Acta Simulatio - International Scientific Journal about Simulation
Volume: 1 2015 Issue: 4 Pages: 1-4 ISSN 1339-9640

BASIC PRODUCTION SCHEDULING CONCEPT SOFTWARE APPLICATION IN A DETERMINISTIC MECHANICAL PRODUCTION ENVIRONMENT

Róbert Jurčišin
T-Systems Slovakia s.r.o., Žriedlová 13, 040 01 Košice, robert.jurcisin@t-systems.sk
Juraj Šebo
Technical University of Kosice, Nemcovej 32, 042 00 Košice, juraj.sebo@tuke.sk

Keywords: Production Scheduling, Permutation Flowshop, Deterministic Mechanical Production, software
Abstract: This paper deals with production scheduling in a deterministic mechanical production environment. We focus on minimizing the objective function makespan using local search. Our aim is to present developed software solution with implemented basic scheduling concepts and our approach to scheduling in deterministic mechanical production environment on the simple case of 4 jobs and 5 machines. The outputs of solution are presented in the figures and they gives reasonable results. As we want to use combinatorial optimization we have implemented a heuristic approach. This heuristic is known as a Local Search Method. This method is commonly used in scheduling.

1 Introduction

This paper deals with production scheduling in a deterministic environment of mechanical production where there are different components processed on different kinds of machines in variable batches according to received orders. Our aim is to find the order of the jobs which has the minimal flowtime of production. In other words we focus on minimizing the makespan. The makespan, defined as max (C1,.....,Cn), is equivalent to the completion time of the last job to leave the system. In the first part of the paper we are concerned with basic concepts of cheduling and permutation flowshops. As we want to use combinatorial optimization we have implemented a heuristic approach. This heuristic is known as a Local Search Method. This method is commonly used in scheduling.

Local search is based on the idea that the current solution can be improved by making a small change. With every single improvement of the solution we are trying to find a better solution. Each of the combinatorial – optimization problems which we solve consist of a certain number of specific elements which describe or define the problem. There are: objective function (optimization criteria) f: $\Pi \rightarrow R$, simple rule generating transition from one solution to another solution and a space of $\prod$ solutions which is determined by some constraints. Sometimes the set of elements can be named the configuration of the problem. It is our aim to find optimal solution π^* where $f(\pi^*) \leq f(\pi)$ for each π Π (see e.g. [6]).

In basic local search we move to the best neighbour if it improves the objective value, and if no neighbour is better than the current solution, we terminate the search. The problem with this strategy is that the search is easily trapped in a local minimum.

2 Basic concepts of scheduling

Fundamental notations in the scheduling problems are machine, operation and job.

Operation o is a basic technological action which can not be divided into more particular actions.

Job J is an operation sequence {o1, o2,...., og} which must be done within one order.

Machine M is a piece of equipment with the ability to carry out one or more operations. In some literature we can find the notion processor instead.

By scheduling problems we need to determine for each operation oij of each job Ji from given set of jobs J= {J1, J2,…., Jn} and machine Mj from set of machine M = {M1, M2,…., Mm} which carry out the operation oij and the time interval of that operation (or more time intervals if the preemption is allowed) (see e.g. [1]).

We assume two elementary conditions in most of the scheduling models [1]:

1. Each machine can carry out only one operation in one time interval,

2. In one time interval it is not possible to carry out two or more operations of one job.

From a scheduling point of view the operation is not preemtable. Some of the models allow the preemption and then continuing in this operation. In most of the models the operation continues where it had been interrupted. There are some models where we need to start from the beginning. Machines from the set of machines M can be universal or specialized. Universal machines can process any operation of any job. Individual machines of set M can process these operations with different processing times. A part of the solution of the scheduling problem can be assigning of machines to the operations. Specialised machines are determined to process only certain operations – each of the operations has been

Volume: 1 2015 Issue: 4 Pages: 1-4 ISSN 1339-9640

assigned the machine. The machine does not need to be available at anytime. As a result of changing the shifts, regular maintenance or other industrial matters there can be some periods when a machine can not carry out any operations. A very important characteristic of each machine is a time of its availability. We assume job J = {o1, o2,….,ok}. There are some scheduling problems where it does not depend what the order of the operations is. On the other hand there are scheduling problems where the order of the operations must be kept because of technology of production. We can define it by using the precedence relations between the individual operations. When operation y can start after operation x is finished then we will say that operation x prevents the operation y and denote it $x \prec y$. Relation $\prec$ is transitive, i.e. if $x \prec y$ and $y \prec z$ then $x \prec z$. We will assume that operation x directly prevents operation y if $x \prec y$ and there is no such operation z that $x \prec z \prec y$. This fact will be denoted as $x \prec\prec y$. There are several jobs given by the scheduling problem. These jobs are denoted as follows J1, J2,...., Jn. Each of the jobs consists of certain number of operations oi1, oi2,....., oigi. Further there is given the set of machines M= {M1, M2,....., Mm}. For each operation oij of any job i it is given time period p(oij) and machine $\mu ij \in M$ which can process this operation [1].

