ICCSPA20 1570672384

The parallel machine job splitting and allocation

Samah A. M. Ghanem
FEUP, University of Porto
Porto, Portugal
Email: samah.ghanem@fe.up.pt

***Abstract*—In this paper, we propose novel optimal / near optimal approach to model and solve the time-dependent job splitting and allocation over parallel machines by exploiting the iterative structure of the model and the search. We compare global search deterministic method to genetic algorithm hueristic.**

***Index Terms*—Job splitting and allocation; Genetic Algorithm; Hueristics; Optimal schedule; Parallel Machine; Processing Time; Scheduling.**

I. Introduction

Scheduling problems in industrial engineering decompose into two main components: the machine configuration and the job characteristics. Generally, machine configuration is categorized into single machine, parallel machines, flow shop, and job shop settings. In their comprehensive review, Cheng and Sin [1] listed five characteristics of a job: job processing time, due-date requirement, preemptive sequencing, precedence constraints, and job release time. The first two job characteristics are self-explanatory. The third characteristic allows an operation of a job to be interrupted and the machine is taken over by another job that is considered to be more urgent. The precedence constraints determine the order in which the jobs have to be processed. In a scheduling problem, if the jobs are released at different times, the condition is called dynamic. Otherwise, it is a static condition. Similar to jobs, machines may be released at different times, which imply dynamic machine availability. Thus, the static and dynamic terms apply to both job allocation and machine reuse.

Parallel machines can be categorized into three types: equivalent, deterministic and uncorrelated parallel machines. The difference between these parallel machines systems is characterized by a job's processing time among the machines in parallel. In equivalent parallel machines system, the processing time of a job is the same on all machines in parallel. In deterministic parallel machines, each machine has a unique speed factor that determines jobs' processing time. Thus, the processing time of a job on each machine varies by the speed factor of the machine. In uncorrelated parallel machines, the processing time of a job varies arbitrarily between the machines.

Uncorrelated parallel machines are very common in the industry. A company may invest in similar machines that have different capability, taking into consideration the capital cost, operation cost, and variation in production demand. Therefore, scheduling tasks on uncorrelated parallel machines is of particular relevance to the scheduling in industrial systems.

Jobs that compete for limited resources, over a set of uncorrelated parallel machines, may have different levels of priority and due date. Factors that contribute to setting due date of jobs are many, some of which are customer demands, machine capacity, and machines congestion. A job with tight due date, high priority, and/or high workload, may need to be split and processed on two machines in parallel. The need for splitting jobs typically appears in an operation that imposes large workload on a machine and requires the entire job completed before the next operation can be started. Thus it is conceivable that the operation following the one performed on split -mode requires a fairly reasonable processing time that it can be performed on one unit of machine. It means that the split portions of the job would need to be combined into one and moved over to the next machine on which the job's operation is scheduled to be performed. Schuften and Leussink [2] used a branch- and -bound algorithm to solve identical parallel machines scheduling with dynamic job release dates, general due dates, and family setup times. The objective was to minimize the maximum lateness of all jobs released. They compared the performance of applying two methods of lower bound to the branch and -bound algorithm by Carlier [3].

Glass et al. [4], and Piersma and Van Dijk [5] applied local search heuristics to solve the job - scheduling problem on uncorrelated parallel machines. The objective was to minimize the maximum completion time.

Tabu search has shown remarkable success in solving production - scheduling problems, as it may allow the desirable consideration of a scheduling problem with multiple objectives. Suresh and Chaudhuri [6] considered minimizing the maximum tardiness and minimizing the makespan simultaneously on uncorrelated parallel machines. They employed a Tabu search algorithm to solve the problem. Via a mixed integer programming optimization formulation that minimizes the weighted tardiness in [7] and [8] or via minimal completion time [9], the authors also proposed Tabu search algorithms besides others to solve the problem, providing optimal/near optimal solutions.

