential function dominates the amount of time required to find the optimal solution [13]. Scheduling, however,

3 A problem is NP-complete if exists no algorithm that solves the problem in a polynomial time. A problem is NP-hard if it is possible to show that it can solve a NP-complete problem.

performs the definition of the optimal sequence of n jobs in m machines. If a set of n jobs is to be scheduled on m machines, there are $(n!)^m$ possible ways to schedule the job.

It has to undergo a discrete number of operations (i.e., *tasks*) on different resources (i.e., *machines*). Each product has a fixed route defined in the planning phase and following processing requirements (i.e., precedence constraints). Other constraints, e.g. zoning which binds the assignment of task to fixed resource, are also taken into consideration. Each machine can process only one operation at a time with no interruptions (pre-emption). The schedule we must derive aims to complete all jobs with minimization (maximization) of an objective function on the given production plant.

Let:

- $J = \{J_1, J_2, \ldots\ldots, J_n\}$ the set of the job order existing inside the system;
- $M = \{M_1, M_2, \ldots\ldots, M_m\}$ the set of machines that make up the system.

JSSP, marked as Π_j, consists in a finite set J of n jobs $\{J_i\}_{i=1}^{n}$. Each $J_{i\,\text{is}}$ characterized by a manufacturing cycle CL_i regarded as a finite set M of m machines $\{M_k\}_{k=1}^{m}$ with an uninterrupted processing time τ_{ik}. J_i, $\forall\ i=1, \ldots, n$, is processed on a fixed machine m_i and requires a chain of tasks $O_{i1}, O_{i2}, \ldots\ldots, O_{im_i}$, scheduled under precedence constraints. O_{ik} is the task of job J_i which has to be processed on machine M_k for an uninterrupted processing time period τ_{ik} and no operations may pre-empted.

To accommodate extreme variability in different parts of a job shop, schedulers separate workloads in each work-centres rather than aggregating them [14]. Of more than 100 different rules proposed by researchers and applied by practitioners exist, some have become common in Operations Management systems: First come- First served, Shortest Processing Time, Earliest Due Date, Slack Time Remaining, Slack Time Remaining For each Operation, Critical Ratio, Operation Due Date, etc. [15]. Besides these, Makespan is often the performance feature in the study of resource allocation [16]. Makespan represents the time elapsed from the start of the first task to the end of the last task in schedule. The minimisation of makespan arranges tasks in order to level the differences between the completion time of each work phase. It tries to smooth picks in work-centre occupancy to obtain batching in load assignment per time. Although direct time constraints, such as minimization of processing time or earliest due date, are sufficient to optimize industrial scheduling problems, for the reasons as above the minimization of the makespan is preferable for general/global optimization performances because it enhances the overall efficiency in shop floor and reduces manufacturing lead time variability [17].

Thus, in JSSP optimization variant of Π_j, the objective of a scheduling problem is typically to assign the tasks to time intervals in order to minimise the makespan and referred to as:

$$C^*_{\max}(t) = f(CL_i, \tau_{ik}, s_{ik}), \forall\ i = 1...n; \forall\ k = 1...m \tag{1}$$

http://dx.doi.org/10.5772/55431

where t represent time (i.e. iteration steps)

$$C^{*}_{\max}(t) = \min(C_{\max}(t)) = \min\{\max_i[\tau_{ik} + s_{ik}] : \forall J_i \in J, \forall M_k \in M\} \quad (2)$$

and $s_{ik} \geq 0$ represents the starting time of k-th operation of i-th job. s_{ik} is the time value that we would like to determinate in order to establish the suited schedule activities order.

3. Representation of scheduling instances

The possible representation of a JS problem could be done through a Gantt chart or through a Network representation.

Gantt (1916) created innovative charts for visualizing planned and actual production [18]. According to Cox *et al*. (1992), a *Gantt chart* is „the earliest and best known type of control chart especially designed to show graphically the relationship between planned performance and actual performance" [19]. Gantt designed his charts so that foremen or other supervisors could quickly know whether production was on schedule, ahead of schedule or behind schedule. A Gantt chart, or bar chart as it is usually named, measures activities by the amount of time needed to complete them and use the space on the chart to represent the amount of the activity that should have been done in that time [7].

A Network representation was first introduced by Roy and Sussman [20]. The representation is based on "*disjunctive graph model*" [21]. This representation starts from the concept that a feasible and optimal solution of JSP can originate from a permutation of task's order. Tasks are defined in a network representation through a probabilistic model, observing the precedence constraints, characterized in a machine occupation matrix $\boldsymbol{M}$ and considering the processing time of each tasks, defined in a time occupation matrix $\boldsymbol{T}$.

$$M = \begin{pmatrix} M_{11} & \dots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{pmatrix}; \; T = \begin{pmatrix} \tau(M_{11}) & \dots & \tau(M_{1n}) \\ \vdots & \ddots & \vdots \\ \tau(M_{n1}) & \cdots & \tau(M_{nn}) \end{pmatrix}$$

JS processes are mathematically described as disjunctive graph $G = (V, C, E)$. The descriptions and notations as follow are due to Adams et. al. [22], where:

- V is a set of nodes representing tasks of jobs. Two additional dummy tasks are to be considered: a *source(0)* node and a *sink(*)* node which stand respectively for the Source (S) task $\tau_0 = 0$, necessary to specify which job will be scheduled first, and an end fixed sink where schedule ends (T) $\tau_* = 0$;
- C is the set of conjunctive arcs or direct arcs that connect two consecutive tasks belonging to the same job chain. These represent technological sequences of machines for each job;

- $E=\bigcup_{r=1}^{m} D_r$, where D_r is a set of disjunctive arcs or not-direct arcs representing pair of operations that must be performed on the same machine M_r.

Each job-tasks pair *(i,j)* is to be processed on a specified machine *M(i,j)* for *T(i,j)* time units, so each node of graph is weighted with *j* operation's processing time. In this representation all nodes are weighted with exception of source and sink node. This procedure makes always available feasible schedules which don't violate hard constraints[4]. A graph representation of a simple instance of JSP, consisting of 9 operations partitioned into 3 jobs and 3 machines, is presented in fig. 1. Here the nodes correspond to operations numbered with consecutive ordinal values adding two fictitious additional ones:*S* = "source node" and *T* = "sink node". The processing time for each operation is the weighted value τ_{ij} attached to the corresponding node,$v \in V$, and for the special nodes, $\tau_0 = \tau_* = 0$.

Let s_v be the starting time of an operation to a node *v*. By using the disjunctive graph notation, the JSPP can be formulated as a mathematical programming model as follows:

Minimize s_* subject to:

$$s_w - s_v \geq \tau_v \ (v, w) \in C \tag{3}$$

$$s_v \geq 0 v \in V \tag{4}$$

$$s_w - s_v \geq \tau_v \vee s_v - s_w \geq \tau_w (v, w) \in D_{r,1} \leq r \leq m, \tag{5}$$

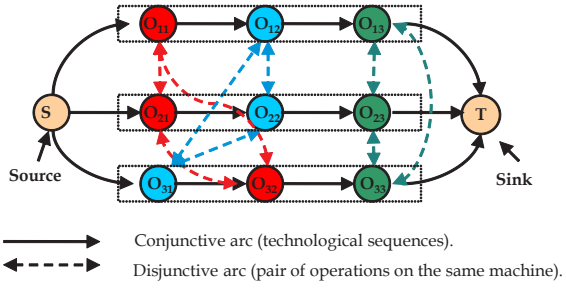


Figure 1. Disjunctive graph representation. There are disjunctive arcs between every pair of tasks that has to be processed on the same machine (dashed lines) and conjunctive arcs between every pair of tasks that are in the same job (dotted lines). Omitting processing time, the problem specification is $O=\{o_{ij}, (i, j) \in \{1, 2, 3\}^2\}$, $J=\{J_i=\{o_{ij}\}, (i, j)=1, 2, 3\}$, $M=\{M_j=\{o_{ij}\}, (i, j)=1, 2, 3\}$. Job notation is used.

