SAM
Science Arts & Métiers
Archive ouverte Arts & Métiers ParisTech
ARTS ET MÉTIERS ParisTech

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

Solving a multi-periods job-shop scheduling problem using a generic decision support tool

Cristóvão Silva*[a], Nathalie Klement[b]

[a]*CEMUC, Department of Mechanical Engineering, University of Coimbra, Rua Luís Reis Santos, 3030-788, Coimbra, Portugal*
[b]*LSIS CNR UMR 7296 (équipe INSM). École Nationale Supérieur des Arts et Métiers. 8, boulevard Louis XIV – 59046 Lille Cedex, France*

Abstract

In this paper a generic and modular decision support tool developed to solve different planning, assignment or scheduling problems is presented. The utilization of this tool is illustrated by solving a real world multi-period job-shop scheduling problem proposed by a case study company which produces refrigerated foodservice equipment. The case study company problem and a list algorithm developed to integrate the proposed tool for this particular problem are presented. Preliminary results show that the proposed tool can be effectively used to solve the company problem. Besides the problem described in this paper, the proposed tool was used in the past to solve two other problems. Thus, it is demonstrated that the proposed tool can be easily adapted to several different planning or scheduling problems variants, overcoming the lack of flexibility generally associated to more problem-tailored methods proposed in the literature.

Keywords: Decision support tool; multi-period job-shop scheduling; Metaheuristics, List algorithms, Case study.

* Corresponding author. Tel.: +351-239-790757; fax: +351-239-790700.
E-mail address: Cristovao.silva@dem.uc.pt

1. Introduction

Research in scheduling theory has evolved over the past forty years and has been the subject of much significant literature with techniques ranging from unrefined dispatching rules to highly sophisticated branch and bound algorithms and bottleneck based heuristics [1]. Numerous scheduling problems arise in industry and while they are relatively simple in their formulation, they typically involve only sequencing and resource constraints, they remain extremely challenging to solve [2].

The various scheduling problem variants has been widely explored in the literature but most authors still propose highly problem-tailored methods that are not applicable to other scheduling variants. As referred in [1], no solution approach with a guaranteed performance has been developed so far. The same authors state that for most approximation methods there are instances for which they perform badly and it is not completely known under which circumstances a given procedure is likely to succeed or fail.

In this paper a generic decision support tool which was developed to solve many different planning problems is presented. The proposed tool consists of a hybridization of a metaheuristic and a list algorithm. The metaheuristic can be used without any changes independently of the problem to be solved. The list algorithm must be adapted according to the studied problem.

The described decision support tool was already tested with two different planning/scheduling problems: (1) an activities planning and resources assignment problem in a multi-place hospital context [3] and (2) a lot-sizing and scheduling problem with setups and due dates, for a plastic injection company [4]. In both cases good results were obtained with the proposed tool.

The main objective of this paper is to explain how the proposed tool was adapted to solve a new problem encountered in a case study company which produces refrigerated foodservice equipment. This adaptation consists in using the base of the tool developed to the previous solved problems and only develop a list algorithm for the current specific problem. Thus, the intent is to demonstrate that the proposed approach is generic, in the sense that it can support the decision process for several different planning and scheduling problems with a minimum development work.

The paper is organized as follow: Section 2 starts with a description of the case study company and the problem to be solved is characterized. Then, a brief literature review on job-shop scheduling which is the type of problem encountered in the case study company is presented. Section 3 describes the tool proposed to solve the case study company problem, focusing on the list algorithm developed for the specific production planning case considered in this paper. In section 4, based on results obtained using a test instance, the ability of the proposed tool to deal with the case study company planning problem is discussed. Finally, section 5 presents some conclusions.

2. Problem description

2.1. The case study company

The case study company is a manufacturer of refrigerated foodservice equipment, like counters, cabinets, blast chillers, freezers and cold rooms. The products are mainly composed by an external structure, doors, shelves and a refrigeration unit. Some clients will buy company standard products, but the majority will demand a complete bespoke unit. Thus, equipment production will be made in small lots or they will be manufactured one of a kind.