To create the schedule J1, J2,....., Jn where job Ji consists of operations oi1, oi2,.... , oigi means that one (in case when preemption of operations is not allowed) or more (in case when preemption is allowed) time intervals (b1x,c1x), (b2x,c2x),...., (bqxx,cqxx) must be determined for each operation oij = x where $bix \leq cix$ (in which the operation x will be processed on the given machine) and that [1]:

1. $\sum i\ (cix - bix) \geq p(x)$ - sum of partial time periods of processed operation x is ≥ as its length,
2. If x, y are two operations where for $x \prec y$ then $c1x \leq b1y$ – when operation x prevents operation y then processing of operation y will not start earlier than x is finished,
3. Each of intervals (bix, cix) will be from the available machine interval.

We consider the scheduling problem to be given as [1]:

1) Set of machines M= {M1, M2,…., Mm},
2) Set of jobs J = {J1, J2,…..,Jn} and for each job Ji there is assigned an operation sequence oi1, oi2,…., oigi and function F which assigns a certain machine to each operation to carry out this operation
3) Precedence relation $\prec$ on a set of operations,
4) Criterion function of the schedule.

Output data from the scheduling criteria [1]:

From a given schedule the following parameters can be computed for each job Ji:

- Ci completion time of job Ji,
- Fi time period of job Ji in the system ,
- Wi total period of job´s Ji waiting in the system,
- Li = Ci - di time deviation of planned time of finished job Ji,
- Ti = max{0, Li = Ci – di} tardiness of job Ji,
- Ei = max{0, - Li} earliness of job Ji,
- Ui = 0, if $Ci \leq di$, or Ui = 1. Penalty unit per job Ji

We can see that parameters Li, Ti, Ei, Ui are defined as a function of the completion time Ci. The period Fi of job Ji in the system can be described as follows Fi = Ci - ri where ri is a time of job Ji entering the system. If pi is a period of processing of the job Ji then its total period of waiting in the system is Wi = Fi - pi = Ci - ri- pi. All the above mentioned output parameters are functions of the parameter Ci [1].

3 Permutation flowshop

Where can we encounter the flowshop environment? How can we define it? The flowshop scheduling problem is a very active field of research which is almost 55 years old. When each job has to undergo a series of operations in many manufacturing and assembly facilities then we consider it to be the flowshop. These operations have to be done on all jobs in the same order and the jobs have to follow the same route. Buffers are considered to be very important elements in the real production scheduling. We can regard the buffer as a place where we need to keep some of the components for a while or a longer time. Sometimes the storage or buffer space in between successive machines may have for all practical purposes virtually unlimited capacity. When the products that are being processed are physically small, making it relatively easy to store large quantities between machines then it is unlimited. On the other hand, when we consider physically large components then the buffer space in between two successive machines may be limited, then the blocking should be taken into account. Blocking is considered when the buffer is full and the upstream machine is not allowed to release a job into the buffer after completing its processing. Having buffers of zero capacity, a job i just finishing on machine r cannot advance to machine r + 1 if this machine is still processing job its predecessor in the job sequence. The job i must remain at machine r. We can consider thus temporarily denying machine r job its successor in the job sequence until job i can advance to machine r + 1. Abadi and Sriskandarajah (1995) described the blocking flowshop problem as „the flowshop has no intermediate buffer therefore a job cannot leave a machine until the next machine downstream is free" [3].

Flowshops that do not allow sequence changes between machines are called permutation flow shops. In these flow shops the same sequence or permutation of jobs is maintained throughout. In another words order of machines is fixed and only the jobs are allowed to change the order because of permutations. Unlike the classical flowshop problem where the compu tional problem is $(n!)^m$, permutation flowshop problem is in that case only n!. We are not able to solve big instances by searching

Acta Simulatio - International Scientific Journal about Simulation
Volume: 1 2015 Issue: 4 Pages: 1-4 ISSN 1339-9640

BASIC PRODUCTION SCHEDULING CONCEPT SOFTWARE APPLICATION IN A DETERMINISTIC MECHANICAL PRODUCTION ENVIRONMENT

Róbert Jurčišin; Juraj Šebo

each permutation in any case. It would not be sensible. For many optimization problems there are very simple methods of scheduling problems. [4]

Given a permutation schedule j1,....,jn for an m machine flowshop, the completion time of job jk at machine i can be computed easily through a set of recursive equations (see e.g. [1]).

