A Generic Decision Support Tool to Planning and Assignment Problems: Industrial Applications and Industry 4.0

Nathalie Klement, Cristóvão Silva

A Generic Decision Support Tool to Planning and Assignment Problems: Industrial Applications and Industry 4.0

Nathalie Klement and Cristóvão Silva

Abstract Decision support tools are essential to help the management of industrial systems at different levels: strategic to size the system; tactical to plan activities or assign resources; operational to schedule activities. We present a generic and modular decision support tool to solve different problems of planning, assignment, scheduling, or lot-sizing. Our tool uses a hybridization between a metaheuristic and a list algorithm. The specification of the considered problem is taken into account in the list algorithm. Several tactical and operational problems have been solved with our tool: a problem of planning activities with resources assignment for hospital systems, a lot-sizing and scheduling problem taking into account the setup time for a textile application and for a plastic injection problem, and a scheduling problem with precedence constraints. At the strategic level, this tool can also be used as part of Industry 4.0 to design reconfigurable manufacturing systems. This paper summarizes some problems solved with the proposed tool and presents its evolution.

9.1 Introduction

Industry 4.0 is the main international program which aims at improving the operational system in companies. More companies are concerned by this approach. Thanks to Internet of Things, "things" are connected to ease the communication, but other improvements may be envisaged. An integration of the whole production system is needed. Managers need to fully control the production, the actual one and the future one. The concept of Decision Support System (DSS) is known for, at least four decades, and it can be defined as a computer-based information system that

N. Klement (✉)
Arts et Métiers Institute of Technology, LISPEN, HESAM Université, Lille, France
e-mail: nathalie.klement@ensam.eu

C. Silva
Department of Mechanical Engineering, University of Coimbra, CEMMPRE, Coimbra, Portugal
e-mail: cristovao.silva@dem.uc.pt

supports business or organizational decision-making activities. They are especially valuable in situations in which the amount of available information is prohibitive for the intuition of an unaided human decision-maker, and in which precision and optimality are of importance. DSS can aid human cognitive deficiencies by integrating various sources of information, providing intelligent access to relevant knowledge, and aiding the process of structuring decisions (Druzdzel and Flynn 2010). DSS can be used to help the decider to solve many problems such as sizing of the shop floor, sizing of resources, planning of activities, assignment of resources, scheduling of activities. For now, most of the DSS are developed to solve specific problems. For instance, McKay and Wiers (2003) developed an integrated decision support for planning, scheduling, and dispatching tasks in a focused factory. Recently, after discussing with the stakeholders, some industrialists from small and medium size enterprises or big companies, we realized that no generic DSS was actually proposed. Given our conclusion from a literature review, no generic DSS found, and our discussion with the stakeholders, we decided to propose a generic and modular decision support tool. In the twenty-first century, we also have to consider the new available technologies.

Because of Industry 4.0, more data are available, more variability can be considered, so DSS is more important than ever. How can the future demand be integrated without major changes in the actual layout? The production system needs to be flexible to face the variety of products and the quantities of products. To propose some decision support tools to help the manager, new trends need to be considered such as continuous improvement, Big Data, or collaborative robotics (Guérin et al. 2016). For instance, it would be necessary to use the data in the shop floor to treat them in real time to adapt the schedule and the planning to the hazards, a link with the used Enterprise Resource Planning by the company needs to be done. Thanks to collaborative robotics, flexible production means can be used in the future flexible and agile production system, which will be reconfigurable. Many companies are actually thinking about converting their actual system into a reconfigurable manufacturing system.

This paper proposes a decision support tool which can be used to design reconfigurable manufacturing system. At the beginning, four static problems have been studied, in which the demands are already known. We focused on many problems: planning, scheduling, resources assignment. These problems can be summarized: they represent a system in which activities have to be done over a horizon planning. Each activity has some characteristics such as processing time and needed resources. Some resources are available to process the activities. The system is ruled by some constraints. Different objectives can be achieved such as optimizing the productivity of the system. Once this tool has been validated for the static problems, we can focus on the dynamic ones: the demands are not completely known, they can vary in quantity and/or variety of products. Information system has to be considered to integrate results from the tool to the shop floor.

Our research proposal may be considered as part of Information Technology (March and Smith 1995). Section 9.2 presents the developed generic and modular

decision support tool. Some examples of problems that have been identified and solved are presented in Sect. 9.3. The future work will focus on the reconfigurable manufacturing systems and Information systems provided by Industry 4.0, presented in Sect. 9.4. This paper ends with a conclusion in Sect. 9.5.

9.2 Generic and Modular Decision Support Tool

9.2.1 Genericity

The proposed tool, illustrated by Fig. 9.1, uses a hybridization of a metaheuristic and a heuristic, specifically a list algorithm. A single solution based metaheuristic or a population based metaheuristic can be used. The encoding used by the metaheuristic is a list Y of activities. List algorithm L considers the activities according to their order in list Y, to plan and assign them to the required resources, considering the problem constraints. This builds solution X. Objective function H evaluates solution X. According to this evaluation, the solution is chosen or not by the metaheuristic. At the end of the computation, the solution given by the hybridization is the best list Y^* of activities: the one which optimizes the objective function by applying the list algorithm. This hybridization can be used to solve many problems: only the list algorithm and the objective function need to be specific to the considered problem by integrating the different constraints which rule the system and the objectives to achieve.

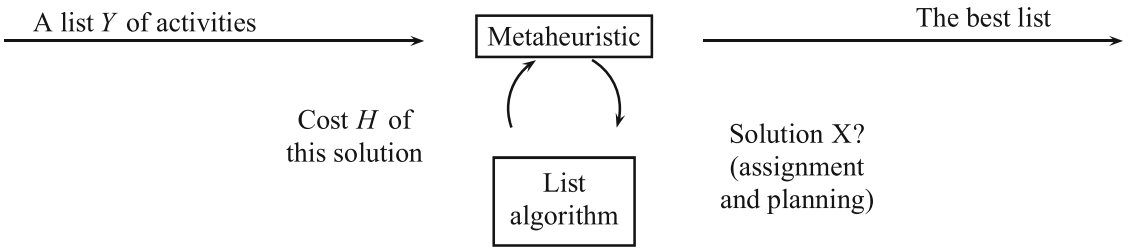

Fig. 9.1 Hybridization metaheuristic—list algorithm

9.2.2 A List Y of Activities

The general scheme of the encoding is given by Eq. (9.1), with Ω the set of all lists Y and S the set of all admissible solutions X built by the list algorithm L. Ω is the set of all permutations of activities. Cardinal of Ω is $N!$ with N the number of activities.

$Y \in \Omega$ is a list of activities. More details about the encoding are given in Gourgand et al. (2014).

$$Y \in \Omega \underset{Heuristic\ L}{\longrightarrow} L(Y) = X \in S \underset{Criterion\ H}{\longrightarrow} H(X) \tag{9.1}$$

9.2.3 Metaheuristic

The metaheuristic performs in Ω. An initial solution is randomly computed: a list of activities randomly sorted between one and the number of activities. A neighborhood system is used to visit the set of solutions; it allows to go from one solution to another one. Neighborhood system V is a permutation of two activities in list Y: the activity at position i permutes with the one at position j, with i and j being two different random numbers. V satisfies the accessibility and reversibility properties. Several metaheuristics have been used: some single solution based metaheuristics such as iterated local search and simulated annealing.