4 Hard constraints are physical ones, while soft constraints are generally those related to human factor e.g., relaxation, fatigue etc…

http://dx.doi.org/10.5772/55431

s_* is equal to the completion time of the last operation of the schedule, which is therefore equal to C_{max}. The first inequality ensures that when there is a conjunctive arc from a node v to a node w, w must wait of least τ_v time after v is started, so that the predefined technological constraints about sequence of machines for each job is not violated. The second condition ensures time to start continuities. The third condition affirms that, when there is a disjunctive arc between a node v and a node w, one has to select either v to be processed prior to w (and w waits for at least τ_v time period) or the other way around, this avoids overlap in time due to contemporaneous operations on the same machine.

In order to obtain a scheduling solution and to evaluate makespan, we have to collect all feasible permutations of tasks to transform the undirected arcs in directed ones in such a way that there are no cycles.

The total number of nodes, $n=(|O|+2)$ - fixed by taking into account the total number of tasks $|O|$, is properly the total number of operations with more two fictitious ones. While the total number of arcs, in job notation, is fixed considering the number of tasks and jobs of instance:

$$\text{arcs}=\binom{n}{2}+2\times|J| \quad =\frac{(|O|)\times\{(|O|)-1\}}{2}+2\times|J| \tag{6}$$

The number of arcs defines the possible combination paths. Each path from source to sink is a candidate solution for JSSP. The routing graph is reported in figure 2:

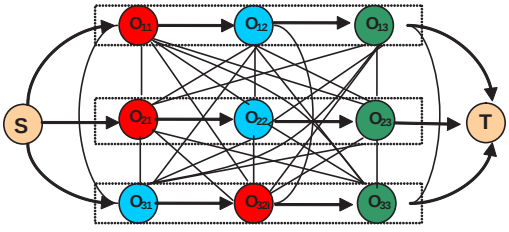


Figure 2. Problem routing representation.

4. Meta-heuristics for solving the JSSP

A logic has to be implemented in order to translate the scheduling problem into an algorithm structure. Academic researches on scheduling problems have produced countless papers [23]. Scheduling has been faced from many perspectives, using formulations and tools of various disciplines such as control theory, physical science and artificial intelligence systems [24]. Criteria for optimization could be ranked from applying simple priority rules to determine

which job has to be processed next at the work-centres (i.e., dispatching) to the use of advanced optimizing methods that try to maximize the performance of the given environment [25]. Their way to solution is generally approximate – heuristics – but it constitutes promising alternatives to the exact methods and becomes the only one possible when dimension and/or complexity of the problem is outstanding [26].

Guidelines in using heuristics in combinatorial optimization can be found in Hertz (2003) [27]. A classification of heuristic methods was proposed by Zanakis et al. (1989) [28]. Heuristics are generally classified into *constructive heuristics* and *improvement heuristics*. The first ones are focused on producing a solution based on an initial proposal, the goal is to decrease the solution until all the jobs are assigned to a machine, not considering the size of the problem [29]. The second ones are iterative algorithms which explore solutions by moving step by step form one solution to another. The method starts with an arbitrary solution and transits from one solution to another according to a series of basic modifications defined on case by case basis [30].

Relatively simple rules in guiding heuristic, with exploitation and exploration, are capable to produce better quality solutions than other algorithms from the literature for some classes of instances. These variants originate the class of meta-heuristic approaches [31]. The meta-heuristics[5], and in general the heuristics, do not ensure optimal results but they usually tend to work well [32]. The purpose of the paper is to illustrate the most promising optimization methods for the JSSP.

As optimization techniques, metaheuristics are stochastic algorithms aiming to solve a broad range of hard optimization problems, for which one does not know more effective traditional methods. Often inspired by analogies with reality, such as physics science, Simulated Annealing [33] and Electromagnetic like Methods [34], biology (Genetic Algorithms [35], Tabu Search [36]) and ethnology (Ant Colony [37,], Bees Algorithm [38]), human science (Neural Networks [39]), they are generally of discrete origin but can be adapted to the other types of problems.

4.1. Genetic Algorithms (GAs)

The methodology of a GAs - based on the evolutionary strategy- trasforms a population (set) of individual objects, each with an associated *fitness* value, into a new *generation* of the population occurring genetic operations such as *crossover* (*sexual recombination*) and *mutation* (fig. 3).

The theory of evolutionary computing was formalized by Holland in 1975 [40]. GAs are stochastic search procedures for combinatorial optimization problems based on Darwinian principle of natural reproduction, survival and environment's adaptability [41]. The theory of evolution is biologically explained, the individuals with a stronger fitness are considered better able to survive.. Cells, with one or more strings of DNA (i.e., a chromosome), make up an individual. The gene (i.e., a bit of chromosome located into its particular locus) is, responsible for encoding traits (i.e., alleles). Physical manifestations are raised into genotype (i.e., disposition of genes). Each genotype has is physical manifestation into phenotype. According to these parameters is possible to define a fitness value. Combining individuals through a

5 The term metaheuristics was introduced by F. Glover in the paper about Tabu search.

http://dx.doi.org/10.5772/55431

crossover (i.e., recombination of genetic characteristics of parents) across the sexual reproduction, the chromosomal inheritance process performs to offspring. In each epoch a stochastic mutation procedure occurs. The implemented algorithm is able to simulate the natural process of evolution, coupling solution of scheduling route in order to determinate an optimal tasks assignment. Generally, GA has different basic component: representation, initial population, evaluation function, the reproduction selection scheme, genetic operators (mutation and crossover) and stopping criteria. Central to success of any GA is the suitability of its representation to the problem at hand [42]. This is the encoding from the solution of the problem domain to the genetic representation.

During the last decades, different representation's schemata for JS have been proposed, such as *permutation with repetition*. It uses sequence of repeated jobs identifier (e.g., its corresponding cardinal number) to represent solutions [43]. According to the instance in issue, each of the N jobs identifiers will be repeated M times, once for each task. The first time that job's identifier, reading from left to right, will appear means the first task of that job. In this way, precedence constraints are satisfied. The redundancy is the most common caveat of this representation. A proposal of permutation with repetition applying a Generalized Order crossover (GOX) with band |2 3 1 1| of parent 1 moves from PARENT1 [3 2 3 1 1 1 3 2 2] and PARENT2 [2 3 2 1 3 3 2 1 1] to CHILD1 [2 3 1 1 3 2 3 2 1] and CHILD2 [3 2 1 3 2 1 1 3 2].

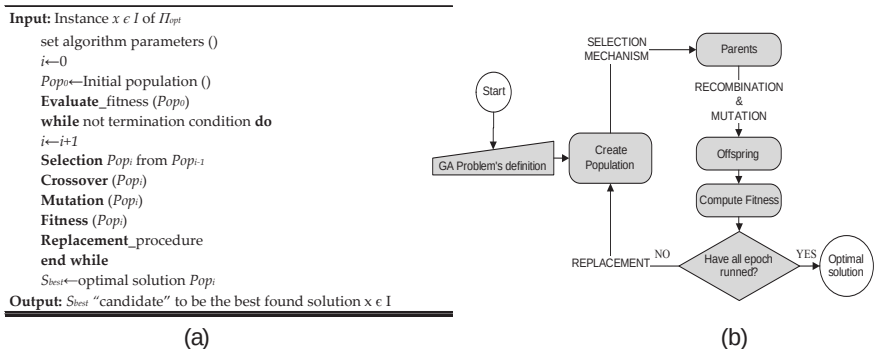


Figure 3. The Genetic Algorithms (GAs) model; 3a. the pseudo-code of a GA; 3b. the flow chart of a general GA.