We can also consider another approach to evaluation of the makespan. It can be computed under a given permutation schedule by determining the critical path in a directed graph that corresponds to the schedule [5].

4 Our approach and software application

For the simplest algorithm of the production scheduling of mechanical production we can use as an input of jobs Si, $i \in \{1,\ldots,m\}$ according to the order in which they come to the production system and on which machines VZj, $j \in \{1,\ldots,n\}$ are assigned (its priority in the system is assigned by the principle of FIFO (First In – First Out) from the assigning matrix and for calculation is strictly given) [2].

We start in time zero with the assignment of jobs to the machines whereby the algorithm works iteratively. We store the shortest production time of the particular operations in each iteration and in the end we sum up all the "the shortest production times" $\sum_f T_k \rangle \max$; $f \in k \ \forall k_{S_i}$ and this represents the makespan [2].

We can see from the figure 1 below how the original algorithm works. We have 5 jobs and 4 machines in the production system. We do not consider a limited buffer in this example so we use an unlimited buffer for each machine [2].

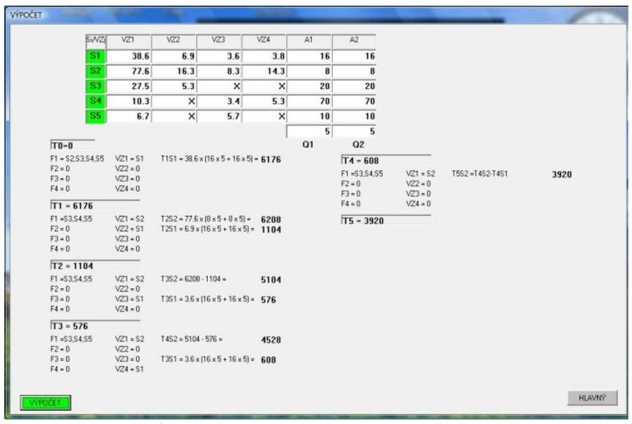

Figure 1 Softvare calculation with 5 jobs and 4 machines

In the figure 2 and the following we can notice particular steps in a software application which has been developed for testing scheduling and optimization techniques.

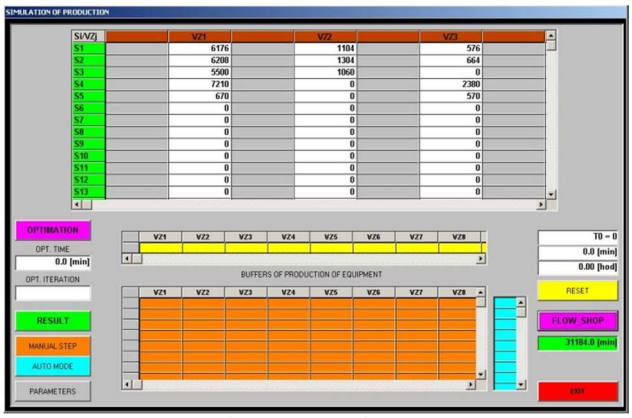

Figure 2 Aplication of permutation flowshop with local search

In the figure 3 we can see a demonstration of permutation flowshop with local search heuristic. There is only the small example with 5 jobs. In this application it is possible to use manual mode where we can see step by step what is going on inside the system. This manual mode uses the original algorithm which works on the principle i.e. „time jumps" which is very similar to jobshop problem.

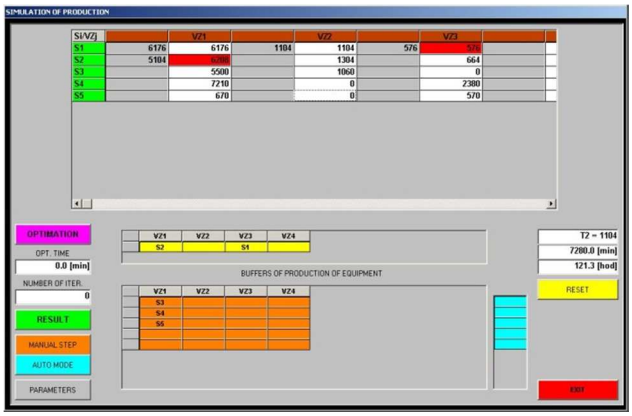

Figure 3 Aplication of permutation flowshop with local search (manul mode)

In the figure 5 we can see evaluation of the results within local search. We have demonstrated it with 10 jobs and can compare it to brute force which is 10! with its 3 628 800 permutations. It is clear it would not be sensible to compute it without using combinatorial optimization.