9.2.3.1 Iterated Local Search

A simple descent stays in the first found local minimum. Lourenço et al. (2002) showed that an iterated local search allows to go out from this local minimum. After having applied a local search, the current solution is disrupted to go out from the local minimum. Then, a new local search is applied to the disrupted solution.

Kangaroo algorithm is an iterated local search. It consists of applying a stochastic descent, but if there is no improvement of the current solution during A iterations, a jump is made. The used formula to compute the number of iterations A is given by Eqs. (9.2) and (9.3). To make this jump, a solution is chosen in a neighborhood system W, different from V. Kangaroo algorithm converges in probability to the set of optimal solutions if neighborhood system W satisfies the accessibility property. We choose W as the consecutive application five times of V.

$$A \geq card(V) \times \ln(2) \tag{9.2}$$

$$card(V) = \frac{N \times (N-1)}{2} \tag{9.3}$$

9.2.3.2 Simulated Annealing

Simulated annealing is inspired by a process used in metallurgy which consists of alternating cycles of slow cooling and heating. Inhomogeneous simulated annealing was used by Metropolis et al. (1953) to simulate the physical cooling in metallurgy.

Applied to the optimization field, it consists of executing a descent with a non-zero probability to choose a worst solution than the current one. This probability decreases while the number of iterations increases. Aarts and Laarhoven (1987) proved that simulated annealing converges in probability to the set of optimal solutions if neighborhood system V satisfies the accessibility and reversibility properties.

Two parameters are used:

- The initial temperature T_0 is chosen such that all the transitions are accepted at the beginning, defined by Eq. (9.4).

$$e^{-\frac{H(Y')-H(Y)}{T_0}} \simeq 1, \forall (Y, Y') \tag{9.4}$$

- The decreasing factor α is chosen such that the final temperature T_a is close to zero, computed as Eq. (9.5), with $IterMax$ the maximum number of iterations.

$$\alpha = \sqrt[IterMax]{(\frac{T_a}{T_0})} \tag{9.5}$$

9.2.4 List Algorithm

A list algorithm is used to build solution X from list Y: it assigns the activities to resources over the horizon planning according to the problem constraints.

List scheduling algorithms are one-pass heuristics that are widely used to make schedules. Zhu and Wilhelm (2006) defined standard list scheduling algorithm as the construction of a schedule by assigning each activity in the listed order to the first resource that becomes idle. It is important to work with a list algorithm because the metaheuristic browses the set of lists Y.So the used algorithm needs to consider the order of the list to assign activities to resources over the horizon planning.

The developed list algorithms will be detailed according to the considered problem in Sect. 9.3.

9.2.5 Objective Function

Solutions are compared according to an objective function which characterizes the quality of the solution. The aim of our tool is to find the solution X^* defined by Eq. (9.6).

$$X^* = \min_{\forall X \in S} H(X) \tag{9.6}$$

Depending on the considered problem and the objectives to achieve, the objective function can be computed differently. The used objective function for each application will be detailed in Sect. 9.3. Most of the time, the weighed criteria method

Algorithm 1: Hybridization of simulated annealing and a list algorithm

Data: Initial solution Y; Temperature T_0; Decreasing factor α

$T := T_0$; $X := L(Y)$; $X^* := X$
while *necessary* **do**
  Choose uniformly and randomly $Y' \in V(Y)$
  $X' := L(Y')$
  if $H(X') < H(X^*)$ **then**
    $X^* := X'$; $Y^* := Y'$
  if $H(X') \leq H(X)$ **then**
    $X := X'$; $Y := Y'$
    else if $rand[0, 1) \leq e^{-\frac{H(Y')-H(Y)}{T}}$ **then**
      $X := X'$; $Y := Y'$
  Compute the new temperature $T := \alpha \times T$
return Y^*

defined by Coello (2000) is used. The objective function is a weighed sum between some criteria. An example with two criteria H_1 and H_2 is defined by Eq. (9.7). ω_2 is defined by Eq. (9.8), so both criteria can be easily readable. This method can be used with more than two criteria. The function H has to be minimized.

$$H(X) = 10^{\omega_2} \times H_2(X) + H_1(X) \tag{9.7}$$

$$10^{\omega_2} > \max_{\forall X \in S} H_1(X) \tag{9.8}$$

9.2.6 The Best List Y*

Algorithm 1 describes the whole method with the example of simulated annealing as the used metaheuristic. Set Ω of the lists of activities is browsed, thanks to the metaheuristic using neighborhood system V. Lists are compared, thanks to list algorithm L and objective function H. According to an acceptance criterion, some lists are selected. At the end, the metaheuristic gives the best found list Y^*.

9.3 Applications

In the industrial engineering field, many problems can be solved according to the used decision level. Some of these problems are summarized in Table 9.1. Some identified real problems have already been solved with our tool. Each of these problems has been classified, thanks to a literature review. The already identified applications are about tactical or operational level. Section 9.3.1 presents the first application: a

Table 9.1 Decision levels and problems

Decision level	Horizon planning	Problems	Use case	Literature
Strategic	Years	Sizing of the system	Ø	Ø
Tactical	Months	Activities planning	Hospital case	BPP
		Resources assignment	Textile	DLSP
Operational	Weeks	Activities scheduling	Injection	CLSP
	Days	Response to hazards	Fridge	JSSP

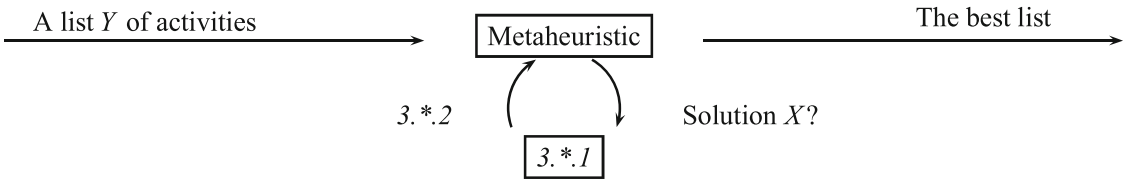

Fig. 9.2 Presentation of the applications

problem of activities planning and resources assignment in the hospital system. The second application is about a lot-sizing and scheduling problem in a textile company, presented in Sect. 9.3.2. The third one is also a lot-sizing and scheduling problem but about an injection plastic case, presented in Sect. 9.3.3. The difference between these two problems is the fabrication order policy: in the textile case, only one product may be made per period, in the injection plastic case, several products may be produced in one period. The last one, presented in Sect. 9.3.4, takes into account precedence between activities. For each problem, an analysis of the problem is made. Then, the description of the proposed list algorithm and the objective function, which are the only two parts of our tool which need to be specific to the problem, is given and is explained in Fig. 9.2. Then, the results of the use of our tool are presented.

