Some Algorithms Used in Parallel Machine Scheduling

Adel Hashem Nour (adilh.alhajjar@uokufa.edu.iq)
Department of Computer Science, College of Education, University of Kufa, Iraq.

Hussam Abid Ali Mohammed
Department of Mathematics, College of Education for Pure Sciences, University of Karbala, Karbala, Iraq.
Kareema Abed Al-Kadim (kareema.kadim@yahoo.com)
Department of Mathematics, College of Education for Pure Sciences, University of Babylon, Hilla, Iraq

Abstract: The paper discusses the challenge of organizing the execution of separate tasks on machines that are otherwise similar. The goal is a lower number of Makes. We create, examine, and evaluate several local search techniques, including the (BAB) , (GA) and (TS) algorithms. Based on past computational experience, we know that these local search algorithms can efficiently complete up to 2,000 tasks in Matlab .

Keywords- Parallel machines, Genetic algorithm. , Tabu Search , Branch and Bound algorithm.

1. Introduction

The scheduling of several machines in parallel is a serious challenge for modern industry. The goal of this challenge is to minimize some performance metric, such as the maximum time it takes to finish all of the tasks, the number of jobs that finish late, the penalty for finishing late, and so on. It is common practice in many industries to run multiple, identical machines in parallel. This is seen in the baking industry, where multiple ovens are used to ensure that enough baked goods are produced, in the printing industry, where a publisher will need multiple identical printing facilities to ensure that no special titles (or, worst case, bestsellers) are unavailable to retailers, in the pharmaceutical industry, where multiple machines are used to ensure adequate drug production, and in many other settings as well. Objectives including earliness and tardiness penalties are growing in popularity due to the growing popularity of just-in-time production systems, in which the early and late completion of work processing are both undesirable.[1]

Many search-based and enumerative methods may be used to scheduling issues

because of their combinatorial character. These methods include genetic algorithms, branch and bound, simulated annealing, tabu search, etc. These procedures often provide high-quality results. Therefore, scheduling using quick heuristic algorithms is not only very effective, but often the only viable approach.[2]

If you have a combinatorial optimization issue, a Genetic Algorithm is a great tool for undertaking a global search. Using a Genetic algorithm and a backward-forward heuristic technique (Sule, 2007), this study proposes a hybrid solution for the early-late, non-common due-date scheduling issue, in which a group of tasks must be scheduled to run on a group of identical parallel computers. The cost of keeping goods on hand for each batches is included into the early arrival penalty.[3]

When solving optimization issues, the Branch and Bound algorithm is an effective method for quickly arriving at the best possible answer. Algorithmically, the issue is broken down into more manageable chunks, each of which is solved independently before the final, optimum solution is combined.[4]

The objective of making use of identical parallel machines is to reduce the makespan, or the time it takes to finish all of the operations, by scheduling them in such a manner that they run in parallel as efficiently as possible. The Branch and Bound method may be used to find a better schedule by searching through all of the potential schedules and eliminating the branches of the search tree that don't go somewhere better.

The method achieves its goals by repeatedly breaking down the search space into smaller issues and then finding lower and upper limits on the optimum makespan for each of those problems. For the lower bound, we solve a simplified version of the issue, while for the upper limit, we may either use heuristics or solve the problem precisely.

At each stage, the algorithm chooses the subproblem with the lowest lower limit from a priority queue and then extends it by branching on a variable. By branching, the method generates two new subproblems, each with a restriction on the branched-on variable. The method keeps going until every possible subproblem has been investigated, and the best solution is the one with the shortest makespan.[5]

In conclusion, the Branch and Bound algorithm is an effective method for resolving optimization issues, such as the scheduling of tasks on similar parallel computers to

reduce the makespan. Using a recursive partitioning into smaller subproblems, computing lower and upper bounds on the optimal makespan for each subproblem, and pruning branches of the search tree that cannot lead to a better solution than the best solution found so far, the algorithm explores the search space of all possible schedules.

In order to handle difficult combinatorial optimization problems, such scheduling issues for a set of identical parallel machines, the Tabu method is used as a metaheuristic optimization tool. The Tabu technique avoids getting stuck at local optimums and effectively searches the whole search space by combining local search with memory-based search.[6]

To plan a series of tasks on a set of identical parallel machines in a manner that minimizes the makespan (the time needed to finish all the jobs) is the objective in the context of identical parallel machines. By producing a series of candidate solutions and employing a set of memory structures called Tabu lists to avoid revisiting solutions that have previously been studied, the Tabu algorithm may be used to solve this issue.[6]

The technique generates a list of potential solutions by iteratively making modest changes to the original answer. Each potential answer is ranked by the algorithm, and the one with the shortest makespan is chosen. The algorithm also remembers a number of "Tabu lists," which it uses to avoid redoing work it has already done. The algorithm cannot revert the changes that have been made to the present solution because of the Tabu lists, which store this information.