A mutation operator is applied changing the genes into the same genotype (in order to generate only feasible solutions, i.e., without the rejection procedure). Mutation allows to diversify the search over a broader solution domain and it is needed when there is low level of crossover. Among solutions, the allocation with favourable fitness will have higher probability to be selected through the selection mechanisms.

Another important issue for the GA is the selection mechanism (e.g., Tournament Selection procedure and Roulette Wheel as commonly used [44] - their performances are quite similar attending in the convergence time). The *tournament selection* procedure is based on analogy with competition field, between the genotypes in tournament, the individual which will win

(e.g., the one with the best fitness value) is placed in the mating pool. Likewise, in the *roulette wheel selection* mechanism each individual of population has a selection's likelihood proportional to its objective score (in analogy with the real roulette item) and with a probability equal to one of a ball in a roulette, one of the solutions is chosen.

It is very important, for the GAs success, to select the correct ratio between crossover and mutation, because the first one allows to allows to diversify a search field, while a mutation to modify a solution.

4.2. Ant Colony Optimization (ACO) algorithms

If we are on a pic-nic and peer into our cake bitten by a colony of ants, moving in a tidy way and caring on a lay-out that is the optimal one in view of stumbling-blocks and length, we discover how remarkable is nature and we find its evolution as the inspiring source for investigations on intelligence operation scheduling techniques [45]. Natural ants are capable to establish the shortest route path from their colony to feeding sources, relying on the phenomena of *swarm intelligence* for survival. They make decisions that seemingly require an high degree of co-operation, smelling and following a chemical substance (i.e. pheromone[6]) laid on the ground and proportional to goodness load that they carry on (i.e. in a scheduling approach, the goodness of the objective function, reported to makespan in this applicative case).

The same behaviour of natural ants can be overcome in an artificial system with an artificial communication strategy regard as a direct metaphoric representation of natural evolution. The essential idea of an ACO model is that „good solutions are not the result of a sporadic good approach to the problem but the incremental output of good partial solutions item. Artificial ants are quite different by their natural progenitors, maintaining a memory of the step before the last one [37]. Computationally, ACO [46] are population based approach built on stochastic solution construction procedures with a retroactive control improvement, that build solution route with a probabilistic approach and through a suitable selection procedure by taking into account: *(a) heuristic information* on the problem instance being solved; *(b)* (mat-made) *pheromone amount*, different from ant to ant, which stores up and evaporates dynamically at run-time to reflect the agents' acquired search training and elapsed time factor.

The initial schedule is constructed by taking into account heuristic information, initial pheromone setting and, if several routes are applicable, a self-created selection procedure chooses the task to process. The same process is followed during the whole run time. The probabilistic approach focused on pheromone. Path's attractive raises with path choice and probability increases with the number of times that the same path was chosen before [47]. At the same time, the employment of heuristic information can guide the ants towards the most promising solutions and additionally, the use of an agent's colony can give the algorithm: *(i)* Robustness on a fixed solution; *(ii)* Flexibility between different paths.

The approach focuses on co-operative ant colony food retrieval applied to scheduling routing problems. Colorni et al, basing on studies of Dorigo *et al.* [48], were the first to apply Ant System

6 It is an organic compound highly volatile that shares on central neural system as an actions' releaser.

http://dx.doi.org/10.5772/55431

(AS) to job scheduling problem [49] and dubbed this approach as **A**nt **C**olony **O**ptimization (ACO). They iteratively create route, adding components to partial solution, by taking into account heuristic information on the problem instance being solved (i.e. visibility) and "artificial" pheromone trials (with its storing and evaporation criteria). Across the representation of scheduling problem like acyclic graph, see fig. 2, the ant's rooting from source to food is assimilated to the scheduling sequence. Think at ants as agents, nodes like tasks and arcs as the release of production order. According to constraints, the ants perform a path from the row material warehouse to the final products one.

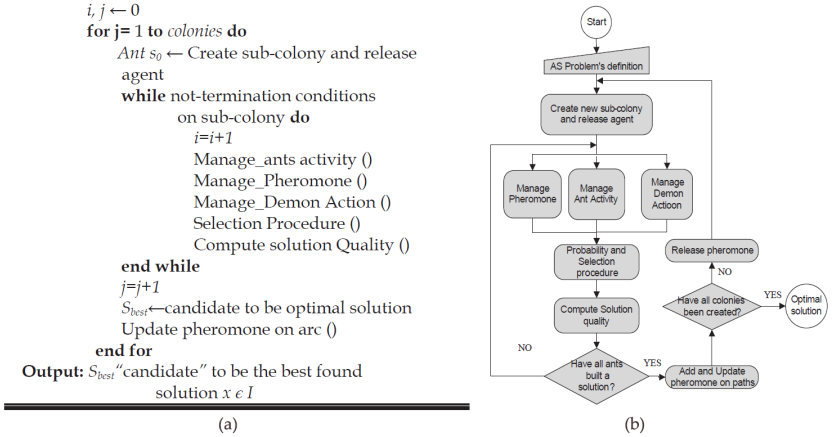


Figure 4. The Ant Colony Optimization (ACO) model; 4a. the pseudo-code of an ACO algorithm; 4b. the flow chart of a general ACO procedure.

Constraints are introduced hanging from jobs and resources. Fitness is introduced to translate how good the explored route was. Artificial ants live in a computer realized world. They have an overview of the problem instance they are going to solve across a visibility factor. In the Job Shop side of ACO implementation the visibility has chosen tied with the run time of the task (Eq. 7). The information was about the inquired task's (i.e., *j*) completion time $Ctime_j$ and idle time $Itime_j$ from the previous position (i.e., *i*):

$$\eta_{ij}(t) = \frac{1}{(Ctime_j - Itime_j)} = \frac{1}{Rtime_j} \tag{7}$$

The colony is composed of a fixed number of agents *ant=1,…, n*. A probability is associated to each feasible movement ($S_{ant}(t)$) and a selection procedure (generally based on *RWS* or *Tournament* procedure) is applied.

$0 \leq P_{ij}^{ant}(t) \leq 1$ is the probability that at time t the generic agent *ant* chooses edge $i \rightarrow j$ as next routing path; at time t each *ant* chooses the next operation where it will be at time $t+1$. This value is valuated through visibility (η) and pheromone (τ) information. The probability value (Eq. 8) is associated to a fitness into selection step.

$$P_{ij}^{ant}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{h \in S_{ant}(t)} [\tau_{ih}(t)]^{\alpha}[\eta_{ih}(t)]^{\beta}_{i}} & \text{if } j \in S_{ant}(t) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Where: $\tau_{ij}(t)$ represents the intensity of trail on connection (i, j) at time t. Set the intensity of pheromone at iteration $t=0$: $\tau_{ij}(o)$ to a general small positive constant in order to ensure the avoiding of local optimal solution; α and β are user's defined values tuning the relative importance of the pheromone vs. the heuristic time-distance coefficient. They have to be chosen $0<\alpha, \beta \leq 10$ (in order to assure a right selection pressure).

For each cycle the agents of the colony are going out of source in search of food. When all colony agents have constructed a complete path, i.e. the sequence of feasible order of visited nodes, a pheromone update rule is applied (Eq. 9):

$$\tau_{ij}(t+1) = (1-\lambda)\tau_{ij}(t+1) + \Delta\tau_{ij}(t) \quad (9)$$

Besides ants' activity, *pheromone trail evaporation* has been included trough a coefficient representing pheromone vanishing during elapsing time. These parameters imitate the natural world decreasing of pheromone trail intensity over time. It implements a useful form of forgetting. It has been considered a simple decay coefficient (i.e., $0<\lambda<1$) that works on total laid pheromone level between time t and $t+1$.