By using combinatorial optimization we have solved only 45 neighbourhoods from defined neighbourhoods within the first iteration with starting sequence of jobs.

Acta Simulatio - International Scientific Journal about Simulation
Volume: 1 2015 Issue: 4 Pages: 1-4 ISSN 1339-9640

BASIC PRODUCTION SCHEDULING CONCEPT SOFTWARE APPLICATION IN A DETERMINISTIC MECHANICAL PRODUCTION ENVIRONMENT

Róbert Jurčišin; Juraj Šebo

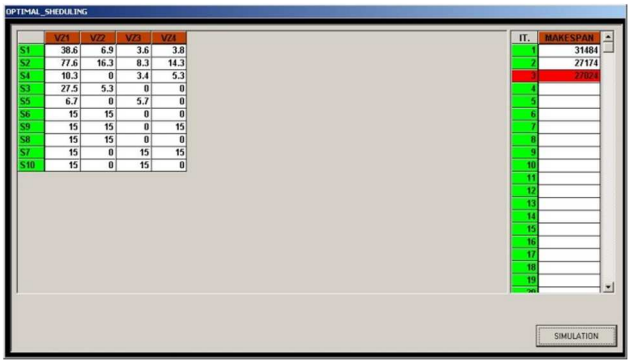

Figure 3 Aplication of permutation flowshop with local search (results)

Conslusion

A survey in conventional mechanical enterprises in Slovakia and abroad has shown that there are opportunities by managing of the production gaps in the practical and user friendly application software which would be used in practise.

On the base of standard scheduling concept and with the notion of real production constrains we have developed the software and test our approach of makespan minimization on simple cases (e.g. of 5 jobs and 4 machines). The outputs of solution are presented in the figures and they gives reasonable results. In more coplex test of software solution with 100 jobs and 100 machines it takes 45 minutes to find the optimum. It is important to mention that the searching time is obviously related to the methods used, but is also dependent on the hardware of used computers.

Acknowledgement

This article was supported by the state grant agency KEGA research grant 004TUKE-4/2013 *"Intensification of modeling in education of II. and III. degree in the field of study 05/02/52 Industrial Engineering"*.

References

[1] PALÚCH, S.: *The graph theory*, Zilina, University of Zilina, Slovakia, 2001. (Original in Slovak)

[2] ŠEBO, D.: "*Thesis of the inaugural lecture to promotion to Professor of Industrial Engineering: Logistics in the Managerial activities*", Technical University of Kosice, Faculty of Mechanical Engineering, Kosice, Slovakia, 2007. (Original in Slovak)

[3] HEJAZI S.R., SAGHAFIAN S.: Flowshop-scheduling problems with makespan criterion: a review, *International Journal of Production Research*, Vol. 43, No. 14, p. 2895-2929, Taylors & Francis Group Ltd., 2005.

[4] PINEDO, M.L.: S*cheduling: Theory, Algorithms, and Systems*, Springer Science + Business Media, LLC, New York, NY, 2008.

[5] NAWAZ, M., ENSCORE JR., E.E., HAM, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA, *The International Journal of Management Science*, Vol. 11, No. 1, p. 91–95, 1983.

[6] HOOS, H.HOLGER, STÜTZLE, T.: *Stochastic Local Search, Foundations and Applications*, Morgan Kaufmann Publishers, Elsevier Inc., 2005.

[7] JURČIŠIN, R.: "*Examination paper of the thesis: The suggestion of the optimization criteria of scheduling algorithm in mechanical production*", Technical University of Kosice, Faculty of Mechanical Engineering, Kosice, Slovakia, 2008. (Original in Slovak)

[8] ŠEBO, D., JURČIŠIN, R.: *Algorithm of simulation and optimalization of conventional mechanical production.* In: Zbornik radova: Proceedings: 32nd conference on production engineering of Serbia with foreign participants, 2008.

Review process

Single-blind peer reviewed process by two reviewers.