Almost all state of the art work of parallel machines problems used the hypothesis that each job can be processed on at most one machine at a time although preemption is allowed. Lawler et al. [10] gave a comprehensive review about PMS in a classffication of regular performance, like minimizing sum criteria, minimizing maximum criteria and precedence constraints. Cheng and Sin [1] also gave a comprehensive review on parallel machines problems according

978-1-7281-6535-6/20/$31.00 ©2020 IEEE

to the viewpoints: completion-time-based, due-date-based and flow-time-based performance measures. Lam and Xing [11] gave a short review of new developments of parallel machines associated with the problems of just-in-time (JIT) productions, preemption with setup and capacitated machine scheduling.

This paper addresses scheduling of jobs over uncorrelated parallel or partially parallel machines without the usual hypothesis that eacj job can be processed at one machine at a time. The processing time of each job would be different on machines of different types that constitute to the uncorrelated-parallel machine environment. Splitting jobs is considered in this research by utilizing a framework that optimizes the weights that allow a split to exist or not to exist at a portion of time in the machine. Therefore, the number and the size of jobs that need to be processed in split-modes, as well as the number of machines are fundamental parts of the of the scheduling decision. Each split portion of a job will be considered as a separate job but with time dependency to the previous parts of the formerly splitted job.

The objective of this paper is to find optimal/near- optimal schedule that minimizes the sum of the maximum processing times of time-dependent jobs on parallel or partially parallel machines. Such objective is important as it allows for an optimal/near optimal job splitting/ordering/scheduling that minimizes the completion time. Besides, it breaks the parallel machine scheduling problem into less complex iterative/cyclic/geometric formulation. The assumptions are that scheduling environment is dynamic in both job timing allocation and machine availability. Besides, we allow a job to be split and run over multiple machines, however, taking into consideration the time-dependency in the order of the job(s).

The main contributions of the paper are four fold: 1. The paper formulates the problem based on novel iterative or cyclic sturcture of the time due to job-splits per job time-dependency for partially parallel machines, and uses such framework for formulate fully parallel machines. 2. The formulation is shown as a solution represenation that have geometric representation, through which a clear insights can be extracted to adapting the scheduling of partially parallel machines to the case of fully parallel machines. 3. The paper associates to the formulation a probabilistic framework that allows for joint optimization of multiple objectives, one is the completion time via optimal job ordering, and another is by weighting the times, thus splitting the jobs over parallel machines in such a way that yet minimizes the completion time. 4. The paper provides a comparison between global search method to the genetic algorithm heuristic method.

The remiainder of the paper is organized as follows, section II introduces the modeling and problem formulation. Section III introduces illustrative results and evaluation of deterministic and hueristic approaches. Finally, Section IV concludes the paper.

II. $N/M/PMS$ Modeling and Problem Formulation

We consider $N/M/PMS$ an N-jobs, M-machines, parallel machine scheduling problem. To minimize the complexity of the problem, we rely on the iterative structure of any solution representation set, and the time-dependency between jobs to model the job processing over multiple machines, where parallel jobs are run after the first job takes place (for the formulation of the problem). The maximum span will lead to $M \times M + N - 1$ time steps despite the difference in the timing of each job expressed as elements on each column of the timing matrix. Therefore, to this end, its clear that an optimal schedule will -geometrically- shift back such a structure to be all run in parallel over all machines, which corresponds to a timing of size $M \times N$. While a completion time should correspond to the sum of the maximum of each column (parallel processed jobs in the time-dependent sequence).

A. The Completion Time

To derive the completion time of the $N/M/PMS$ scheduling, first we construct a timing matrix that expresses parallel jobs and their dependency across machines as follows:

$$T = \begin{pmatrix} T_{1,1} & T_{1,2} & \cdots & T_{1,N} & 0 & 0 & 0 & 0 \\ .. & .. & T_{2,3} & \cdots & T_{2,N} & 0 & 0 & 0 \\ .. & .. & \vdots & \ddots & \vdots & \vdots & 0 & 0 \\ .. & .. & .. & \vdots & T_{M,1} & T_{M,2} & \cdots & T_{M,N} \end{pmatrix} \tag{1}$$

The timing matrix T is of size $M \times M + N - 1$, and the non-zero element $T_{i,j}$ corresponds to the time of job or job (or job split) j over machine i. To clarify, we provide the following example.