9.3.1 Activities Planning and Resources Assignment

This method has firstly been used to solve an activities planning and resources assignment problem in a multi-place hospital context (Klement et al. 2017). The application has been made for the medical imaging department. The planning horizon is divided into periods. Each period represents one half-day. Activities are exams.

The hospital system, called Hospital Territorial Grouping (HTG), is made up of several places. Some material resources belong to each place. Each material resource has an opening schedule. All material resources cannot treat all the exams, there are some incompatibilities between exams and material resource. Each exam has a given

Algorithm 2: List algorithm for a hospital system

```
Data: List of exams (Y_i)_{i∈{1,N}}; Opening schedule of all resources during all periods;
      Processing time of all exams
Occupied time := 0 for all resources and all periods
forall the i do
    First resource, first period, assigned := false
    while (assigned = false) AND current period ≤ max of periods do
        while (assigned = false) AND current resource ≤ max of resources do
            if Exam Y_i is compatible with current resource then
                if Exam Y_i fits in couple (resource, period) then
                    Assign exam Y_i to couple (resource, period)
                    Update occupied time of couple (resource, period)
                    assigned := true
            Next resource
        Next period
```

processing time and a due date before it has to be done. The objective is to assign each exam to one material resource during one period, respecting the time constraints and the compatibility constraints.

9.3.1.1 List Algorithm

This problem has been analyzed as a bin packing problem Gourgand et al. (2014). The list algorithm used to assign each exam to one resource and one period is an extension of an existing list algorithm to solve the bin packing problem, presented by Algorithm 2.

N exams have to be assigned. The proposed list algorithm consists in assigning the exams to the first available couple (resource, period). Compatibility between exam and couple must be respected. Exams have to be done as soon as possible so all resources from one period are tested before going to the next period. Current exam Y_i is assigned to couple (resource, period) if available time in this couple is bigger than the processing time of exam Y_i and if exam Y_i is compatible with the resource. A new period is considered if there is no couple in the current one with enough time to receive the exam.

9.3.1.2 Objective Function

The objective function represents the timing aspect of our problem. Exams have to be done as soon as possible, thus the makespan, the period assigned to the last exam, should be considered. Because many solutions may have the same makespan, we choose instead the sum of assigned periods to all exams, so the solutions can be

dissociated. This criterion is written H_S. Another criterion is considered to ensure that most of the exams are assigned before their due date. This criterion, written H_D, is computed as the number of exams assigned after their due date. The weighed criteria method presented in Sect. 9.2.5 is used.

9.3.1.3 Results

The proposed tool provides a good planning of exams, with the assignment of one resource and one period, minimizing the objective function.

The data are randomly generated but the characteristics and the size of the data represent real instances. The HTG is made up of three places. The planning horizon is made by 8–40 periods. As a reminder, one period represents one half-day, thus the planning horizon is between 4 and 20 days. 4–8 resources are available. 50–500 exams need to be planned and assigned. Incompatibilities between exams and resources are randomly generated. Each processing time is between 5 and 100 min. Each material resource has an opening schedule equal to 300 min.

The results are detailed in Table 9.2. The host machine is powered by an Intel Xeon X5687 quad-core CPU running at 3.6 GHz. The computation has been stopped after thirty minutes. Each reported result is the value of the objective function for the best solution found in less than 30 min, but most of the time, the best solution is found in a few minutes. The results are presented as a couple of values $(H_D; H_S)$ with H_D the number of exams assigned after their due dates, and H_S the sum of assigned periods to all the exams.

The results compare three methods:

- The resolution of the mathematical model with an exact method by using the solver CPLEX. If no optimal solution has been found in less than thirty minutes by the solver, no result is written.

Table 9.2 Results (number of exams assigned after their due dates; sum of assigned periods to all the exams; best found solution in bold)

Number of exams	CPLEX	Gourgand et al. (2014)	ILS*	SA*
50	(**0**; **51**)	(0; 51)	(0; 51)	(**0**; **51**)
50	(**1**; **150**)	(10; 147)	(1; 151)	(**1**; **150**)
100	(**0**; **131**)	(0; 131)	(0; 131)	(**0**; **131**)
100	(**0**; **517**)	(2; 535)	(1; 516)	(0; 518)
200	(**0**; **266**)	(0; 266)	(0; 266)	(**0**; **266**)
200	–	(3; 1197)	(0; 1154)	(**0**; **1135**)
300	–	(0; 548)	(0; 537)	(**0**; **534**)
400	(**0**; **830**)	(0; 890)	(0; 841)	(0; 835)
500	–	(0; 1350)	(0; 1241)	(**0**; **1234**)
500	–	(194; 8218)	(19; 6382)	(**18**; **6659**)

- Results from the method previously published (Gourgand et al. 2014), using two single solution based metaheuristics (iterated local search and simulated annealing) in a classical way: the best value found by all these methods is reported.
- Results from our proposed method detailed in the current paper. The used metaheuristics are distinguished: iterated local search and simulated annealing, written ILS* and SA*.

The results are promising. Firstly, this problem has been solved by CPLEX, thanks to our mathematical model previously proposed (Gourgand et al. 2015). The solver finds an optimal solution only for small size of problems (less than 200 exams over 4 days). The solver does not find any solutions when the size of the problem increases. Then, it has been solved with two approximate methods: in a classical way and with a hybridization. Both methods find an optimal solution for the small instances. For biggest instances, the hybridization between a metaheuristic and a list algorithm outperforms the previous method. Simulated annealing seems to work better than iterated local search. Recently, the results have been improved, using the particle swarm optimization as a metaheuristic (Laurent and Klement 2019).

The assignment to human resources could also be added. Human resources can move over the different places belonging to the hospital system. Each human resource has a planning defining its availability. Each human resource has some skills to use some material resource. The new objective could be to assign each exam to one material resource and one human resource during one period. Some additional constraints have to be considered for each assigned exam: the capability of the human resource to work on the material resource, the availability of the human resource, and the location of the human resource at the place where the material resource is.

9.3.2 Discrete Lot-Sizing and Scheduling

The second application has been identified in a company (Klement et al. 2017). The horizon planning is 1 month, divided by periods of 1 day. Activities are jobs to produce.

The considered company is producing acrylic fiber used in the textile industry. These fibers are made following three steps. Dope preparation: a polymer is dissolved in an organic solvent to make the dope. Spinning: dope is going through a spinneret (a tool which is a metal plate with holes from different diameters) to obtain the synthetic filaments. After this step the fiber is called tow. Cutting and packing: the tow obtained in the previous step can suffer two different types of operation: it can simply be packed before being sent to the warehouse or, it can be cut in small segments, originating a new kind of fiber called raw, which is also packed before expedition.

Our study focuses on the spinning area. There are 10 non-identical spinning machines on the shop floor. Machines are dedicated, all products cannot be produced on all machines. A compatibility machine-product is defined. All machines do not have the same production rate: some of them can produce more tons of product per hour than others.