The laid pheromone on the inquired path is evaluated taking into consideration how many agents chose that path and how was the objective value of that path (Eq. 10). The weight of the solution goodness is the makespan (i.e., L_{ant}). A constant of pheromone updating (i.e., Q), equal for all ants and user, defined according to the tuning of the algorithm, is introduced as quantity of pheromone per unit of time (Eq. 11). The algorithm works as follow. It is computed the makespan value for each agent of the colony ($L_{ant}(0)$), following visibility and pheromone defined initially by the user ($\tau_{ij}(0)$) equal for all connections. It is evaluated and laid, according to the disjunctive graph representation of the instance in issue, the amount of pheromone on each arc (evaporation coefficient is applied to design the environment at the next step).

$$\Delta\tau_{ij}(t) = \sum_{ant=1}^{ants} \Delta\tau_{ij}^{ant}(t) \quad (10)$$

http://dx.doi.org/10.5772/55431

$$\Delta\tau_{ij}^{ant}(t) = \left\{ \begin{array}{ll} \dfrac{Q}{L_{ant}} & \text{if ant-th followed edge (i,j)} \\ 0 & \text{otherwise} \end{array} \right\} \tag{11}$$

Visibility and updated pheromone trail fixes the probability (i.e., the fitness values) of each node (i.e., task) at each iteration; for each cycle, it is evaluated the output of the objective function ($L_{ant}(t)$). An objective function value is optimised accordingly to partial good solution. In this improvement, relative importance is given to the parameters α and β. Good elements for choosing these two parameters are: $\alpha / \beta \cong 0$ (which means low level of α) and little value of α ($0<\alpha\leq 2$) while ranging β in a larger range ($0<\beta\leq 6$).

4.3. Bees Algorithm (BA) approach

A colony of bees exploits, in multiple directions simultaneously, food sources in the form of antera with plentiful amounts of nectar or pollen. They are able to cover kilometric distances for good foraging fields [50]. Flower paths are covered based on a stigmergic approach – more nectar places should be visited by more bees [51].

The foraging strategies in colonies of bees starts by scout bees – a percentage of beehive population. They wave randomly from one patch to another. Returning at the hive, those scout bees deposit their nectar or polled and start a recruiting mechanism rated above a certain quality threshold on nectar stored [52]. The recruiting mechanism is properly a launching into a wild dance over the honeycomb. This natural process is known as waggle dance" [53]. Bees, stirring up for discovery, flutter in a number from one to one hundred circuits with a waving and returning phase. The waving phase contains information about direction and distance of flower patches. Waving phases in ascending order on vertical honeycomb suggest flower patches on straightforward line with sunbeams. This information is passed using a kind of dance, that is possible to be developed on right or on left. So through this dance, it is possible to understand the distance from the flower, the presence of nectar and the sunbeam side to choose [54].

The waggle dance is used as a guide or a map to evaluate merits of explored different patches and to exploit better solutions. After waggle dancing on the dance floor, the dancer (i.e. the scout bee) goes back to the flower patch with follower bees that were waiting inside the hive. A squadron moves forward into the patches. More follower bees are sent to more promising patches, while harvest paths are explored but they are not carried out in the long term. A swarm intelligent approach is constituted [55]. This allows the colony to gather food quickly and efficiently with a recursive recruiting mechanism [56].

The Bees Algorithm (i.e., BA) is a population-based search; it is inspired to this natural process [38]. In its basic version, the algorithm performs a kind of neighbourhood search combined with random search. Advanced mechanisms could be guided by genetics [57] or taboo operators [58]. The standard Bees Algorithm first developed in Pham and Karaboga in 2006 [59, 60] requires a set of parameters: no. of scout bees (n), no. of sites selected out of n visited sites (m), no. of best sites out of m selected sites (e), no. of bees recruited for the best e sites (nep),

no. of bees recruited for the other *m-e* selected sites (*nsp*), initial size of patches (*ngh*). The standard BA starts with random search.

The honey bees' effective foraging strategy can be applied in operation management problems such as JSSP. For each solution, a complete schedule of operations in JSP is produced. The makespan of the solution is analogous to the profitability of the food source in terms of distance and sweetness of the nectar. Bees, *n* scouts, explore patches, *m* sites - initially a scout bee for each path could be set, over total ways, *ngh*, accordingly to the disjunctive graph of fig. 2, randomly at the first stage, choosing the shorter makespan and the higher profitability of the solution path after the first iteration.

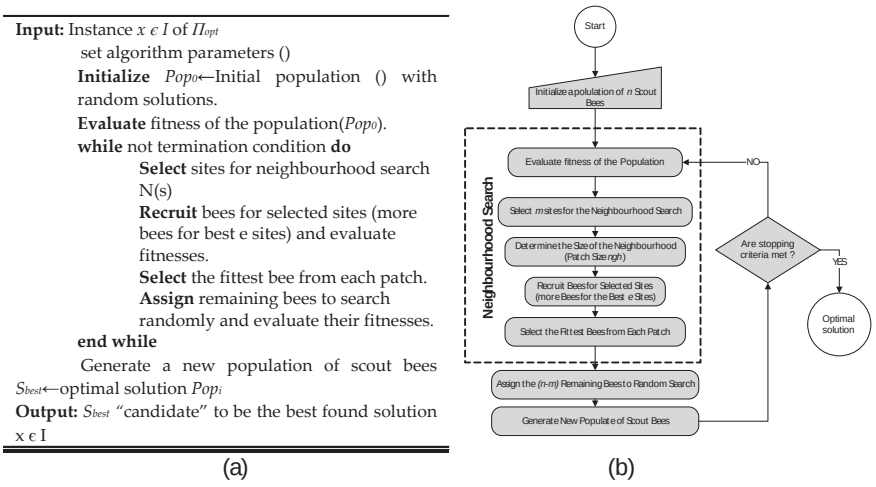


Figure 5. The Bees Algorithm model; 6a. the BA pseudo code; 6b. the flow chart of a general BA procedure.

Together with scouting, this differential recruitment is the key operation of the BA. Once a feasible solution is found, each bee will return to the hive to perform a waggle dance. The output of the waggle dance will be represented by a list of "elite solutions", *e* best selected sites, from which recruited bees, *nep*, are chosen for exploration from the population into the hive. Researches of patches are conducted, other *nsp* bees, in the neighbourhood of the selected sites, *m-e* sites. System maintains, step repetition: *imax*, where each bee of the colony of bees will traverse a potential solution. Flower patches, *e* sites, with better fitness (makespan) will have a higher probability for "elite solutions", promoting the exploitation to an optimal solution.

4.4. Electromagnetism like Method (EM)

The Electromagnetic Like Algorithm is a population based meta-heuristics proposed by Birbil and Fang [61] to tackle with combinatorial optimisation problems. Algorithm is based on the

http://dx.doi.org/10.5772/55431

natural law of attraction and repulsion between charges (Coulomb's law) [62]. EM simulates electromagnetic interaction [63]. The algorithm evaluates fitness of solutions considering charge of particles. Each particle represents a solution. Two points into the space had different charges in relation to what electromagnetic field acts on them [64]. An electrostatic force, in repulsion or attraction, manifests between two points charges. The electrostatic force is directly proportional to the magnitudes of each charge and inversely proportional to the square of the distance between the charges. The fixed charge at time iteration (t) of particle i is shown as follows:

$$q_i(t) = \exp\left(-n * \left(f\left(x_i, t\right) - f\left(x_{best}, t\right) / \left(\sum_{i=1}^{m} \left(f\left(x_i\ , t\right) - f\left(x_{best}, t\right) \right) \right) \right) \right) \quad \forall i = 1, .., m \tag{12}$$

Where t represents the iteration step, $q_i(t)$ is the charge of particle i at iteration t, $f(x_i,t)$, $f(x_{best},t)$, and $f(x_k,t)$ denote the objective value of particle i, the best solution, and particle k from m particles at time t; finally, n is the dimension of search space.The charge of each point i, $q_i(t)$, determines point's power of attraction or repulsion. Points (x_i) could be evaluated as a task into the graph representation (fig. 2).

