Presented at the DLSU Research Congress 2019
De La Salle University, Manila, Philippines
June 19 to 21, 2019

A Practical Branch and Bound Method for Heuristic Open Shop Scheduling to Minimize Total Weighted Tardiness

Eric A. Siy

Department of Industrial Engineering
De La Salle University
email : eric.siy@dlsu.edu.ph

Abstract: The open shop scheduling problem is sequencing *n* jobs with deterministic processing times at *m* machines when order of machine processing is immaterial. A branch and bound method of open shop scheduling for minimizing total weighted tardiness is suggested based on the partial backward job list on the longest total processing (i.e. "bottleneck") machine. Depth-first schedule generation is suggested to arrive at full schedule fit on weighted tardiness (WT), and continuing search on branches whose fathomed lower bound WT are lower than all other branches searched so far. Generating sequences on non-bottleneck machines were suggested, as well as neighborhood swap steps to find incremental improvements on WT schedules. A simple illustrative example was presented, and a longer worked out example (in the full paper) demonstrates the different cases of local search decisions that can be encountered in the practical scheduling procedure.

Key Words: Open shop scheduling; total weighted tardiness; heuristic

1. INTRODUCTION

The open shop scheduling problem poses the sequencing decision of *n* jobs J_1, J_2….J_n on *m* machines M_1, M_2….M_m. A denoted deterministic processing time P_{ij} on machine *i* for each job *j*. The open shop is distinguished from flow shop and job shop setups whereby the latter two setups have prescribed sequences of machine processing particular to the jobs, while the open shop of this paper's interest has no prescribed sequencing constraints; that is, each job passes through all of the machines that it requires to be processed on in any order. Each machine can attend to only one job at a time, and Non-preemptive constraints require that once a job has started on a machine, it cannot be interrupted until completion for that job. It goes without saying that each job can only be processed at one machine at a time. Each job is initially available to be processed at time zero, and has an associated due date d_j. A job can be sequenced through the m machines so that it completes all required operations at time C_j. When jobs are completed beyond its due date d_j, the job is considered tardy T_j (T_j=max(0, C_j-d_j)). A linear penalty factor W_j is assigned for every time unit that a job is tardy, and the total weighted tardiness for all jobs of a schedule $\Sigma W_j T_j$ denotes the decision fitness of a schedule. The objective for the proposed scheduling method in this paper is to minimize total weighted tardiness.

Open shop scheduling has many practical applications, including inspection, testing, and maintenance (Liaw, 2003), where the order of the operations does not matter. For example, a mobile cellphone factory's quality control center may inspect

a sampled item from the end of production for size dimensions and performance specifications, and such testing can be done in any order; however, the physical specimen need to be at only one of the inspection stations, and inspections cannot proceed simultaneously on the same cellphone item. Other examples of open shop set ups are teacher-class assignments and automobile repair.

Brasel, H. et al (2005) proposed heuristic algorithms for minimizing mean flow time (or completion time) where all jobs have the same priority (or weights). Their survey of list scheduling and matching algorithms concluded that appropriate scheduling algorithms depends on the ratio of n/m where n is the number of jobs and m is the number of machines or processing centers.

Brucker P *et al* (1997) studied the weighted completion time minimization criteria in open shop scheduling using disjunctive graphs and a branch and bound method for scheduling. In their approach, each graph's path from beginning to end node denotes a sequencing solution, and a lower bound for a projected makespan (or maximum completion time of all jobs) is used to evaluate each partial solution's attractiveness.

In this paper, the proposed branch and bound method tackles a different scheduling criterion (i.e. total weighted tardiness) and does not use disjunctive graphs to represent scheduling but rather a list schedule generation process. Furthermore, the proposed "branch" evaluations in this paper is a lower bound for the total weighted tardiness based on the last job sequenced in the bottleneck machine

2. PROPOSED BRANCH AND BOUND HEURISTIC

The search for possible schedules that minimizes total weighted tardiness can be approached using the branch-and-bound (B&B) procedure (Hillier and Lieberman, 2010). A backward scheduling process is prescribed in this process since the final completion times can be ascertained from the total processing times of each machine and of each job. The search begins with a search for the last job on the machine with the highest total processing times, heretofore denoted as the *bottleneck* machine. The last job on the bottleneck machine will end at the highest completion time, so a lower bound for total weighted tardiness (WT) can be initialized based on the last completed job's finish time.