Example: The timing matrix for the $3/3/PMS$ can be written as follows,

$$T = \begin{pmatrix} T_{1,1} & T_{1,2} & T_{1,3}0 & 0 & \\ ... & T_{2,1} & T_{2,2} & T_{2,3} & 0 \\ ... & ... & T_{3,1} & T_{3,2} & T_{3,3} \end{pmatrix} \tag{2}$$

Therefore, we can construct a simple guess about the job scheduling, that follows a structure that takes into consideration the time dependency of the processes while preserve the parallel runs over all machines, therefore, intuitively, the allocation over machines can be thought of first job over each machine at the same time. However, we are yet not going so far to the splitting for optimal/near optimal scheduling. One non-unique solution representation can be written as,

$$T = \begin{pmatrix} T_{1,1} & T_{2,2} & T_{3,3} \\ T_{1,2} & T_{2,1} & T_{3,2} \\ T_{1,3} & T_{2,3} & T_{3,1} \end{pmatrix} \tag{3}$$

Notice that the first timing matrix with the solution representation in (2) expresses a partially parallel machine (PPM) with time-dependent sequences per job-splits. The second timing matrix with the solution representation in (3) expresses a fully parallel machine (PM) with time-dependent sequences per job-splits. The swap in the second and thrid columns entries of in (3) is to provide intuitions on the time-dependency that yet can exsist in a fully parallel machine. Most importantly is to notice that the PPM formulation and solution framework differs in core points from that of PM. In particular, both systme

coincide if the jobs have equal time, thus, the scheduling problem is not anymore a burden or if the PPM is optimized accross all possible permutations of of jobs and machines at which we preserve time dependency and parallel constraints, however, we forbid certain swap moves that would break the time-dependency requirment. Further, its worth to notice the geometric structure that transfers a parallelogram in time to a rectangle or square under a geometrical rotation which is translated by a swap move. We focus on this paper, on the PPM, future research will consider the same framework adapted to the PM problem.

B. Problem Formulation

We define a splitting strategy of jobs over time and over machines, taking into consideration a minimal delay. Therefore the completion time of such model reads as;

$$CT^{PPM} = \sum_{j=1}^{M+N-1} \max\{T_{i,j}\}, \forall i = 1, .., M \tag{4}$$

Where PPM is partially parallel machine as in (2), and for the full parallel ordered machines, the completion time reads as,

$$CT^{PM} = \sum_{j=1}^{N} \max\{T_{j,i}\}, \forall i = 1, .., M \tag{5}$$

Where PM corresponds to fully parallel machines, running jobs in parallel, as in (2).

C. Partially Parallel Machines

We focus on the PPM and define the following optimization problem defines the allocation with splitting constraints,

$$\min_{\pi(\mathcal{P}:T_{i,j})_{p_a^j(m_i)}} CT^{PPM} \forall j = 1, .., N, \forall i = 1, ..., M \tag{6}$$

Subject to,

$$\sum_{i=\ell}^{M} T_{i,j} = Tj, \forall j \in 1, ..., N \tag{7}$$

$$\sum_{j=\ell}^{N} T_{i,j} = Time - Window, \forall i \in 1, ..., M \tag{8}$$

$$Time - Window \geq T_{i,j} \geq 0, \forall i \in 1, ..., M, \forall j \in 1, ..., N \tag{9}$$