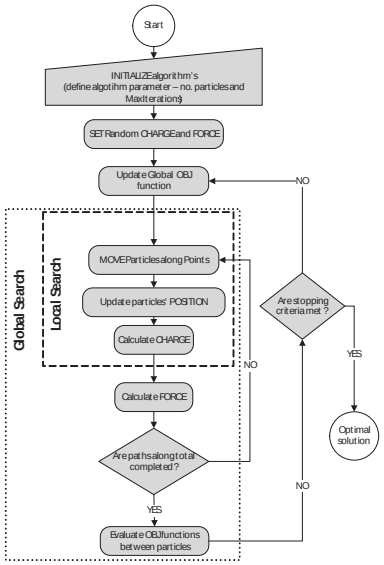
The particles move along with total force and so diversified solutions are generated. The following formulation is the resultant force of particle i:

$$F_i(t) = \sum \left\{ \begin{array}{l} \left(x_j(t) - x_i(t)\right) * \dfrac{\left(q_i(t) * q_j(t)\right)}{\left\| x_j(t) - x_i(t) \right\|^2} : f\left(x_j, t\right) < f\left(x_i, t\right) \\ \left(x_i(t) - x_j(t)\right) * \dfrac{\left(q_i(t) * q_j(t)\right)}{\left\| x_j(t) - x_i(t) \right\|^2} : f\left(x_j, t\right) \geq f\left(x_i, t\right) \end{array} \right\}, \forall i = 1, ..., m \tag{13}$$

The following notes described an adapted version of EM for JSSP. According to this application, the initial population is obtained by choosing randomly from the list or pending tasks, as for the feasibility of solution, particles' path. The generic pseudo-code for the EM is reported in figure 6. Each particle is initially located into a source node (see disjunctive graph of figure 2). Particle is uniquely defined by a charge and a location into the node's space. Particle's position in each node is defined in a multigrid discrete set. While moving, particle jumps in a node based on its attraction force, defined in module and direction and way. If the force from starting line to arrival is in relation of positive inequality, the particles will be located in a plane position in linear dependence with force intensity. A selection mechanism could be set in order to decide where particle is directed, based on node force intensity. Force is therefore the resultant of particles acting in node. A solution for the JS is obtained only after a complete path from the source to the sink and the resulting force is updated according to the normalized makespan of different solutions.

Input: Instance $x \in I$ of Π_{opt}
set algorithm parameters ()
$i \leftarrow 0$
Initialize: m sample points are selected at random from the feasible region.
Evaluate Force and Distance of current state
while not termination condition **do**
GLOBAL SEARCH
while job is not completed **do**
LOCAL SEARCH:
Move the sample points toward the local minimums (distance and force need to be evaluated).
Calculate force: $F \leftarrow$ CalcF charge value is assigned to each point (q_i).
Move (F) points along the total force vector.
Evaluate particles()
end while
$S_{best} \leftarrow$optimal solution
Update *Global Force* according to S
end while
output: S_{best} "candidate" to be the best found solution x ϵ I

(a)



(b)

Figure 6. The Electromagnetic like Method; **6a**. the EM pseudo code; **6b**. the flow chart of a general EM procedure.

4.5. Simulated Annealing (SA)

The simulated annealing was presented by Scott Kirkpatrick *et al*. in 1983 [65] and by Vlado Černý in 1985 [66]. This optimization method is based on works of Metropolis *et al*., [67] which allows describing the behaviour of a system in thermodynamic equilibrium at a certain temperature. It is a generic probabilistic metaheuristic used to find a good approximation to the global optimum of a given objective function. Mostly it is used with discrete problems such as the main part of the operations management problems.

Name and inspiration come from annealing in metallurgy, a technique that, through the heating and a controlled process of cooling, can increase the dimensions of the crystals inside the fuse piece and can reduce the defects inside the crystals structure. The technique deals with the minimization of the global energy E inside the material, using a control parameter called temperature, to evaluate the probability of accepting an uphill move inside the crystals structure. The procedure starts with an initial level of temperature T and a new random solution is generated at each iteration, if this solution improves the objective function, i.e., the E of the system is lower than the previous one. Another technique to evaluate the improvement of the system is to accept the new random solution with a likelihood according to a probability $exp(-\Delta E)$, where ΔE is the variation of the objective function. Afterwards a new iteration of the procedure is implemented.

As follows there is the pseudo-code of a general simulated annealing procedure:

http://dx.doi.org/10.5772/55431

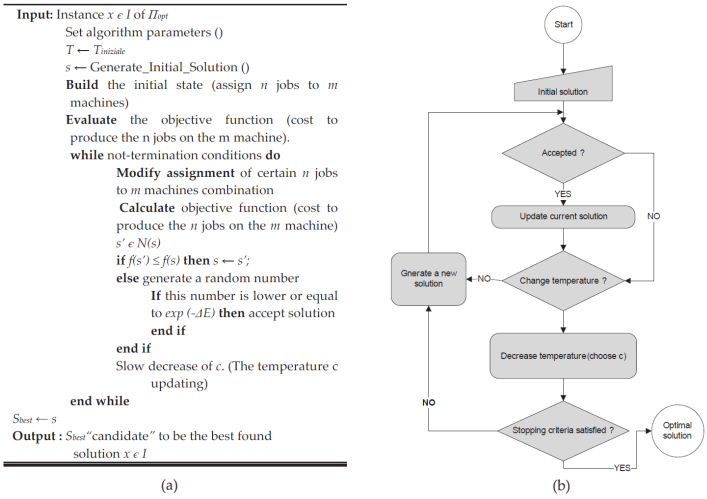


Figure 7. The Simulated Annealing model; 7a. the SA pseudo code; 7b. the flow chart of a general SA procedure

For the scheduling issues, the application of the SA techniques requires the solutions fitness generated by each iteration, that is generally associated to the cost of a specific scheduling solution; the cost is represented by the temperature that is reduced for each iteration [68]. The acceptance probability can be measured as following:

$$Aij = \left\{ min \quad \left[1, \; exp \left[\left(- \frac{C(j) - C(i)}{c} \right) \right] \right] \right\} \tag{14}$$

Another facet to be analysed is the stopping criteria, which can be fixed as the total number of iterations of the procedure to be computed.

4.6. Tabu Search (TS)

Tabu search (Glover, 1986) is an iterative search approach characterised by the use of a flexible memory [69]. The process with which tabu search overcomes local optimality is based on the evaluation function that chooses the highest evaluation solution at each iteration. The evaluation function selects the move, in the neighbourhood of the current solution, that produces the most improvement or the least deterioration in the objective function. Since, movement are accepted based on a probability function, a tabu list is employed to store characteristics of accepted moves so to classify them as taboo (i.e., to be avoided) in the later iteration. This is used to dodge cycling movements. A strategy called forbidding is employed to control and update the tabu list. This method was formalized by Glover [69]. An algorithm based on tabu search requires some elements: (i) the move, (ii) the neighbourhood, (iii) an initial solution, (iv) a search strategy, (v) a memory, (vi) an objective function and (vii) a stop criterion. The of TS is based on the definition of a first feasible solution S, which is stored as the current seed

and the best solution, at each iteration, after the set of the neighbours is selected between the possible solutions deriving from the application of a movement. The value of the objective function is evaluated for all the possible movements, and the best one is chosen. The new solution is accepted even if its value is worse than the previous one, and the movement is recorded in a list, named taboo list.

For the problem of the scheduling in the job shops, generally a row of assignments of n jobs to m machines is randomly generated and the *cost* associated is calculated to define the fitness of the solution [70]. Some rules of movements can be defined as the crossover of some jobs to different machines and so on, defining new solutions and generating new values of the objective functions. The best solution between the new solutions is chosen and the movement is recorded in a specific file named taboo list. The stopping criterion can be defined in many ways, but simplest way is to define a maximum number of iterations [71].