Fibers can be made from different diameters, within three colors: shiny, mat, and black. There are two types of fibers: tow and raw. In total, the company can make 60 different products. A change of color may induce the stop of the machine for its cleaning. Transition shiny–shiny and shiny–mat/black does not induce a stop, but transition mat/black–shiny does. If there is a setup (change of tool), change of color is made in the meantime. A tool can produce fibers from different diameters by changing the used tensity during the production. Moreover, each tool has a lifetime, from 8 to 45 days. When its lifetime is over, a new tool has to be used. A setup lasts 2 h. Lifetime of tools depends on the type and the color of the product and depends on the used machine. Constraints have to be respected:

- All or nothing: if a product is planned, production lasts 24 h even if the needed quantity is less. Over-quantity is stocked and used next month, it will be deduced from the next orders.
- Possible setup between two products:
  - Change of tool between two products,
  - Cleaning of the machine (transition mat–shiny or black–shiny),
  - Setup if lifetime is over.

The company wants to plan products within a planning horizon of 1 month, by periods of 24 h. Customers' orders are analyzed to make a production plan. Our objective is to schedule these fabrication orders on the different machines and the different periods during the considered month. The result will be a schedule over 28 days by periods of 24 h.

This problem has been identified as a Discrete Lot-Sizing and Scheduling Problem (Klement et al. 2017).

9.3.2.1 List Algorithm

As a reminder, the algorithm is considering jobs one by one respecting the order defined by the list. The used list algorithm is summarized in Algorithm 3. The list algorithm needs to be efficient to find a solution (good or not) in a small computation time. It does not need to be effective because a good solution will be found after having tried some neighbors, thanks to the metaheuristic.

The hypothesis is made that at the beginning of the month, all machines are empty, there is no tool on them, no raw material. This hypothesis is not restrictive. Maybe, the needed tool at the beginning of the month was already on the machine at the end of the previous month, so a setup will be avoided.

9.3.2.2 Objective Function

The objective function is used to compare solutions in the metaheurisitic. The main used criterion is the remaining quantity of all products at the end of the month, written

Algorithm 3: Principle algorithm of the list algorithm

```
   Data: Initialize all variables at zero
1  forall the JOB in the list do
2      while all quantity of JOB has not been scheduled do
3          MACHINE := first machine
4          while Next machines can be considered do
5              DAY := first day
6              while Next days can be considered do
7                  if MACHINE is compatible with JOB then
8                      if MACHINE is available then
9                          Assign JOB, the needed tool, and the needed color to MACHINE
                           during DAY
10                         Update the lifetime of the tool, change the tool if needed with a
                           new tool (lifetime = 100%)
11                         Update the number of setups if needed (change of tool, transition
                           from black to shiny, transition from mat to shiny)
12                         Update the remaining quantity of JOB to produce
13                 DAY := Next day
14             MACHINE := Next machine
```

C_q. This quantity would be produced next month. To provide all customer demands, our system must produce as many quantities as possible in a month. Another used criterion is the number of setup (caused by a change of tool, a change of color, or an exceeded lifetime), written C_s. A hierarchy of both criteria is used.

9.3.2.3 Results

To make our experiments, instances have been created, representing the data used by the company. Each instance is made by:

- A list of jobs to be done, with the needed quantity, the type and color, and the used tool.
- A matrix representing the effectiveness of the machines to produce the jobs, which can be null if the machine and the job are not compatible.
- The lifetime of each tool on each machine, depending on the type and color of the made product.

The experiments are made on a computer powered by an i7 CPU running at 2.6 GHz with 16 Go RAM. Table 9.3 summarizes the results of the proposed method giving the remaining quantity C_q and the number of setups C_s, with the needed computational time. The number of products is assigned to 10 machines over 28 days. Results are compared to the ones found by using the method presented by Silva and Ferreira (2006). The results show that our method gives better results than the ones from the previous method. Indeed, the previous method finds a solution given

Table 9.3 Results of our method

Instance	Silva and Ferreira (2006): (C_q; C_s)	Proposed method: (C_q; C_s; Time)
22 products	(552; 17)	(522.6; 17; 13 s)
25 products	(232; 16)	(68.0; 21; 8 s)
33 products	(387; 17)	(196.0; 19; 11 s)

by a constructive heuristic, while our method browses a set of solutions and gives the best one among all tested solutions.

9.3.3 Capacitated Lot-Sizing and Scheduling

The third solved problem with our tool is a capacitated lot-sizing and scheduling problem with setup and due dates, for the injection plastic case, presented by Silva and Ferreira (2004). Activities are jobs to schedule.

A set of N jobs has to be scheduled on shared machines with their respective mold. Each job has a given size, which determines its processing time, and an associated due date. A sequence dependent setup time is required when the production changes over from a job requiring a given mold to a job requiring a different one. A job is not allowed to be split but several jobs, requiring the same mold, may be grouped together to form one lot and, thus, saving setup costs. Due to compatibility factors, each mold can only be allocated to a subset of the available machines. Each mold is unique; thus, the same mold cannot be allocated to different machines during the same time period. The objective is to allocate jobs to each available machine and define the processing sequence in each machine in order to minimize the total tardiness.

9.3.3.1 List Algorithm

The proposed list algorithm to solve this problem is given by Algorithm 4.

9.3.3.2 Objective Function

The evaluation of a solution is made according to the value of the total tardiness. For a given solution, for each job, we compute the difference between its due date and its actual final date given by the solution. Then the sum of all tardiness for all jobs is made. The number of setups is not considered because all that matters is to minimize the tardiness while delivering the jobs.

Algorithm 4: List algorithm for an injection problem

```
   Data: List of jobs (Y_i)_{i∈{1,N}}, Processing time of all jobs
 1 forall the i do
 2     Order the machines according to their release date
 3     First machine
 4     while Job Y_i is not assigned do
 5         if Job Y_i and the machine are compatible then
 6             if The needed mold is available then
 7                 if The actual used mold on the machine is the good one then
 8                     Actualize the release date of the machine without setup
 9                     else
10                         Actualize the release date of the machine with setup
11                 Assign the job
12                 else
13                     Assign the job to the machine which uses the needed mold
14                     Actualize the release date of that machine, taking into account the
                       setup if needed
15         Next machine (modulo number of machines)
```

Table 9.4 Results of the use of our tool on the injection problem

	Silva and Ferreira (2004)		Proposed method	
Instance	Aver. tardiness	Aver. setup	Aver. tardiness	Aver. setup
3 machines	68.6	21.6	53.6	35.3
5 machines	212.9	32.7	98.3	45.7
10 machines	1020	82	964	114

9.3.3.3 Results

To test the proposed tool, company historical data were collected and used to randomly generate instances of several sizes: 3 instances with 3 machines (16, 18, and 20 molds; 47, 53, and 57 jobs, respectively); 3 instances with 5 machines (25, 26, and 26 molds; 81, 80, and 79 jobs), and one instance with 10 machines (59 molds and 177 jobs). Jobs processing times follow an exponential distribution with an average of 10.75 h and due dates were generated using a uniform distribution ranging between 24 and 312 h. Setup times consider the time to dismount the current mold (ranging between 15 and 45 min) and to mount the next one (ranging between 20 and 60 min).

For small instances, 2 machines and 10 jobs and 2 machines and 15 jobs, the solutions built by our method were compared to the optimal solution obtained solving a mathematical model with CPLEX. It was found that the proposed method gives the optimal solution for the two tested instances. Next, the solutions built by our method were compared to the ones used by the injection company in real life.