The constraints in (8) enforce parallel processing due to fixed size of the window where the jobs need to be consied in. We also define constraints (7) that define each demand. Further, the constraint (9) allows for zero or positive allocation over time fragments. To express the association of weights or probabilities of job allocation over a machine by $p_a^j(m_i), \forall j = 1, .., N, \forall i = 1, .., M$ that vary to allow optimal/near optimal splitting, we can re-write the constraint in (7) and (8) respectively in a more precize way as follows;

$$\sum_{i=1}^{M} p_a^j(m_i) Tj = Tj, \forall j \in 1, ..., N \tag{10}$$

$$\sum_{j=1}^{N} p_a^j(m_i) Tj = Time - Window, \forall i \in 1, ..., M \tag{11}$$

Thus, consequently, the following constraints are included,

$$\sum_{j=1}^{N} p_a^j(m_i) = 1, \forall i \in 1, ..., M, 1 \geq p_a^j(m_i) \geq 0 \tag{12}$$

Therefore, this allocation and splitting are joint parameters that allow for minimizing the completion time.

D. Fully Parallel Machines

The fully PM will have the following optimization problem to define the allocation with splitting constraints,

$$\min_{\substack{\pi(\mathcal{P}:\rho(T_{i,j},T_{\ell,k}))\\ p_a^j(m_i)}} CT^{PM} \forall j = 1, .., N, \forall i = 1, ..., M, \forall k > j \tag{13}$$

Subject to the same constraints in (9), (10), (11), and (12), and $\pi(\mathcal{P} : \rho(.,.))$ does not only express all possible permutations of job splits over machines but also all possible moves or swaps between jobs that yet preserves the time dependency in per job-splits of jobs over the parallel machines.

For each possible permutation along jobs over machines where $N!$ permutation exsits, there yet exist $(N \times M)!$ possibilities of orders that correspond to possible swaps. However, we can forbid, or make it a Taboo to conduct any swap of the following shape:

$$T_{i,j} \leftarrow T_{i,j+k} \forall i = 1, .., M, \forall j = 1, ..., N \tag{14}$$

The following section addresses the implementation of the PPM, and establishes a comparison between a global search deterministic method to Genetic hueristic algorithm. Future research will consider addressing the implementation and evaluation of soultions for the setup of fully PM.

III. Simultaiton Results

This section address the implementation and evaluation of the optimization problem of PPM. The evaluation is done using MATLAB 2015. The evaluation is done for a small size problem for 4 jobs, 4 machines, with 4 jobs splits. The result is a vector of 16 optimal weights or probabilities that allows splitting of jobs jointly with finding optimal job ordering over all possible permutations of the jobs over machines. The objective function is the completion time as mathematical framework discussed.

The sizes of the jobs used are as follows: $Tj1 = 15; Tj2 = 8; Tj3 = 20; Tj4 = 6;$. The $Time - Window = 13$ that explains the window of runs over all machines is constrained to its given value. Therefore, the basic steps of the algorithms are as follows:

step1: Define an inital probability of allocation $vector(x) = 0.25 \forall x(1), ..., x(16)$ where $4! = 16$.

step2: Define the job splitting over machines with vectors: $Job_{j+s-4} = (x(1 + s), ..., x(4 + s))Tj + s - 4, s = \{5, 6, 7\}, j = 1;$

step3: Construct a matrix of jobs based on step2.

step4: Call objective by an optimization solver (fmincon for global search with gradient based solvers, or GA for genetic algorithm heuristic).
step5: The constraints are as defined in the previous section.
step6: The objective function generate all possible permutations of jobs orders from 1,2,3,4 and associate job times.
step7: Construct the timing matrix T based on one permutation of step2.
step8: Find the maximum at each coulmn of matrix T, the sum all maximums to find the completion time.
step9: Find the completion time and job sequence that corresponds to the minimum of completion times at step4.
step10: The solver returns values of vec(x) that allows for minimum fval (minimum completion time), then after a certain number of iterations it converges to a solution.