A stopping condition, such as a maximum number of iterations or a minimum improvement threshold, is used to determine when the Tabu algorithm stops generating and evaluating potential solutions. There are ways to improve the algorithm, such as diversification mechanisms that push it to look in uncharted areas of the search space and intensification processes that narrow in on the most promising ones.[7]

In conclusion, the Tabu technique is a metaheuristic optimization approach useful for solving complicated combinatorial optimization problems, such as scheduling issues for a set of identical parallel machines. In order to prevent re-exploring solutions that have previously been studied, the method builds a series of potential solutions and

employs Tabu lists.If conventional algorithms fail to converge to a reasonable solution, the Tabu technique is a potent tool for discovering excellent solutions to optimization problems.

2. Problem Formulation

Suppose we have two identical parallel machines, the goal is to minimize of the completion time cost as well as the earliness time cost with the lateness time cost . We will have the following **objective function:**

Min $(\sum_i^p \boldsymbol{C_j}\ ,\ \sum_i^p(\boldsymbol{\alpha E_j + \beta T_j})$ ……..(1)

The constraints for the objective function of two identical parallel machines are:

1.Capacity constraint: The total processing time of all jobs assigned to each machine cannot exceed its capacity. Mathematically, it can be expressed as:

∑_(pij ≤ Cj), where p_ij is the processing time of job i on machine j and Cj is the capacity of machine j.

2.Job assignment constraint: Each job must be assigned to exactly one machine. Mathematically, it can be expressed as:

∑_(xij = 1), where x_ij is a binary decision variable that equals 1 if job i is assigned to machine j and 0 otherwise.

3.Non-preemptive constraint: Once a job is assigned to a machine, it cannot be interrupted or moved to another machine. Mathematically, it can be expressed as:

∑_xij pij = ∑_(xkj pkj), for all j∈{1,2}, where x_ij and xkj are binary decision variables that equal 1 if job i and k are assigned to machine j, respectively, and 0 otherwise.

4. Objective function constraint: The objective function is the sum of the completion times on each machine and the weighted sum of earliness and tardiness penalties. Mathematically, it can be expressed as:

Minimize: ∑_(Cj)+∑_((αEj+βTj)).....(2)

where Ej is the earliness of job i on machine j (i.e., the difference between its completion time and its due date if it is completed before the due date, and 0 otherwise), Tj is the tardiness of job i on machine j (i.e., the difference between its completion time and its due date if it is completed after the due date, and 0 otherwise), α and β are weighting factors for earliness and tardiness, respectively.

These constraints ensure that the jobs are assigned to the machines in a way that minimizes the objective function while respecting the capacity and assignment constraints.

3.1. BAB Algorithm

The Branch and Bound (BAB) algorithm is a widely used optimization technique in computer science and operations research. It is particularly useful for solving combinatorial optimization problems such as the matching of parallel machines.[8]

In the context of matching parallel machines, the goal is to assign a set of tasks to a set of machines in such a way that the total time, early time, and delay time are minimized. The BAB algorithm is well-suited for this problem because it systematically explores the solution space by dividing it into smaller and smaller sub-problems until the optimal solution is found.[8]

At a high level, the BAB algorithm works by creating a search tree that represents all possible assignments of tasks to machines. The root node of the tree represents the initial state of the problem, and each child node represents a possible assignment of a single task to a machine. The algorithm then evaluates each child node to determine its quality and selects the most promising node to explore further. This process continues until the optimal solution is found or all nodes have been evaluated.[9]

The BAB algorithm is particularly effective for solving matching parallel machines problems because it can quickly eliminate large portions of the solution space that are unlikely to contain optimal solutions. By using various heuristics to guide the search process, the BAB algorithm can efficiently find near-optimal solutions even for large problem sizes.

Overall, the BAB algorithm is a powerful tool for optimizing the assignment of tasks to parallel machines. By reducing the total time, early time, and delay time, it can help organizations save time and resources while improving their productivity and efficiency.