Table 9.4 presents the results obtained for each instance size with the heuristic proposed in Silva and Ferreira (2004), used by the company, and with the proposed

tool. Comparison considers the average tardiness and average number of setups among all instances of a given size. The results show that the developed tool is effective. An average reduction of 25% of tardiness is achieved. For some instances the reduction of tardiness is up to 50%. Nevertheless, the proposed method leads to an increase in the number of setups. It is important to note that, since the main objective of the company was to reduce the tardiness, the number of setups was not considered in the objective function of the proposed method. For other problems, an objective function considering tardiness and number of setups, with different weights, may be envisaged.

Silva and Ferreira (2004) used a two-phase algorithm: first it assigned molds to machines, and then it schedules jobs on each machine. So it was not possible to assign one mold on several machines at different times. In this way, the proposed method is less constraining which explains the better results. The proposed method is easy to develop and gives good results. For small instances, 3 and 5 machines, results are achieved in a small computational time: a few minutes. For larger instances, the computational time required to attain a good solution increases and can reach a few hours.

9.3.4 Scheduling with Precedence Constraints

The fourth application is about a scheduling problem but considering precedence constraints between activities. Activities are operations to make jobs. Several operations are needed to make a job, operations have to be done in order.

A set of metallic components are to be produced to satisfy the demand from an assembly line. Each component has to follow a processing sequence to be produced and each operation in this sequence requires a given resource (machine). The planning horizon is a week which is divided into five periods of one day. To satisfy the demand from the assembly line, a set of different lots of components is to be produced in each day of the planning horizon.

Thus, we have a set of N jobs which have to be processed on a set of M machines. Each job is defined by a sequence of operations that are associated with a particular machine. Each operation has a processing time and there is a setup time between the processing of two consecutive operations which is sequence dependent. Each job has a requested period. A penalty function is considered, with two parts:

- A storage cost (earliness) if the job is produced in a period prior to the requested one,
- A tardiness cost if the job is produced in a period after the requested one.

The objective is to define the operations sequence in each machine in order to minimize the total penalty.

This problem has been identified as a multi-period job-shop scheduling problem Silva and Klement (2017).

9.3.4.1 List Algorithm

The list algorithm developed for the case study company problem is in two steps presented in Algorithms 5 and 6. It is important to note that the list of jobs considered by the proposed algorithm is similar to the one presented in Table 9.5.

In the first step, described by Algorithm 5, the algorithm ignores the capacity constraints. All jobs are assigned to the required machine in the day they are expected

Algorithm 5: First step

```
forall the jobs in the list do
    forall the operations of the selected job j do
        Assign the selected operation i to the required machine m and production day d
        if operation i is the first one of job j then
            Start time of operation 1 of job j = release date of machine m for day d
            Finish time of operation 1 of job j = start time of operation 1 of job j +
            processing time of operation 1 of job j + setup time
            Release date of machine m = finish time of operation 1 of job j
        else
            Start time of operation i of job j = Max[release date of the machine m for day
            d; finish time of operation i-1 of job j]
            Finish time of operation i of job j = start time of operation i of job j +
            processing time of operation i of job j + setup time
            Release date of machine m = finish time of operation i of job j
```

Algorithm 6: Second step

```
forall the days do
    forall the machines do
        while capacity of machine m on day d is violated do
            Identify operation i from job j which leads to the capacity violation
            for operation 1 to i do
                for day d-1 to day 1 do
                    if capacity on selected day is available to process selected operation
                    then
                        Move operation to selected day
                        Update schedule of selected day and of day d
            for operation i to last operation of job j do
                for day d+1 to day 6 do
                    if capacity on selected day is available to process selected operation
                    then
                        Move operation to selected day
                        Update schedule of selected day
                        Update schedule of day d
```

Table 9.5 Examples of DD assignment rules

Part	Priority	Mon.	Tue.	Wed.	Thu.	Fri.	Operation 1			Operation 2			Operation 3		
							Mach.	Tool	PT	Mach.	Tool	PT	Mach.	Tool	PT
xxx	a		6				M1	T11	46						
yyy	b	2		7	5		M2	T22	3	M3	T32	2	M4	T43	43
zzz	c		18	20	16	32	M1	T11	9	M4	T42	7			
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

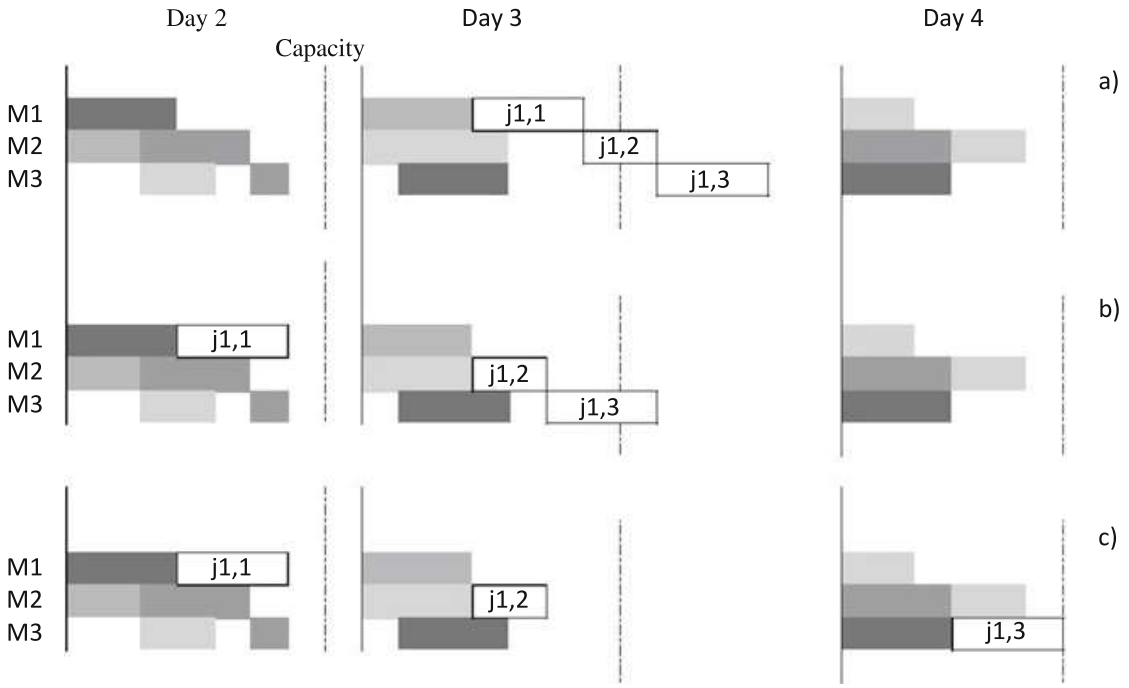

Fig. 9.3 Example of the application of step 2 from the list algorithm

to be concluded. To solve this problem and repair the schedule to respect capacity constraints, the second step described by Algorithm 6 is performed.