The company shop-floor is composed by two main departments: part production and assembly. The assembly department is composed by two assembly lines where the external structure, the doors and the refrigeration unit (produced in house) and the internal partition elements, like shelves, drawers and baskets (purchased from suppliers) are assembled to obtain the final product. In the part production department, cutting machines and punching systems are used to manufacture external structure and door components from stainless steel sheets.

Based on customer firm orders, the company planner defines a production plan for the two assembly lines. A weekly demand file is generated to the part production department, containing the list of metallic parts required to feed the assembly lines, see Table 1 for an example.

Thus, the weekly demand for parts required to feed the assembly lines represents a list of jobs to be produced containing the following information:

- The part to be produced;
- Their priority: (a) the part must be produced by the required day, (b) the part can suffer a delay of a maximum of one day and (c) the part can suffer a delay of a maximum of two days;
- The quantity to be produced each day of the week;

The sequence of operations for each job, considering the machine and the tool used for each operation and the unit processing time in seconds.

Table 1. Examples of DD assignment rules.

							Operation 1			Operation 2			Operation 3		
Part	Priority	Mon.	Tue.	Wed.	Thu.	Fri.	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			
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

The scheduling problem present in the case study company part production department can, thus, be described as follow. Each component has to follow a processing sequence to be produced and each operation in this sequence requires a given set of resources (machine and tool). The planning horizon is a week which is divided in five periods of one day of 8 working hours. 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, the problem consists in defining the execution of 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 pair machine/tool. 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 is to be produced in a required period (day). A penalty function, composed by two parts, is considered: (1) a storage cost (earliness) if the job is produced in a period prior to the requested one, (2) 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.

The part production department is composed by 6 cutting/punching machines. The number of operations per job varies between one and three. Each operation of a job requires a pair machine/tool to be produced. The tools are not shared among different machines, i.e., each machine has its own set of tools. Table 2 presents an example of the setup times (in seconds) required in a given machine when producing two consecutive jobs.

Table 2. ANN based DD assignment rule input data set.

Machine M1	T11	T12	T13	T14
T11	72	360	300	300
T12	360	72	300	300
T13	300	300	30	240
T14	300	300	240	30

From Table 2 it is possible to observe two important characteristics of the studied problem. (1) there is always a setup time when changing from one operation of a job to a different one, even if they share the same tool and (2) The setup times are quite large when compared with the processing times.

2.2. Literature review

The problem described in the previous section can be classified as a multi-period job-shop scheduling. The multi-period job-shop scheduling problem consists in *T* job-shop scheduling problems which must be solved consecutively trying to access to a makespan less than the available capacity of each period of a given planning horizon. The multi-period job-shop scheduling problem is close to the integration of lot sizing and scheduling problems formulated by [5]. The job-shop scheduling problem (JSSP) is considered as a particular hard combinatorial optimization problem [6]. JSSP has been extensively researched over the last decades since it has several practical applications. Nevertheless, to the best of our knowledge, the only multi-period job-shop scheduling problem with characteristics (objective and constraints) similar to the one presented in this paper is the one presented in [5].

A broad classification of JSSP techniques is presented in [1]. The classification proposed by the authors divides the JSSP techniques in two main groups: approximation and optimization techniques.

Optimization techniques are essentially based on Branch and Bound algorithms, see for example [6, 7], or Mathematical optimization based techniques, like linear or mixed integer programming, see for example [8]. Due to the complexity of the JSSP, the use of optimization techniques has been limited to small size problems [9].

Approximation based techniques were also used to solve the JSSP. These techniques can be divided in two classes: dispatching rules and heuristics, and general algorithms like local search meta-heuristics [10], genetic algorithms [11] or, more recently, artificial neural networks [12].

Since JSSP is a hard combinatorial optimization problem, the multi-period JSSP which implies solving consecutively a set of JSSP is even more challenging. In [1] an interesting discussion about techniques used to solve the JSSP is presented. The main conclusions from this paper can be summarized as follow. Optimal procedures have generally appeared to be unsuitable for the JSSP and most researchers have turned their attention to approximation techniques. Priority Dispatching rules are very popular due to their ease of implementation and their small computational requirements. Nevertheless, they are outperformed by other techniques and, thus, they are usually used as an initial solution technique rather than a complete JSSP solution approach. Artificial Intelligence technique, like Neural Networks, are only suitable for small instances because they require excessive computing time. Local search methods and meta-heuristics have shown to be the most appropriate approximation techniques especially as problem dimensionality increases. no solution approach with a guaranteed performance has been developed so far and more flexible/generic approaches are required.

