RESEARCH PAPER Mathematics Volume : 3 | Issue : 12 | Dec 2013 | ISSN - 2249-555X

Tabu Search Algorithm For Solving of Job Shop Scheduling with Single Objective - Makespan Minimization

KEYWORDS Tabu Search, makespan, Job Shop Scheduling,NP hard

K.C.UDAIYAKUMAR
Research scholar, Sathyabama university,Chennai

M.CHANDRASEKARAN
Director,Vels University, Chennai

ABSTRACT *In this paper we defiance the job shop scheduling problem with total makespan minimization. We intended to extend the disjunctive graph model used for makespan minimization to represent the translation of the problem with total makespan minimization. Using this representation, adapting local search neighbourhood structures actually defined for makespan minimization . This proposed neighbourhood structures are used in a evolutionary algorithm technique hybridised with a simple Tabu search method, one of the best and outperforming state-of-the-art methods in solving problem NP hard problem like Jssp and various other engineering problems.*

1. INTRODUCTION

In this paper we confront the Job Shop Scheduling Problem (JSP) with makespan minimization. Many researchers started showing more interest in solving JSSP in the last decades, but in most of the cases they solved single objective function mainly makespan. The Job Shop Scheduling problem is among the NP-Hard [6] problems with the most practical applications such as manufacturing indusries,salesman problems and other engineering fields. Industrial tasks starts from assembling automobiles to scheduling airplane maintenance crews are easily modeled as instances of this problem, and developing solutions atleast by one percent can have a significant financial impact. In addtion, this problem is interesting from a theoretical standpoint as one of the most difficult NP-Hard problems to solve in practice. To cite the canonical example, one 10 x 10 (that is, 10 jobs with 10 machines each) instance of this problem which is denoted as MT10 in the literature was introduced by Muth and Thompson in 1963, but not provably optimally solved until 1989.

2. LITERATURE SURVEY

First local search algorithms were tailored for the job shop problem in late 1980's, many different approaches have been developed. P.J.M. van Laarhoven et al.[13] introduced the first simulated annealing algorithm for the job shop problem in 1988. At the same year, H. Matsuo et al. [9] introduced a similar algorithm which was considerably more efficient. From that, there has been considerable improvement, and an algorithm of computational analysis of jssp using tabu search by Aarts et al. [1] published in 1994 is now the standard bearer of the area in terms of mean percentage error from the optimal [14]. Genetic Algorithms have also thrived as an area of study. Yamada and Nakano[15] introduced one of the first such algorithms tailored to this problem in 1992. After couple of years , Aarts et al. [1] published one that was most efficient and robust. In 1995, Della Croce, et al.[5] proposed another efficient algorithm amongst a flurry of activity. Tabu Search has also been an active field of study. Taillard [12] introduced the first Tabu search-based algorithm in 1989.

Dell'Amico and Trubian [4] moved forward with several new advances in 1993. Barnes and Chambers [3] developed another Tabu search algorithm in 1995, and Nowicki and Smutnicki [10] published a Disjunctive (machine) arcs Conjunctive (job) arcs fast and robust one in 1996. One other entry of note is the Guided Local Search algorithm of Balas and Vazacopoulos [2] first published in 1994. While slow, it tends to find very good solutions on hard instances of the job shop problem.

Vaessens, et al. [14] in their 1996 survey of local search algorithms describe and explain that the available Tabu search algorithms dominate the genetic algorithms and perform substantially better than the simulated annealing algorithms in most cases. The guided local search algorithms of Balas and Vazacopoulos compares favourably with the robust Tabu search algorithms on the data sets tested. Hence Tabu search seems to be a good basis framework for exploring a wider class of problems.

3. PROBLEM DEFINITION

The Job Shop Scheduling problem is formalized as a set *J* of *n* jobs, and a set *M* of *m* machines. Each job *Ji* has *ni* subtasks (called *operations*), and each operation *Jij* must be scheduled on a predetermined machine, µ*ij* Є *M* for a fixed amount of time, *dij*, without interruption. No machine may process more than one operation at a time, and each operation *Jij* Є *Ji* must complete before the next operation in that job (*Ji*(*j*+1)) begins. The successor of operation *x* on its job is denoted *SJ*[*x*], and the successor of *x* on its machine is denoted *SM*[*x*]. Likewise, the predecessors are denoted *PJ*[*x*] and *PM*[*x*]. Every operation *x* has a *release* (start) time denoted *rx*, and *tail* time denoted *tx* which is the longest path from the time *x* is completed to the end.