Note that when checking available capacity in a given machine for a given day to move a given operation, the algorithm considers not only if the required capacity is available, but also if it can be used by the moving operation while respecting precedence constraints. Figure 9.3 presents an example of the second step of the proposed list algorithm. After the first step of the algorithm, the schedule presented in Fig. 9.3a was obtained. This schedule represents several jobs (in gray scale) for which operations can be processed within the capacity limit. Operations 2 and 3 of job $j1$, represented in white, violate the capacity constraint for day 3.

Operation 1 of job $j1$, processed in machine 1, can be moved to day 2 where machine 1 has available capacity. This move leads to the schedule represented in Fig. 9.3b. After this move, the problem associated with operation 2 of job $j1$ is solved, but operation 3 continues to violate the capacity limit. Thus, operation 3 of job $j1$ is moved to day 4 where there is available capacity in machine 3 to process it. It is important to note that the algorithm considers a 6-period planning horizon, but the real planning horizon has only 5 periods. Jobs with operations assigned to period 6 will be considered as not produced and will be highly penalized.

9.3.4.2 Objective Function

The evaluation of a solution is made according to the value of the total lateness as defined by the penalty function. For each job, the difference between the period in

which it is concluded and the period for which it has been required is computed. The weights of earliness and tardiness can be different.

The objective function considered for the problem consists in minimizing the total penalty which considers three parcels:

- Tardiness: priority index × number of job units × number of days delayed;
- Earliness: 0.5 × number of job units × number of days anticipated;
- Unproduced jobs: 15 × number of job units.

9.3.4.3 Results

Historical company production data show that the weekly production scheduling problem faced by the company considers, on average, 300 orders and a total of approximately 450 operations. To avoid unnecessary complexity during the development and test of the tool it was decided to generate a smaller instance.

The instance generated considers 49 jobs, each one having a number of operations ranging between 1 and 3, which can be processed in one of the 6 machines that compose the shop floor. The total number of operations to be processed is 73. Jobs and their respective operations characteristics were generated, taking into account real data from the case study company. Since the generated instance is smaller than the real company problem, the capacity limit for each period was adjusted. The capacity limit was considered constant for the five considered periods and was fixed to 5760 s. This value was chosen after some pre-test runs of the tool, and it guaranties that there is at least one solution where all the jobs may be processed within the five available periods, i.e., no operation is delayed until period 6. On the other hand, it is sufficiently tight to guaranty that most lists of jobs generated by the metaheuristic permutation process will lead to a solution where one or more operations are delayed until period 6.

Results obtained by the tool are presented in the form of a list of jobs/operations to process each day of the week, indicating: the job number, the operation number, the machine where it will be processed, the tool to be used, the start time and the finish time of each operation of the jobs. In this list, anticipated operations (produced before the job required day) or delayed (produced after the required day) are highlighted.

To solve the test instance problem, the proposed method is run for 1000 iterations. Results are obtained in less than 10 min. The test was repeated 5 times. Results obtained for each test are presented in Table 9.6. For all the solutions, all jobs are processed within the 5 days planning horizon. The total penalty obtained is on average 210 units, the number of anticipated operations (5 for the worst case) or delayed operations (3 for the worst case) is low. The solution obtained for each iteration was registered. Figure 9.4 shows the evolution of the solutions obtained for the first 500 iterations for test 1. It can be seen that more than 97% of the solutions, all the ones with a penalty higher than 250, are not feasible, i.e., they have operations delayed until period 6. This confirms that the capacity limit is tight and that the number of feasible solutions is low. Since the tool is always able to find a feasible

Table 9.6 Results obtained for the considered instance

Test	Penalty	Number of anticipated operations	Number of delayed operations
1	207	4	2
2	214	2	1
3	211	5	1
4	203.5	3	3
5	209.5	3	2

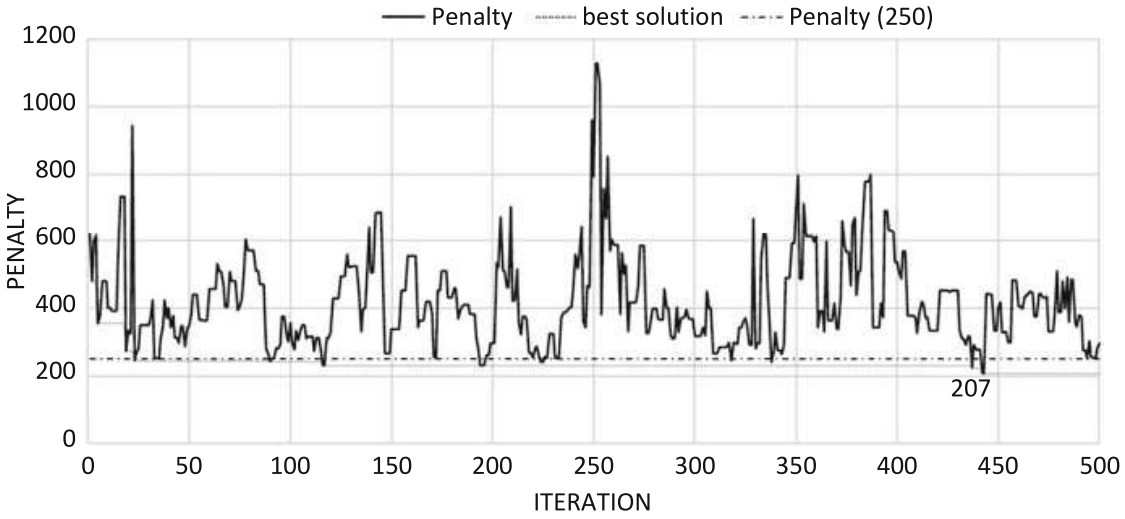

Fig. 9.4 Solutions obtained during the iterative process

solution it can be concluded that it can be effectively used to solve the problem proposed by the case study company.

9.3.5 To Sum Up

Table 9.7 summarizes the considered problems with their specificity. These problems have been chosen because they are from real case study and because they integrate progressive constraints.

Table 9.8 summarizes the previous results. The proposed tool has been used on real data for real use cases. It finds better result than the current solution used by the concerned company. In the hospital case, it has not been used on real data, but has been compared to literature instances, and gives the optimal solution for small instances or a solution close to the lower bound in less than half an hour.

For each of the solved problem, we got the confirmation by the stakeholder that the output given by our tool is correct and useful. The proposed tool is efficient: it finds solutions in a few seconds and minutes. It is general because it can be used to solve many problems. We still have to propose a graphical interface so it will easily be used by any managers. Our tool is efficient on such problems at tactical

Table 9.7 Sum up of the considered problems

Section	Identified problem	NP-Hard	Type	Incompatibilities	Setup	Precedence
9.3.1	Hospital (BPP)	✓	Planning	✓		
9.3.2	Textile (DLSP)	✓	Planning	✓	✓	
9.3.3	Injection (CLSP)	✓	Scheduling	✓	✓	
9.3.4	Metal (JSSP)	✓	Scheduling	✓	✓	✓

Table 9.8 Sum up of the obtained results

Section	Identified problem	Math. model	Size of instances	Resolution time	Results
9.3.1	Hospital (BPP)	Yes	$\leq$ 500 activities 10 resources 1 month	Less than 30 min	A few % than lower bound
9.3.2	Textile (DLSP)	No	$\leq$ 33 products 10 machines 1 month	A few seconds	Better than current solution
9.3.3	Injection (CLSP)	Yes	$\leq$ 200 jobs 10 machines a few days	A few minutes	Close to optimal 30% better than reality
9.3.4	Fridge (JSSP)	No	$\leq$ 50 jobs 6 machines a few days	A few minutes	A lgood feasible solution

and operational level. It is now used to deal with problems in Industry 4.0 context, for example, Reconfigurable Manufacturing Problem.