In figure 8 are reported the pseudo-code and the flowchart for the application of TS to JSSP.

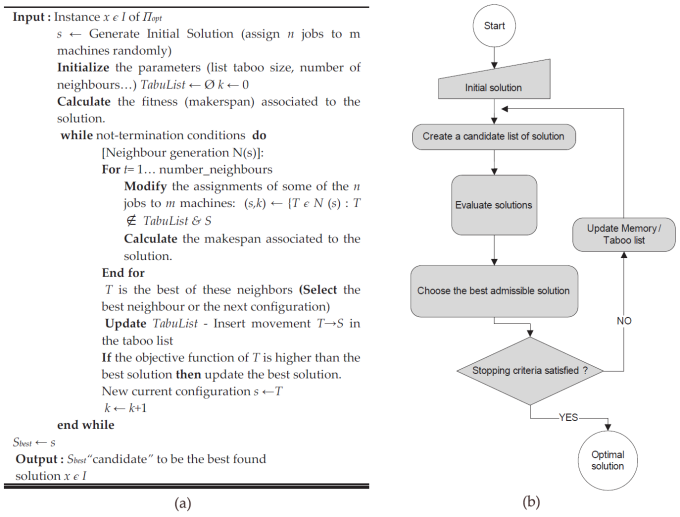


Figure 8. The Tabu Search approach; 8a. the TS pseudo code; 8b. the flow chart of a general TS procedure.

4.7. Neural Networks (NNs)

Neural networks are a technique based on models of biological brain structure. Artificial Neural Networks (NN), firstly developed by McCulloch and Pitts in 1943, are a mathematical model which wants to reproduce the learning process of human brain [72]. They are used to simulate and analyse complex systems starting from known input/output examples. An algorithm processes data through its interconnected network of processing units compared to neurons. Consider the Neural Network procedure to be a "black box". For any particular set of inputs (particular scheduling instance), the black box will give a set of outputs that are

http://dx.doi.org/10.5772/55431

suggested actions to solve the problem, even though output cannot be generated by a known mathematical function. NNs are an adaptive system, constituted by several artificial neurons interconnected to form a complex network, those change their structure depending on internal or external information. In other words, this model is not programmed to solve a problem but it learns how to do that, by performing a *training* (or *learning) process* which uses a record of examples. This data record, called *training set*, is constituted by inputs with their corresponding outputs. This process reproduces almost exactly the behaviour of human brain that learns from previous experience.

The basic architecture of a neural network, starting from the taxonomy of the problems faceable with NNs, consists of three layers of neurons: the *input layer*, which receives the signal from the external environment and is constituted by a number of neurons equal to the number of input variables of the problem; the *hidden layer* (one or more depending on the complexity of the problem), which processes data coming from the input layer; and the *output layer*, which gives the results of the system and is constituted by as many neurons as the output variables of the system.

The error of NNs is set according to a testing phase (to confirm the actual predictive power of the network while adjusting the weights of links). After having built a training set of examples coming from historical data and having chosen the kind of architecture to use (among feed-forward networks, recurrent networks), the most important step of the implementation of NNs is the learning process. Through the training, the network can infer the relation between input and output defining the "strength" (weight) of connections between single neurons. This means that, from a very large number of extremely simple processing units (neurons), each of them performing a weighted sum of its inputs and then firing a binary signal if the total input exceeds a certain level (activation threshold), the network manages to perform extremely complex tasks. It is important to note that different categories of learning algorithms exists: (i) supervised learning, with which the network learns the connection between input and output thank to known examples coming from historical data; (ii) unsupervised learning, in which only input values are known and similar stimulations activate close neurons otherwise different stimulations activate distant neurons; and (iii) reinforcement learning, which is a retro-activated algorithm capable to define new values of the connection weights starting from the observation of the changes in the environment. Supervised learning by back error propagation (BEP) algorithm has become the most popular method of training NNs. Application of BEP in Neural Network for production scheduling is in: Dagli et al. (1991) [73], Cedimoglu (1993) [74], Sim et al. (1994) [75], Kim et al. (1995) [76].

The mostly NNs architectures used for JSSP are: searching network (Hopfield net) and *error correction network* (Multi Layer Perceptron). The Hopfield Network (a content addressable memory systems with weighted threshold nodes) dominates, however, neural network based scheduling systems [77]. They are the only structure that reaches any adequate result with benchmark problems [78]. It is also the best NN method for other machine scheduling problems [79]. In Storer *et al.* (1995) [80] this technique was combined with several iterated local search algorithms among which space genetic algorithms clearly outperform other implementations [81]. The technique's objective is to minimize the energy function E that

corresponds to the makespan of the schedule. The values of the function are determined by the precedence and resource constraints which violation increases a penalty value. The Multi Layer Perceptron (i.e., MLP) consists in a black box of several layers allowing inputs to be added together, strengthened, stopped, non-linearized [82], and so on [83]. The black box has a great no. of knobs on the outside which can be filled with to adjust the output. For the given input problem, the training (network data set is used to adjust the weights on the neural network) is set as optimum target. Training an MLP is NP-complete in general.

In figure 9 it is possible to see the pseudo-code and the flow chart for the neural networks.

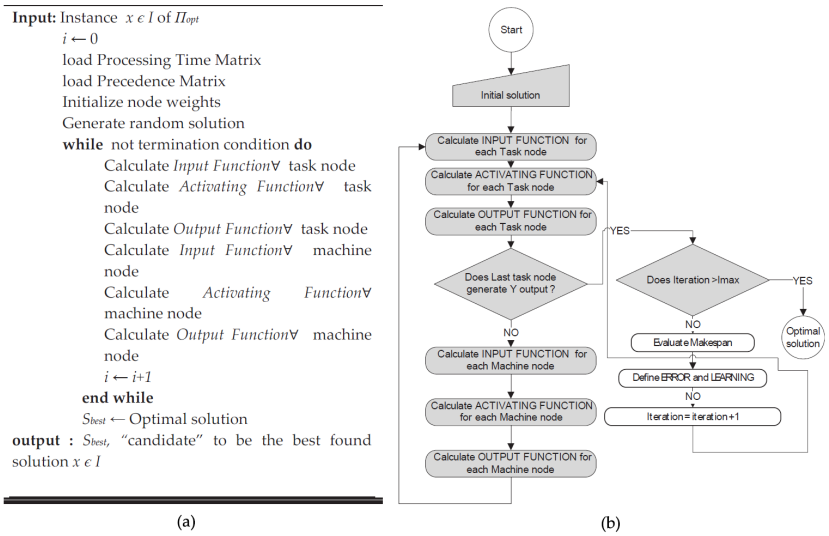


Figure 9. The NNs model; 9a. the implemented NNs pseudo code; 9b. the flow chart of generic NNs.

5. Discussion and conclusions

In this chapter, it was faced the most intricate problem (i.e., Job Shop) in order to explain approaches for scheduling in manufacturing. The JSP is one of the most formidable issues in the domain of optimization and operational research. Many methods were proposed, but only application of approximate methods (metaheuristics) allowed to efficiently solve large scheduling instances. Most of the best performing metaheuristics for JSSP were described and illustrated.

The likelihood of solving JSP can be greatly improved by finding an appropriate problem representation in computer domain. The acyclic graph representation is a quite good way to model alternatives in scheduling. How to fit approaches with problem domain (industrial manufacturing system) is generally a case in issue. Approaches are obviously affected by data

http://dx.doi.org/10.5772/55431

and the results are subject to tuning of algorithm's parameters. A common rule is: less parameters generate more stable performances but local optimum solutions. Moreover, the problem has to be concisely encoded such that the job sequence will respect zoning and sequence constraints. All the proposed approaches use probabilistic transition rules and fitness information function of payoff (i.e., the objective function).