3.2.Genetic Algorithm (GA)

The Genetic Algorithm (GA) is a popular optimization technique inspired by natural selection and genetics. GA is a population-based algorithm that mimics the process of natural selection, where individuals with better fitness values are more likely to survive and pass their genetic traits to the next generation. GA has been widely used in various optimization problems, including the matching of parallel machines. Matching parallel machines involve assigning a set of tasks to a set of machines to minimize the total time, early time, and delay time. This is a complex combinatorial optimization problem that can be challenging to solve using traditional optimization methods. However, GA can be an effective tool for solving this problem by generating a set of potential solutions and iteratively improving them over multiple generations.[10]

In GA, a population of potential solutions, known as chromosomes, is randomly generated. Each chromosome represents a possible solution, which consists of a set of task assignments to the machines. The fitness of each chromosome is then evaluated based on the objective function, which is the sum of the total time, early time, and delay time. The chromosomes with higher fitness values are more likely to be selected for the next generation, and they will pass their genetic traits to the offspring through crossover and mutation operators.Through successive generations, the population evolves to generate better solutions that can minimize the objective function. The GA algorithm continues this iterative process until a stopping criterion is met, such as reaching a predetermined number of generations or achieving a satisfactory fitness value.[10]

In the context of matching parallel machines, GA can be used to optimize the task assignment and scheduling to minimize the total time, early time, and delay time. By generating a diverse set of potential solutions and iteratively improving them over multiple generations, GA can help reduce the time required to match parallel machines and improve the overall efficiency of the system.[11]

Overall, GA is a powerful optimization tool that can be applied to many complex optimization problems, including the matching of parallel machines. By using GA to optimize the task assignment and scheduling, organizations can improve their productivity and reduce the total time, early time, and delay time of their machines

3.3.Tabu search

From 1985 forward, Glover [11] wrote a plethora of publications detailing the different uses of the tabu search. Sequencing, scheduling, oil drilling, and routing are just some of the other applications that have seen rapid uptake since the discovery of this method.

The tabu search has useful qualities that may be used to improve other procedures by keeping them from becoming trapped in the areas of local minima. The tabu search makes use of memory to stop it from fast cycling back to previously searched areas of the solution space. For this purpose, a database is maintained with examples of potential answers. The term "taboo" refers to the social stigma associated with these types of fixes. One of the factors in the tabu search is the length of the tabu list.[6]

The tabu search also includes mechanisms for regulating the search. The tabu list guarantees that at least one solution will be undesirable, but it's possible that the tabu list's restrictions might be too severe in certain situations, leading to the algorithm being stuck on a local maximum. The tabu search provides the idea of aspiration criteria to get around this issue. The ambition criteria supersede the taboo constraints, allowing for a more inclusive search for the global optimum.

4. Lowe Bound of BAB

Min f(σ)= $\underset{\sigma\in S}{\text{Min}}\{\sum_{j=1}^{n}(\underset{\sigma j}{E}+\underset{\sigma j}{T}+\underset{\sigma j}{C})\}$…….(3)

$=\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\{\text{Max}\{\underset{\sigma j}{d}-\underset{\sigma j}{C},0\}+Max\{\underset{\sigma j}{C}-\underset{\sigma j}{d},0\}+\underset{\sigma j}{C}\}$

$=\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\{\text{Max}\{\underset{\sigma j}{d}-\underset{\sigma j}{C},\underset{\sigma j}{C}-\underset{\sigma j}{d},0\}+\underset{\sigma j}{C}\}$

$=\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\{\text{Max}\{\underset{\sigma j}{d},2\underset{\sigma j}{C}-\underset{\sigma j}{d},\underset{\sigma j}{C}\}$……..(4)

Since the third term $\underset{\sigma j}{C}$ is between $\underset{\sigma j}{d}$ and $2\underset{\sigma j}{C}-\underset{\sigma j}{d}$ always, then we can write the

Objective function f(σ) by the form:

Min f(σ)= $\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\{\text{Max}\{\underset{\sigma j}{d},2\underset{\sigma j}{C}-\underset{\sigma j}{d}\}$……..(5)

It is clear from equation (3) a lower bound (LB) is obtained by sequencing

The job by SPT rule.