3. The proposed tool

3.1. concept

The tool proposed in this paper uses a hybridization of a metaheuristic and of a list algorithm. The principle of the method is given in Figure 1.

The encoding used by the metaheuristic is a list Y of jobs. In this project a single solution based metaheuristic, the stochastic descent, is used and, therefore, the neighbourhood system is a permutation of jobs. The list algorithm considers the jobs in the list order to schedule and assign them to the required resource, according to the different constraints. This builds the solution X. The objective function H evaluates the solution X. According to this evaluation, the solution is chosen or not by the metaheuristic. At the end of the running, the solution given by the hybridization is the best list of jobs: the one which optimizes the objective function by applying the list algorithm.

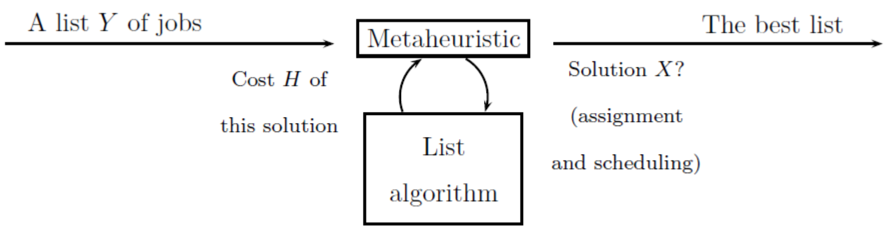
A list Y of jobs
Metaheuristic
The best list
Cost H of this solution
Solution X? (assignment and scheduling)
List algorithm

Fig. 1. Hybridization metaheuristic – list algorithm

3.2. The proposed tool: Metaheuristic

In this project a single solution based metaheuristic, the stochastic descent, was chosen. The stochastic descent is one of the oldest metaheuristic. Its principle is to compute a solution *Y'* which is a neighbor of the current solution *Y*, according to the neighborhood system *V*. After having applied the list algorithm *L*, *X = L(Y)*, if the value of the objective function H(*X'*) is lower or equal to *H(X)* then the solution *X'* so the list *Y'* is accepted. The stochastic descent converges to a local minimum. This metaheuristic was chosen because it was already implemented in previous projects made by the authors and due to it easiness of application, thus, speeding up the tool development process. The objective was to be able to rapidly obtain results to evaluate the ability of the proposed tool to solve the case study company scheduling problem. The encoding of more efficient metaheuristics is envisaged as a development of the current project presented in this paper.

3.1. The proposed tool: List algorithm

List scheduling algorithms are one-pass heuristics that are widely used to prescribe schedules. The standard list scheduling algorithm constructs a schedule by assigning each job in listed order to the first machine that becomes idle [13]. It is important to work with a list algorithm, because the metaheuristic browses the set of solutions by using a neighborhood system which is a permutation between two items in a list. So the used algorithm needs to consider the order of the list to assign jobs to machines and dates.

The list algorithm developed for the case study company problem is composed by two steps presented below in the form of pseudo code. It is important to note that the list of jobs/parts considered by the proposed algorithm is similar to the one presented in Table 1.

First step:

```
For all jobs in the list
    For all operations of the selected job ($j$)
        Assign the selected operation ($i$) to the required machine ($m$) and production day ($d$)
        If operation $i$ is the first one of job $j$
            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 $j$
```

It is possible to observe that in this first step the algorithm ignores the capacity constraints. All jobs are assigned to the required machine in the day they are expected to be concluded. To solve this problem and repair the schedule to respect capacity constraints, after the first step, the algorithm run a second step described below.