ACO and BE manifest common performances in JSSP. They do not need a coding system. This factor makes the approaches more reactive to the particular problem instance in issue. Notwithstanding, too many parameters have to be controlled in order to assure diversification of search. GAs surpasses their cousins in the request for robustness. The matching between genotype and phenotype across the schemata must be investigated in GAs in order to obtain promising results. The difficult of GA is to translate a correct phenotype from a starting genotype. A right balancing between crossover and mutation effect can control the performance of this algorithm. The EM approach is generally affected by local stability that avoid global exploration and global performance. It is, moreover, subject to infeasibility in solutions because of its way to approach at the problem. SA and TS, as quite simpler approaches, dominate the panorama of metaheuristics proposal for JS scheduling. They manifest simplicity in implementation and reduction in computation effort but suffer in local optimum falls. These approaches are generally used to improve performances of previous methodologies and they enhance their initial score. The influence of initial solutions on the results, for overall approaches, is marked. Performances of NNs are generally affected by the learning process, over fitting. Too much data slow down the learning process without improving in optimal solution. Neural Network is, moreover, affected by difficulties in including job constraints with network representation. The activating signal needs to be subordinated to the constraints analysis.

Based on authors experience and reported paragraphs, it is difficult to definitively choose any of those techniques as outstanding in comparison with the others. Measurement of output and cost-justification (computational time and complexity) are vital to making good decision about which approach has to be implemented. They are vital for a good scheduling in operations management. In many cases there are not enough data to compare – benchmark instances, as from literature for scheduling could be useful - those methods thoroughly. In most cases it is evident that the efficiency of a given technique is problem dependent. It is possible that the parameters may be set in such way that the results of the algorithms are excellent for those benchmark problems but would be inferior for others. Thus, comparison of methods creates many problems and usually leads to the conclusion that there is no the only best technique. There is, however, a group of several methods that dominates, both in terms of quality of solutions and computational time. But this definition is case dependent.

What is important to notice here is: performance is usually not improved by algorithms for scheduling; it is improved by supporting the human scheduler and creating a direct (visual) link between scheduling actions and performances. It is reasonable to expect that humans will intervene in any schedule. Humans are smarter and more adaptable than computers. Even if users don't intervene, other external changes will happen that impact the schedule. Contingent maintenance plan and product quality may affect performance of scheduling. An algorithmic approach could be obviously helpful but it has to be used as a computerised support to the

scheduling decision - evaluation of large amount of paths - where computational tractability is high. So it makes sense to see what optimal configuration is before committing to the final answer.

Author details

Marcello Fera[1], Fabio Fruggiero[2], Alfredo Lambiase[1], Giada Martino[1] and Maria Elena Nenni[3]

1 University of Salerno – Dpt. of Industrial Engineering, Fisciano (Salerno), Italy

2 University of Basilicata – School of Engineering, Potenza, Italy

3 University of Naples Federico II – Dpt. of Economic Management, Napoli, Italy

References

[1] Stutzle, T. G. Local Search Algorithms for Combinatorial Problems- Analysis, Algorithms and New Applications; (1998).

[2] Reeves, C. R. Heuristic search methods: A review. In D.Johnson and F.O'Brien Operational Research: Keynote Papers, Operational Research Society, Birmingham, UK, (1996). , 122-149.

[3] Trelea, I. C. The Particle Swarm Optimization Algorithm: convergence analysis and parameter selection.Information Processing Letters(2003). , 85(6), 317-325.

[4] Garey, M. R, & Johnson, D. S. Computers and intractability: a guide to the theory of NP-completeness, Freeman, (1979).

[5] Wight Oliver WProduction Inventory Management in the computer Age. Boston, Van Nostrand Reinhold Company, Inc., New York, (1974).

[6] Baker, K. R. Introduction to sequencing and scheduling, John Wiley, New York, (1974).

[7] Cox James F., John H. Blackstone, Jr., Michael S. Spencer editors,APICS Dictionary, American Production &Inventory Control Society, Falls Church, Virginia, (1992).

[8] Pinedo Michael, Scheduling Theory, Algorithms, and Systems, Prentice Hall, Englewood Cliffs, New Jersey, (1995).

[9] Hopp, W, & Spearman, M. L. Factory Physics. Foundations of manufacturing Management. Irwin/McGraw-Hill, Boston; (1996).

http://dx.doi.org/10.5772/55431

[10] Muth, J. F, & Thompson, G. L. Industrial Scheduling. Prentice-Hall, Englewood Cliffs, N.J., (1963).

[11] Garey, M. R, Johnson, D. S, & Sethi, R. The Complexity of Flow Shop and Job Shop Scheduling. Math. of Operation Research, (1976). , 2(2), 117-129.

[12] Blazewicz, J. Domschke W. and Pesch E..,The jobshops cheduling problem: conventional and new solution techniques,EJOR,1996; 93: 1-33.

[13] Aarts E.H.L., Van Laarhoven P.J.M., LenstraJ.K., and Ulder N.L.J. A computational study of local search algorithms for job shop scheduling. ORSA Journal on Computing, 1994; 6 (2): 118-125.

[14] Brucker, P. Scheduling Algorithms. Springer-Verlag, Berlin, (1995).

[15] Panwalkar, S. S. and Iskander Wafix. A survey of scheduling rules. Operations Research, (1977). , 25(1), 45-61.

[16] Carlier, J, & Pinson, E. An algorithm for solving the job-shop problem. Management Science, (1989). , 35(2), 164-176.

[17] Bertrand, J. W. M. The use of workload information to control job lateness in controlled and uncontrolled release production systems, J. of Oper. Manag., (1983). , 3(2), 79-92.

[18] GanttHenry L.. Work,Wages,andProfits,secondedition,Engineering Magazine Co., NewYork, (1916). Reprinted by Hive Publishing Company, Easton, Maryland, 1973.

[19] CoxJames F., John H. Blackstone, Jr., and Michael S. Spencer, edt.,APICS Dictionary, American Production &Inventory Control Society, Falls Church, Virginia, (1992).

[20] Roy, B, & Sussman, B. Les problèmes d'ordonnancement avec contraintes disjunctive, (1964).

[21] Fruggiero, F, Lovaglio, C, Miranda, S, & Riemma, S. From Ants Colony to Artificial Ants: A Nature Inspired Algorithm to Solve Job Shop Scheduling Problems. In Proc. ICRP-18; (2005).

[22] Adams, J, Balas, E, & Zawack, D. The shifting bottleneck procedure for job shop scheduling. Management Science, (1988). , 34

[23] Reeves, C. R. Modern Heuristic Techniques for Combinatorial Problems. John Wiley & Sons, Inc; (1993).

[24] Giffler, B. and Thompson G.L. Algorithms for solving productions cheduling problems. OperationsResearch, 1960;Vol.8: 487-503.

[25] Gere, W. S. Jr., Heuristics in Jobshop Scheduling, Manag. Science, (1966). , 13(1), 167-175.

[26] Rajendran, C, & Holthaus, O. A comparative Study of Dispatching rules in dynamics flowshops and job shops, European J. Of Operational Research. (1991). , 116(1), 156-170.

[27] Hertz, A, & Widmer, M. Guidelines for the use of meta-heuristics in combinatorial optimization, European Journal of Operational Research, (2003). , 151

[28] Zanakis, H. S, Evans, J. R, & Vazacopoulos, A. A. Heuristic methods and applications: a categorized survey, European Journal of Operational Research,(1989). , 43

[29] Gondran, M, & Minoux, M. Graphes et algorithmes, Eyrolles Publishers, Paris, (1985).

[30] Hubscher, R, & Glover, . Applying tabu search with influential diversification to multiprocessor scheduling, Computers Ops. Res. 1994; 21(8): 877-884.

[31] Blum, C, & Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison,ACM Comput. Surv.,(2003). , 35, 268-308.

[32] Glover, F, & Kochenberger, G. A. Handbook of Metaheuristics, Springer, (2003).