Jobs are then placed in front of the lowest WT rated job, and possible contributions to the current lower bound for WT may be determined. The partial jobs are connected via backward sequence placement until no contributions to WT may be further determined.

Lower bounds on WT may then be scanned across all branches made so far in the B&B search tree, and will then to expand *depth-first* on branches that exhibit a lower value of projected total weighted tardiness. We can "fathom" each branch's lower bound on total WT by generating a full schedule following the prescribed sequence on the bottleneck machine. Here, the non-interference constraint (i.e. no jobs can be simultaneously on two or more machines at any time) for open shop scheduling will be invoked to generate full schedules.

A final incremental improvement step is then prescribed to find better schedule's WT through the recommended processes by this author (Siy, 2011).

As the bounds on WT found under each branch of the expanding search tree can be monitored for the need to expand further on schedule branches with lower bounds that are lower than the currently best-found schedule/s. We terminate the B-&-B procedure when we could not find any branch with a lower bound on total weighted tardiness than the current best schedule made. The search would thus be assured to have found an optimal schedule, without exhaustively searching for all possible schedules via complete enumeration.

The steps can be demonstrated through a worked out illustrative scheduling problem in the open shop for the remainder of this section 2.

2.1 Open shops always have a bottleneck, the main branch for the search tree

We demonstrate the proposed method through the following simple open shop problem shown on Table 1.

Table 1. First illustrative example: Four jobs and three open shop machines' processing times (in time units)

	Job 1	Job 2	Job 3	Job 4
Machine 1	5	8	6	3
Machine 2	7	2	4	5
Machine 3	1	6	9	2
Due date	20	18	20	15
Weight	1	2	3	1

The bottleneck machine is defined as the machine whose total processing times for all jobs is the highest. In Table 1, Machine 1 has a total processing time of 22, while both Machine 2 and 3 are 18. Machine 1 is denoted as the bottleneck machine due to its highest sum of processing times. The processing time of 22 in fact represents the earliest possible finish time for any job sequenced last on Machine 1. Whatever weighted tardiness schedules generated would therefore have to have a non-zero weighted tardiness since the due dates for the jobs on Table 1 are all before t=22 bottleneck completion time.

The proposed branch-and-bound search procedure commences with creating a tree of possible partial schedules that can be generated on the bottleneck machine beginning with the choice of last job on the bottleneck machine. The last job on the bottleneck machine will complete at the theoretical maximum completion time of any job in any schedule. This is a rational way to begin a search for an initial lower bound for the total weighted tardiness.

Suppose initially that Job 1 on Machine 1 is the last job to finish; this scenario may be illustrated by Figure 1. This would result in a Job 1 completion time at time t=22, resulting in a tardiness of 2 time units. The weighted tardiness of this partial schedule should therefore be at least WT=1*(22-20) = 2.

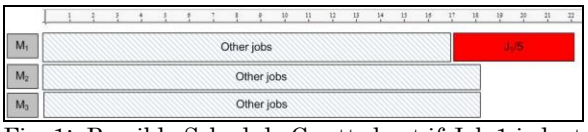

Fig. 1: Possible Schedule Gantt chart if Job 1 is last job to complete

The main branch for our scheduling search tree will have the alternative jobs scheduled at the end of machine 1. All four jobs are possible candidates. When either of the four jobs complete at the end of bottleneck machine 1, they would have contributed varying weighted tardiness.

Let τ denote the current completion time being tested for each branch, which currently stands at τ=22. We may therefore determine the lower bounds for the other two jobs if they were iteratively placed last on machine 1. We can determine that if Job 1 (J1) was scheduled last, then a total weighted tardiness of at least 2 would result. If J2, then the schedule branch would be at least 8 weighted tardiness; J3 would result in 6; J4 with 7. Figure 2 is the initial search tree for our branch and bound search process. As in the known process of the branch and bound search procedure in Integer Programming Operations Research (Hillier and Lieberman, 2010) the main tree suggests that the branch with Job 1 (J1) be the next search focus.