9.4 Industry 4.0

At a strategic level, the objective is to size systems for the next years, for example, to determine the needed number of resources. But because the variety and quantity of products are not stable and not easily predictable for the next years, new trends must be considered, defined by the Industry 4.0 project. Flexible, agile, reconfigurable, or changeable production systems are needed. Section 9.4.1 gives some definition of such systems, and a projection of how the proposed tool could be used to manage them. Other trends which can also be used are presented by the two next sections: Sect. 9.4.2 is about collaborative robotics and Sect. 9.4.3 about Big Data.

9.4.1 *Agility and Flexibility*

The current market context is characterized by global competition between industries, high product variety, and variable volumes. Those are keys that require the launch of products with a short life cycle and a high degree of customization. To do so, reconfigurable layout has been introduced. One first definition has been given by Lacksonen and Hung (1997) as "the change of an existing plant configuration to another that optimizes costs and time." Since then, many researchers are working on the subject. Benkamoun et al. (2015) summarized in Fig. 9.5 different strategies of reconfigurability.

The proposed tool could be used to solve such problems, where production system should be convertible to face the increasing product variety and extensible to face the increasing volumes. The method first starts with an analysis of the system: what are the demands or future demands and what are the constraints ruling the system. At the opposite of the previous applications, type and number of resources could be a result given by our tool, according to the demand.

Many factories or companies are actually focusing on this problematic. To help them to better understand the proposed solution, some simulation tools could also be used. The optimization part, using the proposed tool, will define one good solution to the problem. This solution is then simulated, using the data or future data of the company, with performance indicators that the company is used to understand. The hybridization between the proposed optimization and simulation can be summarized as Fig. 9.6.

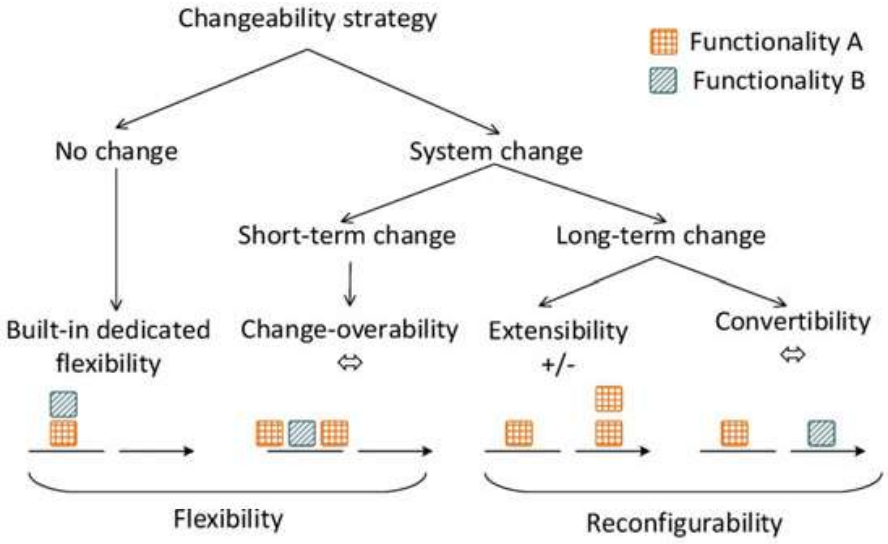

Fig. 9.5 Changeability strategies

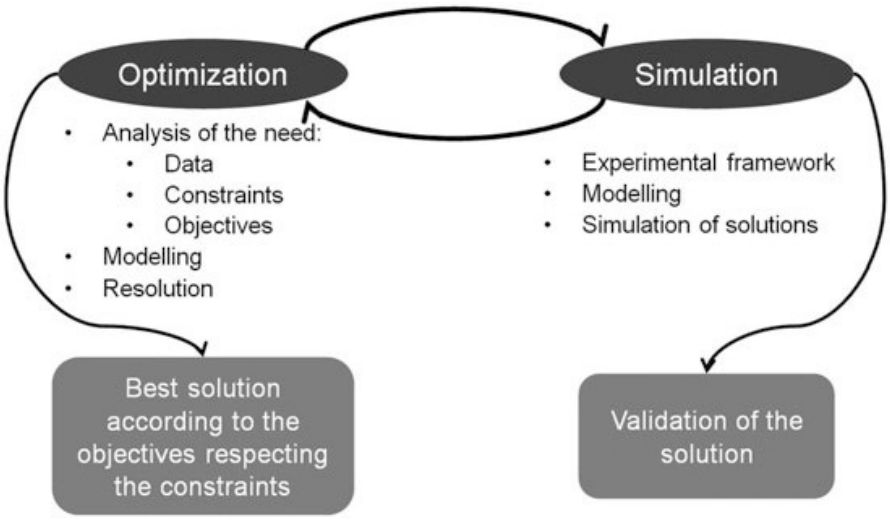

Fig. 9.6 Hybridization: optimization and simulation

9.4.2 Collaborative Robotics in Reconfigurable Manufacturing System

Thanks to collaborative robotics, robots and human can work together. Safety barriers are not needed anymore. Security has increased because these robots are more sensible. Thanks to many captors, they know when the human is close to them. So they can adapt their work and speed. If a contact is made between the robot and the human, the robot stops and needs a human action to start again its work.

These robots are less imposing than traditional robots. Because safety barriers are not needed, these robots can easily be moved from one place to another place of the layout. Mobile collaborative robotics is also existing, where the collaborative robot is put on a mobile platform. In this way, the robot can move independently.

Moreover, these robots are easily configurable. While with a traditional robot or machine, a changing of production needs a changing of tools with setup, a collaborative robot only needs to change its program. The collaborative robot can easily work on one product and then on another one.

Collaborative robotics should be used in reconfigurable manufacturing system. They will be compatible with more products, less setups and shortest setups are needed, they can move from one place to another in the layout and they are smaller than the traditional machines. All these characteristics can easily be modeled and best used with our tool.

This technology attempts to support the reconfigurability of the manufacturing systems and to contribute to adaptive operational conditions. These systems must be responsive to significant changes in demand volume and product mix. The proposed tool has been used to solve an assignment and sequencing problem of reconfigurable assembly lines, where mobile robots collaborate with human workers. The objective is to define a schedule of jobs and decide the allocation of robots to workstations and tasks to minimize the makespan. Preliminary results show that the proposed

methodology can make an efficient robot allocation under high demand variety Maganha et al. (2019).