[33] Kirkpatrick, S, Gelatt, C. D, & Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science 2000; , 220(4598), 671-680.

[34] Birbil, S. I. Fang S An Electromagnetism-like Mechanism for Global Optimization. Journal of Global Optimization. (2003). , 25(3), 263-282.

[35] Mitchell, M. An Introduction to Genetic Algorithms. MIT Press, (1999).

[36] Glover, F, & Laguna, M. (1997). Tabu Search. Norwell, MA: Kluwer Academic Publ.

[37] Dorigo, M, & Di, G. Caro and L. M. Gambardella. Ant algorithm for discrete optimization. Artificial Life, (1999). , 5(2), 137-172.

[38] Pham, D. T, Ghanbarzadeh, A, Koc, E, Otri, S, Rahim, S, & Zaidi, M. The Bees Algorithm. Technical Note, Manufacturing Engineering Centre, CardiffUniversity, UK, (2005).

[39] Zhou, D. N, Cherkassky, V, Baldwin, T. R, & Olson, D. E. Aneuralnetwork approach to job-shop scheduling.IEEE Trans. on Neural Network,(1991).

[40] Holland John HAdaptation in Natural and Artificial Systems, University of Michigan,(1975).

[41] Darwin Charles “Origin of the species”(1859).

[42] Beck, J. C, Prosser, P, & Selensky, E. Vehicle Routing and Job Shop Scheduling: What’s the difference?, Proc. of 13th Int. Conf. on Autom. Plan. and Sched. (ICAPS03); (2003).

[43] Bierwirth, C, Mattfeld, C, & Kopfer, H. On Permutation Representations for Scheduling Problems PPSN, (1996). , 310-318.

[44] Moon, I, & Lee, J. Genetic Algorithm Application to the Job Shop Scheduling Problem with alternative Routing, Industrial Engineering Pusan National Universit; (2000).

[45] Goss, S, Aron, S, Deneubourg, J. L, & Pasteels, J. M. Self-organized shortcuts in the Argentine ant. Naturwissenschaften. (1989). , 76, 579-581.

[46] Van Der Zwaan, S, & Marques, C. Ant colony optimization for job shop scheduling. In Proc. of the 3rd Workshop on genetic algorithms and Artificial Life (GAAL'99), (1999).

[47] Dorigo, M, Maniezzo, V, & Colorni, A. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, (1996). , 26, 1-13.

[48] Cornea, D, Dorigo, M, & Glover, F. editors. New ideas in Optimization. McGraw-Hill International, Published in (1999).

[49] Colorni, A, Dorigo, M, Maniezzo, V, & Trubian, M. Ant system for Job-shop Scheduling". JORBEL-Belgian Journal of Operations Research, Statistics and Computer Science, (1994).

[50] Gould, J. L. Honey bee recruitment: the dance-language controversy.Science.(1975).

[51] Grosan, C, Ajith, A, & Ramos, V. Stigmergic Optimization: Inspiration, Technologies and Perspectives.Studies in Computational Intelligence.(2006). Springer Berlin/ Heidelberg., 31

[52] Camazine, S, Deneubourg, J, Franks, N. R, Sneyd, J, Theraula, G, & Bonabeau, E. Self-Organization in Biological Systems. Princeton: Princeton University Press, (2003).

[53] Von Frisch, K. Bees: Their Vision, Chemical Senses and Language. (Revised edn) Cornell University Press, N.Y., Ithaca, (1976).

[54] Riley, J. R, Greggers, U, Smith, A. D, Reynolds, D. R, & Menzel, R. The flight paths of honeybees recruited by the waggle dance". Nature(2005). , 435, 205-207.

[55] Eberhart, R, & Shi, . ., SwarmIntelligence.Morgan Kaufmann, San Francisco, 2001.

[56] Seeley, T. D. The wisdom of the Hive: The Socal Physiology of Honey Bee Colonies. Massachusetts: Harward University Press, Cambridge, (1996).

[57] Tuba, M. Artificial BeeColony(ABC) with crossover and mutation. Advances in computer Science, (2012). , 157-163.

[58] Chong CS Low, MYH Sivakumar AI, Gay KL. Using a Bee Colony Algorithm for Neighborhood Search in Job Shop Scheduling Problems, In 21st European Conference On Modelling and Simulation ECMS (2007).

[59] Karaboga, D, & Basturk, B. On The performance of Artificial Bee Colony (ABC) algorithm. Applied Soft Computing, (2008).

[60] Pham, D. T, Ghanbarzadeh, A, Koc, E, Otri, S, Rahim, S, & Zaidi, M. The Bees Algorithm- A Novel Tool for Complex Optimisation Problems, Proceedings of IPROMS (2006). Conference, , 454-461.

[61] Birbil, S. I. Fang S An Electromagnetism-like Mechanism for Global Optimization. Journal of Global Optimization. (2003). , 25(3), 263-282.

[62] Coulomb Premier mémoire sur l'électricité et le magnétismeHistoire de l'Académie Royale des Sciences, , 569-577.

[63] Durney Carl H. and Johnson, Curtis C. Introduction to modern electromagnetics. McGraw-Hill. (1969).

[64] Griffiths David J.Introduction to Electrodynamics(3rd ed.). Prentice Hall; (1998).

[65] Kirkpatrick, S, Gelatt, C. D, & Vecchi, M. P. Optimization by Simulated Annealing. Science. (1983). , 220(4598), 671-680.

[66] Cerný, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. J. of Optimization Theory and Applications. (1985). , 45, 41-51.

[67] Metropolis Nicholas; Rosenbluth, Arianna W.; Rosenbluth, Marshall N.; Teller, Augusta H.; Teller, Edward. Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics. (1953).

[68] Van Laarhoven P.J.M, Aarts E.H.L., and J. K. Lenstra. Job shop scheduling by simulatedannealing. Operations Research, Vol. 40, No. 1, pp. 113-125, 1992.

[69] Glover, F. Future paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research (1986). , 5(5), 533-549.

[70] Dell'Amico M and Trubian M. Applying tabu search to the job-shop scheduling problem. Annals of Operations Research, (1993). , 41, 231-252.

[71] Taillard, E.D. Parallel taboo search techniques for the job-shop scheduling problem. ORSAJournal on Computing, 1994; 6(2): 108-117.

[72] Marquez, L, Hill, T, Connor, O, & Remus, M. W., Neural network models for forecast a review. In: IEEE Proc of 25th Hawaii International Conference on System Sciences. (1992). , 4, 494-498.

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

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

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

[76] Kim, S. Y, Lee, Y. H, & Agnihotri, D. A hybrid approach for sequencing jobs using heuristic rules and neural networks, Prod. Planning and Control, (1995). , 6(5), 445-454.

[77] Hopfield, J. J, & Tank, D. W. Neural computational of decisions in optimization problems. Biological Cybernetics, (1985). , 52, 141-52.

[78] Foo, S. Y, & Takefuji, Y. Stochastic neural networks for solving job-shop scheduling: Part 1. Problem representation, in: Kosko B, IEEE International Conference on Neural Networks, San Diego, CA, USA, (1988). , 1988, 275-282.

[79] Haykin S Neural networks: a comprehensive foundation nd edn. Prentice Hall, New Jersey; (2001).

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

[81] Van Hulle, M. M. A goal programming network for mixed integer linear programming: A case study for the jobshop scheduling problem, International Journal of Neural Systems, (1991).

[82] Leshno, M, Lin, V. Y, Pinkus, A, & Schocken, S. Multilayer feedforward networks with a non-polynomial activation function can approximate any function. Neural Net. (1993). , 6(6), 861-867.

[83] Karlik, B, & Olgac, A. V. Performance analysis of various activation functions in generalized MLP architectures of neural networks. Int J. Artif. Int. Expert Syst. (2011). , 1(4), 111-122.