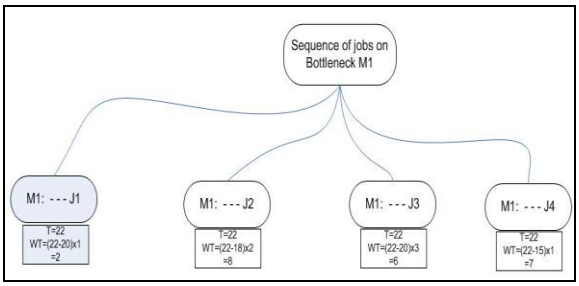

Fig. 2: Initial root branches for the schedule search tree

2.2 Depth first search on the most promising (lowest Weighted Tardiness) branch.

The most promising branch in Figure 2 search tree is having Job 1 scheduled last on Machine M1. The next step is to generate the alternative jobs that can immediately precede last job J1. For this simple three job example, there can only three jobs (either J2, J3 or J4.) When J2 is chosen as the second-to-last job, then two remaining sequences are possible for the last two jobs not yet scheduled. Since J2 can be projected to end at the start of the last job J1 at time t=17, we see that due date is 18 will be met, with no increase in the total weighted tardiness so far of 2. We can evaluate these branches of this search tree through the projected total weighted tardiness that might result for the last two jobs on Machine 1. We generate schedules with the last two jobs on Machine 1 by following the non-interference constraints on the jobs sequenced on the other machine. We generated two such complete schedules as shown on Figure 3, which is the current search tree where the depth-first search for sequences on Machine 1. The lower bound for the total weighted tardiness based on the schedules generated are also shown under each branch's node entry.

The schedule generated in Figure 4 represents the leftmost deep branch on the branch and bound tree in Figure 3. Determining the total weighted tardiness of this schedule can be demonstrated in Table 2.

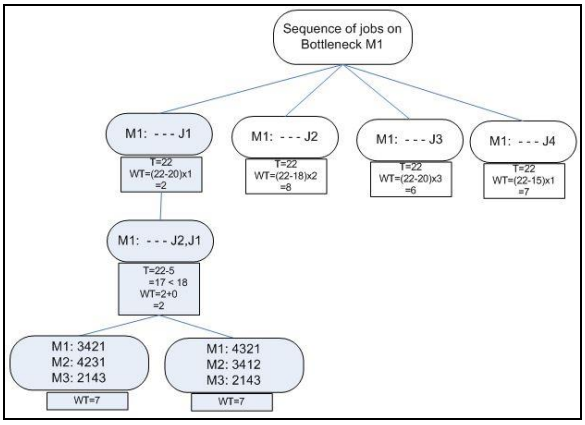

Fig. 3: Partial Branch and Bound Search Tree

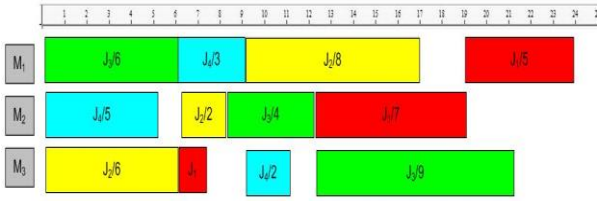

Fig. 4: List Schedule {M1: 3421; M2:4231; M3: 2143} Schedule Gantt chart

Table 2: Demonstration of Total Weighted Tardiness computation for Schedule in Figure 4

Job	J1	J2	J3	J4
Completion time C_j	24	17	21	11
Less: Due date d_j	20	18	20	15
Equals: Tardiness T_j	4	0	1	0
multiplied by Weight Wj	x1		x3	
Equals: Wtd Tardiness	4		3	
ΣTotal Wtd Tardiness=				7

2.3 Incremental improvement on initial generated schedules

When the bottleneck schedule is generated through the branching out process at the main node, the sequence for the other machines has yet to be specified. It is suggested that jobs be sequenced via descending weights and then by due dates. This way, the higher weighted jobs with earlier due dates will finish early. Penalties can be minimized if late jobs are presumably limited to the lower weighed ones.

As a completed sequence is made, the total weighted tardiness of this schedule may be derived. When a completed list schedule has jobs that are tardy, the following suggested sequence improvements may be made to further decrease the weighted tardiness.

(1) Earlier placement of jobs in other machines. Late jobs on non-bottleneck machines can be moved earlier, and thus may even delay the starting times of the other jobs downstream on the bottleneck machine.

(2) Pairwise swapping of jobs within the same machine: where an early job can swap with a late job in the same machine but can improve exchanged tardiness.

(3) Three-way exchanges: partial list "123" can become "312" or "231", as long as job with highest contributed weighted tardiness will finish earlier than in the previous test schedule.

The last schedule in Figure 4 can be improved upon. Pairwise swapping on Machine M2 can be made, resulting in A better schedule with a total weighted tardiness of 4 can be achieved, as shown in Figure 5 and Table 3.