The solution using deterministic (fmincon) search is as follows:
$x = [0.4521; 0.0392; 0.0349; 0.4567; 0.2710; 0.0574; 0.0742; 0.2636; 0.0009; 0.5003; 0.4961; 0.0017; 0.2724; 0.1890; 0.1900; 0.2732]$

that corresponds to the weight of each job split from 1 to 4 that satisfies the total job size of each job, i.e., 4 entries for each Job. The corresponding minimal completion time is: $fval = 32.178$. This setup provides timing of $14.7433; 5.3295; 19.9794; 5.5473$. Better results might be obtained with different initial value of x in step1.

The solution using genetic algorithm heuristic (GA) search method is as follows:
$x = [0.3311; 0.0000; 0.2424; 0.0157; 0.0000; 0.0065; 0.0000; 0.8098; 0.3589; 0.2107; 0.0422; 0.0000; 0.0000; 0.0997 \; 0.4794; 0.0000]$

The corresponding minimal completion time is: $fval = 17.2028$. This setup provides timing of $8.8389; 6.5300; 12.2363; 3.4744$. Despite the results of GA provides much less completion time, however, we see that the solution does not lead to fullfilling all job demands. Besides, we see that deterministic approaches tend to dostribute the weights evenly, while the GA distribute that weightes in a more radical way, where the zeros in the $vector(x)$ imply that lots of jobs are not associated with any resources over certain machines.

It is clear that the deterministic method using fmincon outperforms GA in two sides, one is the precize allocation of jobs, and that the jobs always fullfill their total time requirments.

IV. Conclusion

This paper addresses the parallel machine job splitting and optimal ordering scheduling problem to minimize the sum of the maximum completion time over all jobs across all parallel machines. The paper presents a novel formulation for the problem of PPM and PM, and provides evaluation of deterministic and heuristic approaches in solving such setup. It is clearly shown that deterministic approaches provides well suited solutions than heuristic methods. However, for problems of large size in terms of number of jobs and number of machines, it might be better to consider near optimal heuristics that can reduce the computational complexity. Future research will consider the fully PM with the prohibited moves discussed in a Taboo search heuristic.

References

[1] T. Cheng and C. Sin, "A state-of-the-art review of parallel-machine scheduling research," *European Journal of Operational Research*, vol. 47, no. 3, pp. 271 – 292, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037722179090215W

[2] J. Schutten and R. Leussink, "Parallel machine scheduling with release dates, due dates and family setup times," *Int. J. Production Economics 46–47/119–125*, 1996.

[3] J. Carlier, "Scheduling jobs with release dates and tails on identical machines to minimize the makespan," *European J. or operations research* , pp. 298–306, 1987.

[4] C. A. Glass, C. N. Potts, and P. Shade, "Unrelated parallel machine scheduling using local search," *Mathematical and Computer Modeling*, pp. 41–52, 1994.

[5] N. Piersma and W. Van Dijk, "A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search," *Mathematical and Computer Modeling*, pp. 11 –19, 1996.

[6] V. Suresh and D. Chaudhuri, "Bicriteria scheduling problem for unrelated parallel machines," *Computers and Industrial Engineering* , vol. 30, no. 1, pp. 77 – 82, 1996.

[7] F. Subur, "A methodology for real-time scheduling of jobs with splitting on unrelated parallel machines," *Master Thesis*, 2000.

[8] ?nci Sar?çiçek and C. Çelik, "Two meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness," *Applied Mathematical Modelling*, vol. 35, no. 8, pp. 4117 – 4126, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0307904X1100103X

[9] W. Xing and J. Zhang, "Parallel machine scheduling with splitting jobs," *Discrete Applied Mathematics*, vol. 103, no. 1-3, pp. 259–269, 2000.

[10] E. L. Lawler, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys, "Chapter 9 sequencing and scheduling: Algorithms and complexity," in *Logistics of Production and Inventory*, ser. Handbooks in Operations Research and Management Science. Elsevier, 1993, vol. 4, pp. 445 – 522. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0927050705801896

[11] K. Lam and W. Xing, "New trends in parallel machine scheduling," *International Journal of Operations and Production Manageme nt*, vol. 17, no. 3, pp. 326–338, 1997.