Second step:

```
For all days
    For all machines
        While capacity of machine $m$ on day $d$ is violated
        Identify operation $i$ from job $j$ which leads to the capacity violation
            For operation 1 to $i$
        **For** day *d-1* to day 1
            **If** capacity on selected day is available to process selected operation
                Move operation to selected day
                Update schedule of selected day and of day *d*
    **For** operation *i* to last operation of job *j*
        **For** day *d+1* to day 6
            **If** capacity on selected day is available to process selected operation
                Move operation to selected day
                Update schedule of selected day
    Update schedule of day *d*
```

Note that when checking available capacity in a given machine for 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.

In Figure 2 an example of the second step of the proposed list algorithm is presented. After the first step of the algorithm, the schedule presented in Figure 2 (a) 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 *j*, represented in white, violate the capacity constraint for day 3.

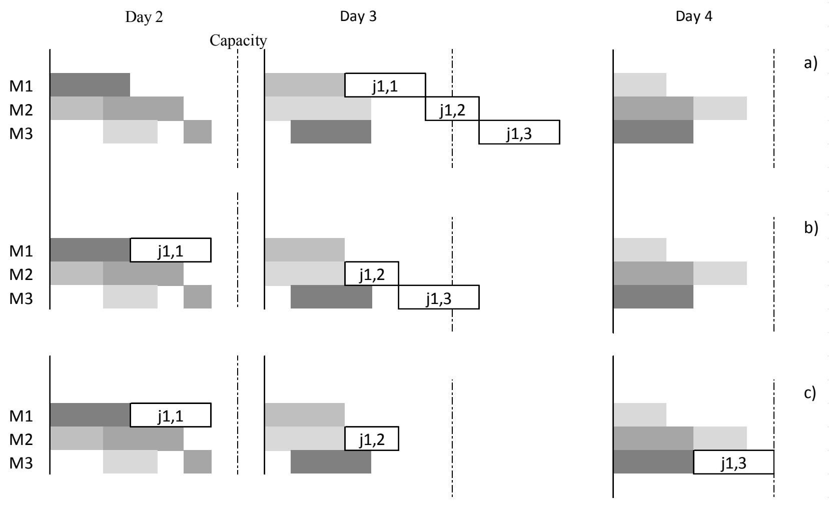

Fig. 2. Example of the application of step 2 from the list algorithm

Operation 1 of job *j*, processed in machine 1, can be moved to day 2 where machine 1 has available capacity. This move leads to the schedule represented in Figure 2 (b). After this move, the problem associated to operation 2 of job *j* is solved, but operation 3 continues to violate the capacity limit. Thus, operation 3 of job *j* 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 periods planning horizon, but the real planning horizon is composed only by 5 periods. Jobs with operations assigned to period 6 will be considered as not produced and will be highly penalized.

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

- Tardiness: priority index x nº of job units x nº of days delayed;
- Earliness: 0,5 x nº of job units x nº of days anticipated;
- Unproduced jobs: 15 x nº of job units

The priority index is considered equal to 3 to jobs with priority (a); 2 for jobs with priority (b) and 1 for jobs with priority (c). As an example, a job with priority (a), representing the production of 5 units of a given part and that is delayed two days, will contribute to the objective function with a penalty of 30 units.

4. Results

As referred previously, the objective of this work was to adapt and test a generic decision support tool to a new problem, a multi-period job-shop scheduling problem proposed by a case study company. Historical company production data show that the weekly production scheduling problem faced by the company, as described in section 2, 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 seconds. 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, we run the proposed tool for 1 000 iterations. Results are obtained in less than 10 minutes. The test was repeated 5 times. Results obtained for each test are presented in Table 3. For all the solutions all jobs are processed within the five 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 3 shows the evolution of the solutions obtained for the first 500 iterations obtained 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 confirm 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 solution it can be concluded that it can be effectively used to solve the problem proposed by the case study company.

Table 3. Results obtained for the considered instance.