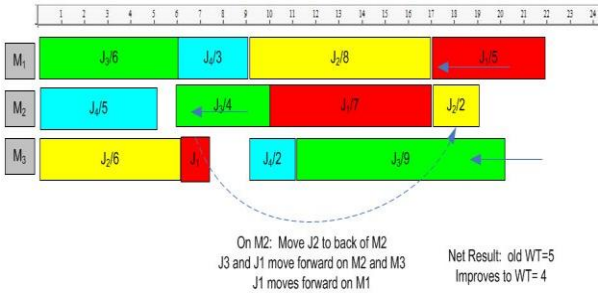

Fig. 5 Illustration of Neighborhood swap List Schedule improvement (Optimal solution)

Our search tree thus can be updated to reflect the improvement on the leftmost branch. The same improvement on the schedule was also done on

the other schedule, with the summarized tree shown on Figure 6.

Table 3: Demonstration of Total Weighted Tardiness computation for Schedule in Figure 5

Job	J1	J2	J3	J4
Completion time C_j	22	19	20	11
Less: Due date d_j	20	18	20	15
Equals: Tardiness T_j	2	1	0	0
multiplied by Weight Wj	x1	x2		
Equals: Wtd Tardiness	2	2		
ΣTotal Wtd Tardiness=				4

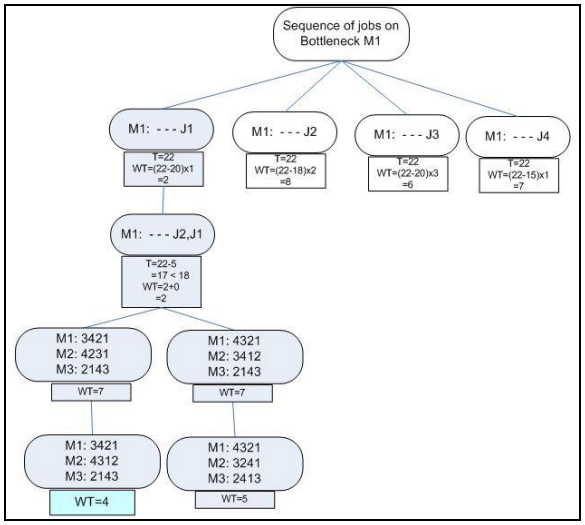

Fig. 6 Depth-first completion of leftmost branch on the Partial search tree

2.4 Continuing the search through the Branch and Bound search tree

The branch and bound procedure can then expand on the other branches that have lower bounds that are less than the current best value of WT=4. The search tree thus expands under the same M1 sequences with J1 as its last schedule. The steps to generate the further schedules was discussed in sections 2.2 and 2.3.

The completed search tree with lower bounds of total weighted tardiness on the branches is shown in Figure 7 (on the next page). Nine complete schedules were generated to present a representative sample of open shop sequences from the illustrative problem. The optimal schedule had the lowest weighted tardiness of 4. The B&B procedure also found lower bounds for the other possible schedules with bottleneck sequences generated from the initial root of the search tree, and showed that other possible schedules would not improve on the WT=4 found for the schedule illustrated in Figure 5.

3. DISCUSSION

The branch and bound (B&B) method of searching through permutation schedules is a kind of tabu search method in the sense that a scheduler searches through only promising sequences of jobs. The partial job sequences (ex. "Sequences ending with Job k on the bottleneck machine") that incur a higher penalty in the weighted criterion are correctly not allocated computational effort due to the lack of promise of finding the optimal sequence there. Although the branch-and-bound process can be tedious using hand calculations, it deservedly ensures that all possible permutation sequences are evaluated to determine optimal solutions.

By dividing the search space between "non-promising", and "optimal schedule may possibly be located here" using the branch-and-bound process, we allocate scarce computational resources to avenues that could yield better results with less time, as compared to the relatively NP-hard ardor of searching through complete enumeration of (n+m)! possible permutation schedules for *n* jobs on *m* machines.

The foregoing prescribed B&B procedure may be extended for larger instances of n jobs and m machines. The hand computations may take considerable time, but the small illustrative problem demonstrated in this paper shows the proof of concept that the procedure is workable even for a larger scale problem. A computer program may be coded to fast-track the computational process

4. CONCLUSIONS

Depth-first branch-and-bound search procedure based on backward scheduled jobs on the bottleneck machine shows promise to finding promising (may be optimal) open shop schedules. The presented heuristic procedure can be used for classroom demonstration of rational decision-making in the open shop scheduling set-up.