Hence, we can prove that:

Min f(σ) $\geq \underset{\sigma\in S}{\text{Min}}\{Max\{\sum_{j=1}^{n}\underset{\sigma j}{d},\sum_{j=1}^{n}Max\{2\underset{\sigma j}{C}-\underset{\sigma j}{d},\underset{\sigma j}{C}\}\}\}$

It is a LB of our problem, since

$\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\{\text{Max}\{\underset{\sigma j}{d},2\underset{\sigma j}{C}-\underset{\sigma j}{d},\underset{\sigma j}{C}\}\geq\underset{\sigma\in S}{\text{Min}}\{Max\{\sum_{j=1}^{n}\underset{\sigma j}{d},\sum_{j=1}^{n}Max\{2\underset{\sigma j}{C}-\underset{\sigma j}{d},\underset{\sigma j}{C}\}\}\}$

Put $\underset{\sigma j}{x}=Max\{2\underset{\sigma j}{C}-\underset{\sigma j}{d},\underset{\sigma j}{C}\}$

To show $\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\text{Max}\{\underset{\sigma j}{d},\underset{\sigma j}{x}\}\geq\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\text{Max}\{\sum_{j=1}^{n}\underset{\sigma j}{d},\sum_{j=1}^{n}\underset{\sigma j}{x}\}$

Since $\underset{\sigma j}{d}$ and $\underset{\sigma j}{x}$ are positive integers .

Hence , it is clear that

$\underset{\sigma\in S}{\text{Min}}\sum_{j=1}^{n}\{\text{Max}\{\underset{\sigma j}{d},2\underset{\sigma j}{C}-\underset{\sigma j}{d},\underset{\sigma j}{C}\}\geq\underset{\sigma\in S}{\text{Min}}\{Max\{\sum_{j=1}^{n}\underset{\sigma j}{d},\sum_{j=1}^{n}Max\{2\underset{\sigma j}{C}-\underset{\sigma j}{d},\underset{\sigma j}{C}\}\}\}$

Hence , LB = $\underset{\sigma \in S}{\mathbf{Min}}\{ \boldsymbol{Max}\{\sum_{j=1}^{n} \mathbf{d}_{\sigma j} , \sum_{j=1}^{n} \boldsymbol{Max} \{ \mathbf{2C}_{\sigma j} - \mathbf{d}_{\sigma j}, \mathbf{C}_{\sigma j}\}\}\}$ ... ... (6)

5. Comparison Results

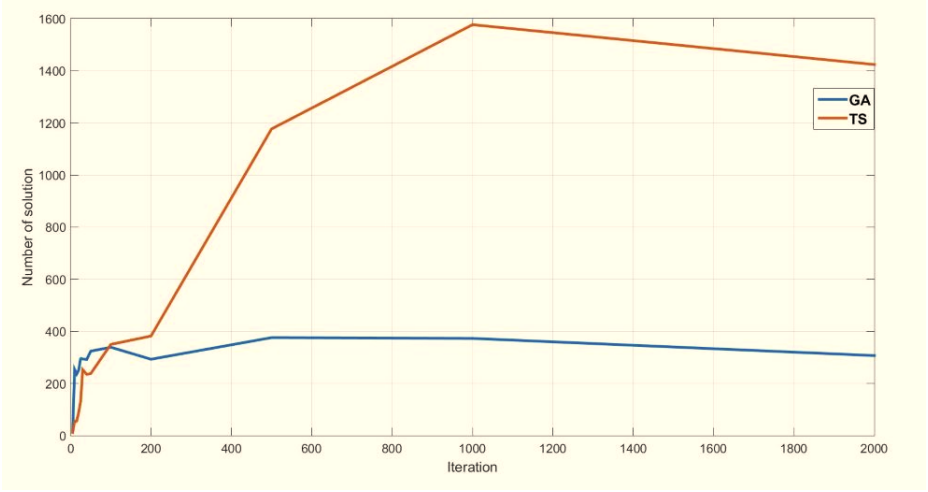
n	Table-1- Comparison between Genetic Algorithm and Tabu Search			
	GA		TS	
	No. of Sol.	Time(s)	No. of Sol.	Time(s)
5	17	0.5155333	12	0.0562403
10	253	0.6339888	56	0.0988966
15	236	0.7376676	54	0.1483479
20	251	1.0125075	89	0.2070926
25	297	1.0125075	132	0.2699514
30	294	1.1297793	253	0.6339888
40	292	1.3990928	235	0.4980252
50	324	1.6517448	238	0.6873296
100	339	2.9507885	350	2.3416946
200	293	5.7969116	382	8.2839836