Test	Penalty	Nº of anticipated operations	Nº 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

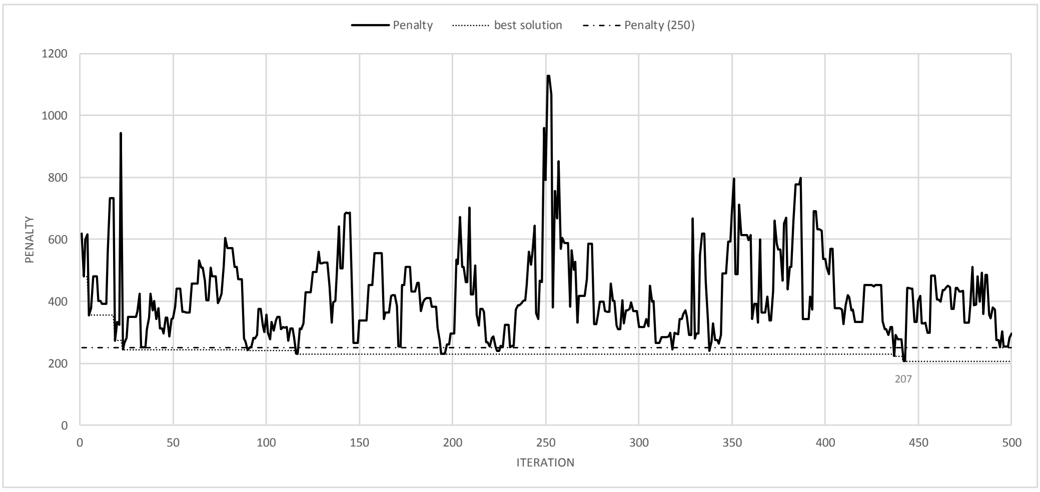

Fig. 3. Solutions obtained during the iterative process.

5. Conclusion

In this paper a generic decision support tool which can be used to solve several different problems was presented. The proposed tool is composed by a generic module, which can be reused for several different problems, and by a specific module, consisting on a list algorithm, which must be specified for each problem. The application of this tool is illustrated in this paper with a real world multi-period job-shop scheduling problem proposed by a case study company.

In the recent past the proposed tool has been used to solve two different problems: an activities planning and resources assignment in a multi-place hospital [1] and a lot-sizing and scheduling problem with setup and due dates, for the plastic injection case [2]. The new problem, presented in this paper, solved by the proposed tool shows its ability to deal with several different problems. The development work needed to adapt the tool to new problems is minimal since the generic part of the tool can be reused. While highly problem tailored-methods might be particularly adequate to achieve good performance, they are difficult to adapt to different planning/scheduling problems. The approach followed by the tool presented in this paper have the advantage of being more generic, allowing to easily adapt it to a large set of planning/scheduling problem variants, without losing sight of the performance.

References

[1] Jain, A.S., Meeran, S. Technical Report. University of Dundee, United Kingdom (1998).
[2] Grimes, D., Hebrard, E., Informs J. Comput., 27 (2015) 268-284.
[3] Klement, N., Gourgand, M., Grangeon, N. 10th International Conference on Health Informatics, 2017.
[4] Silva, C., Klement, N., Gibaru, O. in: International Joint Conference - CIO-ICIEOM-IIE-AIM, San Sebastian, Spain, ICIEOM, 2016.
[5] Dauzère-Pérès, S., Lasserre, J.B. Eur. J. Oper. Res. 75 (1994) 413-426.
[6] Nowicki, E., Smuntnicki, C. Manage. Sci. 42 (1996) 797-813.
[7] Perregaard, M., Clausen, J. Ann. Oper. Res. 83 (1998) 137-160.
[8] Asano, M., Ohta, H. Comput. Ind. Eng. 42 (2002) 137-147.
[9] Harjunkoski, I., Jain, V., Grossmann, I.E. Comput. Chem. Eng. 24 (2000) 337-343.
[10] Noor, S. PhD Thesis. University of Bradford. UK (2007).
[11] Van Laarhoven, P., Aarts, E., Lenstra, J. Oper. Res. 40 (1992) 113-125.
[12] Kumar, P., Kumar, P., Bhool, R., Upneja, V. Int. J. Sci. Eng. Tech. Res. 5 (2016) 1439-1447.
[13] Meeran, S. in: Proceedings of 19th international conference on CAD/CAM robotics and factory of the future. Kuala Lumpur, Malaysia. 2003.
[14] Zhu, X., Wilhelm, W.E. IIE Trans. 38 (2006) 987–1007.