Approximation of optimality for larger problems (n>10 jobs and m>5 machines) is yet to be demonstrated due to constraints on computational

and research time resources, but the practical hand computation process for small problems is demonstrated to be practical. For university paper-and-pencil examinations and learning assessments, this scheduling method/tool is appropriate.

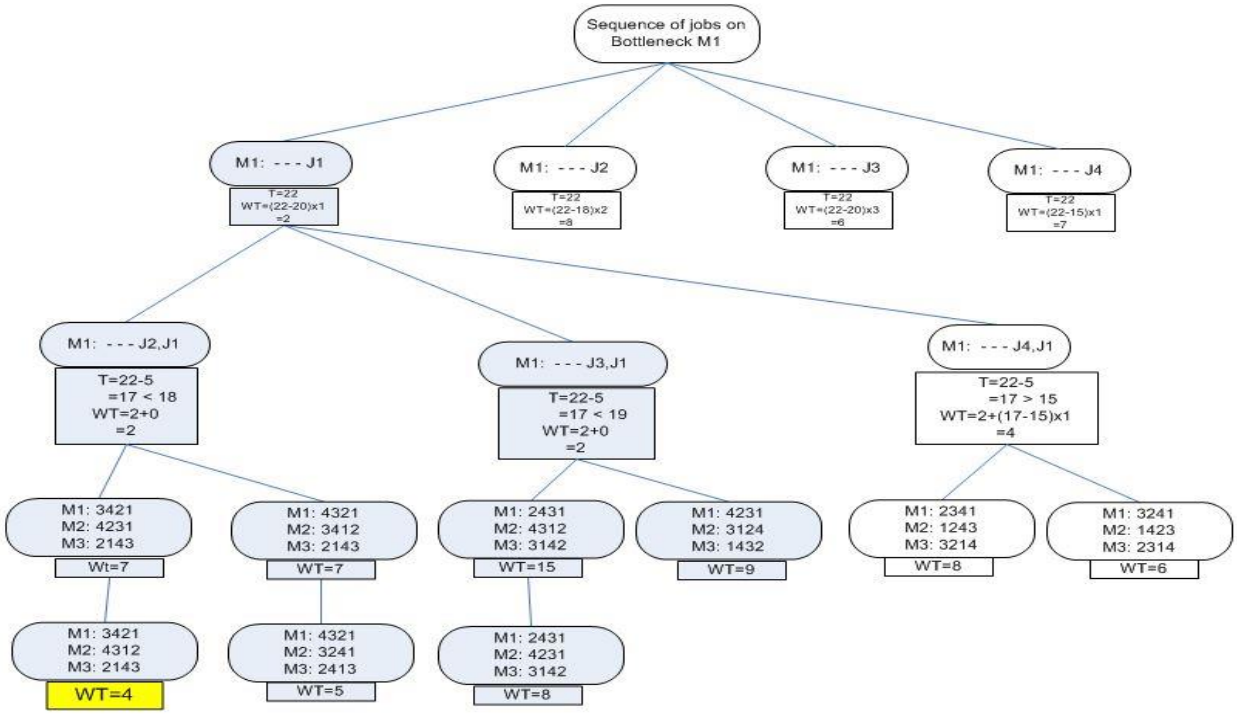

Fig. 7 Complete Branch and Bound search tree with optimal schedule found on leftmost lower node

5. REFERENCES

Brasel, Heidemarie & Herms, Andre & Morig, Marc & Tautenhahn, Thomas & Tusch, Jan & Werner, Frank. (2005). Heuristic Algorithms for Open Shop Scheduling to Minimize Mean Flow Time, Part I: Constructive Algorithms. Preprint FMA, OvGU Magdeburg, 2005.

Brucker, P.; Hurink, J.; Jurisch, B.; Wo¨stmann, B.: A Branch-and-Bound Algorithm for the Open-Shop Problem, *Discrete Applied Mathematics*, Vol 76, 1997, 43-59.

Hillier, F.S. and Lieberman G.J (2010). Introduction to Operations Research 9th International Edition. McGraw-Hill. pp. 491-500.

Liaw, Ching-Fang (2003). "Scheduling pre-emptive open shops to minimize total tardiness", *European Journal of Operational Research* 162 (2005) 173-183. Elsevier. Available through sciencedirect.com retrieved 16 October 2013.

Siy, Eric (2012). "A Neighborhood Search Heuristic for Parallel Identical Machines Minimizing the number of Late Jobs and Maximum Lateness". *Proceedings of DLSU Science and Technology Congress 2012.* Paper 032.