9.4.3 Big Data

Some new technologies have emerged in the last years: Internet of Things, RFID, Cloud Computing, Mobile Devices, etc. Each company can have much more information about the running of its system: it is called Big Data. Manufacturing Execution Systems exploit Big Data to collect data about the system in real time. Almada-Lobo (2016) discussed about the future of MES. It has to be decentralized, vertically integrated to consider the whole supply chain, mobile and connected, and using the cloud computing. The future evolution of our tool needs to consider Big Data:

- How to collect every available data?
- How to treat these data?
- How to exploit them?

Our tool can be used conjointly with Big Data in two ways. Big Data can feed our tool to generate more entry data to describe the system. The better the system is known, the better the proposed solution will be. Once our tool has built some solution, it can feed the MES. For example, if the proposed tool is used at an operational level, it will build the scheduling of activities day to day. This schedule will be transferred to the required resource. Another use is the response to hazards: if some resources are missing, our tool needs to be efficient enough to compute in real time a new scheduling considering the missing resource. Our tool needs to be developed in parallel to actual new technology such as Big Data.

9.5 Conclusion

Because of the quick evolution of the market, future demand in finished products is highly not predictable in quantity and variety. Production system needs to be adapted to this new fact. To do so, some new definitions of production system have been given: they need to be reconfigurable, agile, flexible, or changeable. But how to design them and how to maintain them?

A generic and modular decision support tool has been proposed in this article. It already solved many problems with a static dimension: the demand is known. After having presented the previous applications, our future work has been given: the use of our generic and modular decision support tool to consider reconfigurable manufacturing system. It can design a new layout, defining the right number of needed resources, considering the future demand. Because this demand is supposed highly variable, some new means of production need to be integrated. Collaborative robotics is one of them, the robots are flexible and may be easily adapted to the future

demand. To size the system, many information is needed to describe the situation. Big Data needs to be integrated to our tool. Once the system has been designed, our tool could be daily used, to schedule the activities, facing the possible hazards. The proposed method can build a good solution in less than a few minutes.

With Industry 4.0 related technology, large amounts of data about industrial processes are available in real time. This increase in data availability and the need to process them in real time make the use of decision support tools more important than ever. DSS has been proposed and developed for more than four decades. One problem that can be pointed out to most of the proposed decision support tools is that they are highly tailored for the problem they intend to solve. This means that they are not flexible and they are hardly adaptable to different kinds of problems. Thus, a long development work is required each time a DSS has to be developed for a new problem.

The tool presented in this paper seeks to overcome these problems. It is made with two modules: a generic one that can be reused by several problems and a specific one to be developed for each specific problem. This characteristic allows for a rapid development of new decision support tools each time a new problem is proposed. In fact, in the recent past, four different problems have been solved with the help of our tool. For each problem we simply need to define and code an adequate list algorithm. Future development will pass by the ability to integrate Industry 4.0 to the proposed tool. We also intend to develop user-interfaces to facilitate the use of the proposed tool by non-programmer specialists.

References

Aarts, E. H. L., & van Laarhoven, P. J. M. (1987). *Simulated annealing: Theory and applications*. Dordrecht: Kluwer Academic Publishers.

Almada-Lobo, F. (2016). The industry 4.0 revolution and the future of Manufacturing Execution Systems (MES). *Journal of Innovation Management, 3*(4), 16–21.

Benkamoun, N., Kouiss, K., & Huyet, A.-L. (2015). An intelligent design environment for changeability management - Application to manufacturing systems. In *DS 80-3 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 3: Organisation and Management, Milan, 27–30.07.15* (2015)

Coello, C. A. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys (CSUR), 32*(2), 109–143.

Druzdzel, M. J., & Flynn, R. R. (2010). Decision support systems. *Encyclopedia of Library and Information Sciences* (3rd ed., pp. 1458–1466).

Gourgand, M., Grangeon, N., & Klement, N. (2014). Activities planning and resource assignment on multi-place hospital system - Exact and approach methods adapted from the bin packing problem. In *HEALTHINF 2014 - Proceedings of the International Conference on Health Informatics, ESEO, Angers, Loire Valley*, 3–6 March, 2014 (pp. 117–124).

Gourgand, M., Grangeon, N., & Klement, N. (2014). An analogy between bin packing problem and permutation problem: A new encoding scheme. In *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World* (Vol. 438, pp. 572–579). Berlin/Heidelberg: Springer.

Gourgand, M., Grangeon, N., & Klement, N. (2015). Activities planning and resources assignment on distinct places: A mathematical model. *RAIRO - Operations Research, 49*(1), 79–98.

Guérin, J., Gibaru, O., Nyiri, E., & Thiery, S. (2016). Learning local trajectories for high precision robotic tasks: Application to KUKA LBR iiwa Cartesian positioning. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE* (pp. 5316–5321). New York: IEEE.

Klement, N., Gourgand, M., & Grangeon, N. (2017). Medical imaging: Exams planning and resource assignment: Hybridization of a metaheuristic and a list algorithm. In *10th International Conference on Health Informatics, Porto, Portugal* (2017).

Klement, N., Silva, C., & Gibaru, O. (2017). Solving a discrete lot sizing and scheduling problem with unrelated parallel machines and sequence dependent setup using a generic decision support tool. In *Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing - IFIP WG 5.7 International Conference, APMS 2017, Hamburg, September 3–7, 2017, Proceedings, Part I* (pp. 459–466).

Lacksonen, T. A., & Hung, C.-Y. (1997). Project scheduling algorithms for re-layout projects. *IIE Transactions, 30*(1), 91–99.

Laurent, A., & Klement, N. (2019). Bin packing problem with priorities and incompatibilities using PSO: application in a health care community. In *Manufacturing Modelling, Management and Control - 9th MIM, Berlin* (pp. 2744–2749). IFAC-online.

Lourenço, H. R., Martin, O., Stutzle, T., Glover, F., & Kochenberger, G. (2002). Iterated local search. In *Handbook of metaheuristics* (pp. 321–353).

Maganha, I., Silva, C., Klement, N., dit Eynaud, A. B., Durville, L., & Moniz, S. (2019). Hybrid optimisation approach for assignment and sequencing decision-making in reconfigurable assembly lines. In *Manufacturing Modelling, Management and Control - 9th MIM, Berlin*. IFAC-online.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems, 15*(4), 251–266 (1995)

McKay, K. N., & Wiers, V. C. S. (2003). Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory. *Computers in Industry, 50*(1), 5–14.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics, 21*, 1087–1092.

Silva, C., & Ferreira, L. M. (2004). Microplano – a scheduling support system for the plastic injection industry. In *E-Manufacturing: Business Paradigms and Supporting Technologies* (pp. 81–89). New York: Springer.

Silva, C., & Klement, N. (2017). Solving a multi-periods job-shop scheduling problem using a generic decision support tool. *Procedia Manufacturing, 11*, 1759–1766. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27–30 June 2017, Modena.

Silva, C., & Magalhaes, J. M. (2006). Heuristic lot size scheduling on unrelated parallel machines with applications in the textile industry. *Computers & Industrial Engineering, 50*(1), 76–89.

Zhu, X., & Wilhelm, W. E. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions, 38*(11), 987–1007.