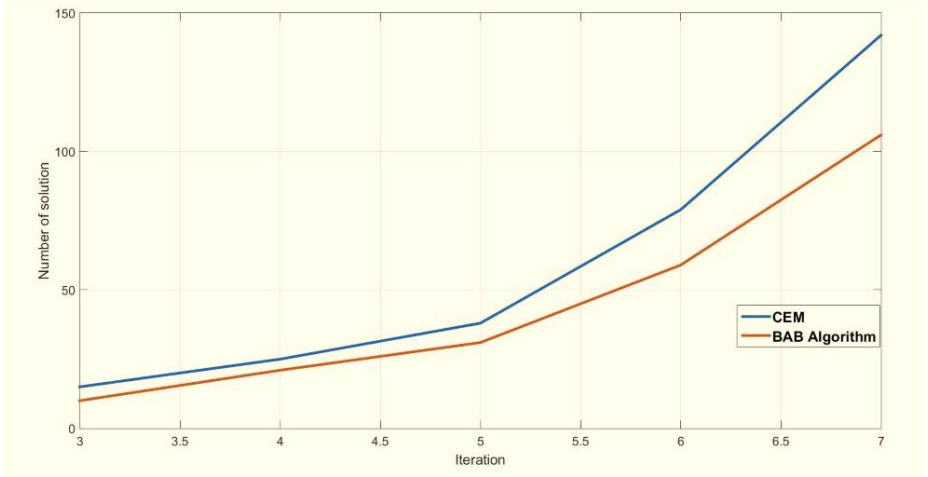
500	**376**	**15.874865**	**1177**	**51.3586491**
1000	**373**	**36.229470**	**1577**	**243.1260402**
2000	**307**	**97.018255**	**1424**	**600.1819503**

Table -2-

Comparison between Complete Solution and BAB Algorithm

n	**CEM**		**BAB**		
	No. of Sol	**Time(s)**	**No. of Sol**	**Time(s)**	**NODS**
3	**15**	**0.3143539**	**10**	**0.0356346**	**29**
4	**25**	**0.0021941**	**21**	**0.0091321**	**132.91**
5	**38**	**0.0048883**	**31**	**0.0103229**	**1189.4**
6	**79**	**0.0923632**	**59**	**0.0460734**	**6671.6**
7	**142**	**9.98810817**	**106**	**2.85608785**	**76648**





Conclusion

The main problem in this study is to find a set of "good solutions" to a problem Reducing the total total time, early time and late time we suggest to develop three Algorithms are BAB, GA, and Tabu Search.
By comparing the proposed algorithms relative to the results, we found that the developed BAB algorithm gives efficient results and better performance than the original algorith

1 References

[1] Wayne L. Winston ," Applications and Algorithms" Cengage Learning,2018.

[2] Ho, W., Ji, P., & Chung, S. H. Scheduling identical parallel machines: a survey. Journal of the Operational Research Society, 63(12), 1671-1686, 2012.

[3] Salavati, M. R., Razzaghi, M., & Hosseini-Motlagh, S. M. Parallel machine scheduling with early and tardy costs. International Journal of Production Research, 49(4), 991-1004, 2011.

[4] Vargas, V. C. Scheduling with early and tardy penalties: a review. Computers & Industrial Engineering, 66(3), 446-454, 2013.

[5]Pinedo, M.L., *Scheduling Theory, Algorithms, and Systems*, Springer Science + Business Media, Fourth Edition, LLC., New York, 2012.

[6] Glover F,*Tabu Search Part I*, ORSA Journal on Computing, 1: 190–206, ., 1989.

[7] Liang, Y–C., Chen, H–L.A., and Tien, C–Y., *VariableNeighborhood for Multi–Objective Parallel Machine Scheduling Problems*, In: Proc. of the 8th International Conference on Information andManagement Science (IMS 2009), Chine, 519–522, 2009.

[8] Puchinger, J., Raidl, G.R., *Combining Metaheuristics and Exact Algorithms on Combinatorial*: *A Survey and Classification*. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2005, LNCS, 3562:41–53, Springer,Heidelberg, 2005.

[9] Tapan, S., Farhad, M.E., and Parthasarati, D,*A Branch and Bound Approach to the Bicriteria Scheduling Problem Involving Total Flowtime and Range of Lateness*, Management Science, 34(2):254–260, 1988.

[10] Holland, J.H., *Adaptation in Natural and Artificial Systems*, Cambridge, MA: MIT Press. Second edition (1992), (First edition, University of Michigan Press, 1975), 1975/1992.

[11]Glover F., Tabu Search Part I, ORSA Journal on Computing, 1: 190–206, 1989,

Article submitted 1 April 2023. Accepted at 24 May

Published at 30 Jun 2023.