To solve this problem, for each machine, find an ordering of the operations to be scheduled on it that optimizes the objective function. There are several objective functions which are frequently applied to this problem. Far and away the most common is the minimization of the *makespan*, or the total time to complete all tasks. This objective function is widely used because it models many industrial problems well, and because it is very easy to compute efficiently. Others of note are the minimization of the total (weighted) tardiness, which is useful when modeling a problem where each job has its own due date, and minimization of total (weighted) cost, which is useful for modeling problems in which there is a cost associated with the operation of a machine.

4. DISJUNCTIVE GRAPH IMPLEMENTATION

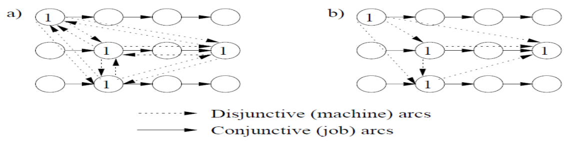

Figure 1: An illustration of a disjunctive graph

The disjunctive graph representation for scheduling problems was first introduced by Roy and Sussmann in 1964 [11]. In this representation, the problem is modeled as a directed graph with the vertices in the graph representing operations, and with edges representing precedence constraints be-

Volume : 3 | Issue : 12 | Dec 2013 | ISSN - 2249-555X

tween operations. More precisely, a directed edge ($v1; v2$) exists if the operation at $v1$ completes before the operation at $v2$ begins. These edges are divided into two sets called conjunctive arcs and disjunctive arcs. The conjunctive arcs are the precedence's deriving from the ordering of the operations on their respective jobs. These edges are inherent in the problem definition and exist irrespective of the machine configurations. The disjunctive arcs, on the other hand, represent the precedence constraints imposed by the machine orderings. Before an ordering is imposed, ¥ $xi; yi$ to be performed on machine Mi, there exist two conjunctive arcs, ($xi; yi$) and ($yi; xi$). Selecting a machine ordering is performed by removing exactly one arc from each pair to form a directed acyclic sub-graph.

Figure 1 is an example of a subset of a disjunctive graph where 4 operations are to be scheduled on machine 1. In diagram a none of machine 1's disjunctive arcs have been selected, and so every operation has a pair of disjunctive arcs linking it with every other operation on the same machine. In diagram **b** is a selection of disjunctive arcs which defines an ordering of the operations on machine 1.

5. TABU SEARCH

The performance of a local search algorithm, both in terms of the quality of solutions, and in the time required to reach them is heavily dependent on the neighborhood structure. Formally, given a solution s a *neighborhood* is a set $N(s)$ of candidate solutions which are adjacent to s. This means that if the search are currently examining solution s the next solution we examine will be some $s \in N(s)$. Typically, the solutions in $N(s)$ are generated from s with small, local modifications to s commonly called *moves.*

A neighbourhood function must strike a balance between efficient exploration and wide coverage of the solution space. Using neighbourhoods which are small and easy to evaluate may not allow the program to find solutions very different from the initial solution, while using those that are very large may take a long time to converge to a reasonably good solution. Some properties that seem to be useful for job shop neighbourhood functions are described below, as are several neighbourhood functions described in the literature.

6. EXPERIMENTAL IMPLEMENTATION

There has been a great deal of research to find good, efficient heuristics to the job shop scheduling problem. Notably among these are the so-called *List Scheduling* (or *Priority Dispatch*) algorithms. These are constructive heuristics which examine a subset of operations and schedule these operations one at a time. While there are no guarantees on their quality, these algorithm have the advantage of running in sub-quadratic time (in normal use), and producing reasonable result with any of a number of good priority rules. List scheduling algorithms were first developed in the mid 1950's, and until about 1988 were the only known techniques for solving arbitrary large (≥ 100 element) instances. While List Scheduling algorithms are no longer considered to be the state of the art for solving large job shop instances, they can still produce good initial solutions for local search algorithms. One of the most popular is the *Lawrence Schedule* which selects the operation with the most work remaining (i.e. with the greatest tail time).

Tabu Search (JSSP)

```
1 JSSP is an instance of the Job Shop Scheduling problem
2 sol :   Initial Solution (JSSP)
3 best Cost :  cost (sol)
4 best Solution :  sol
5 Tabu List   ;
6 while : keep Searching()
7 do Nvalid(sol)   { S Є N(sol) | Move[sol; s] Є # TabuList }
8 if Nvalid(sol) # γ
9  then sol :    x Є Nvalid(sol) ¥y Є Nvalid(sol) cost(x) ≤ cost(y)
10 update Tabu List (sol)
11 if cost (Move[sol; soll]) < bestCost
12 then bestSolution  :  sol
13 bestCost  :  cost(sol)
14 sol : sol
15 return bestSolution
```

Figure 4.1: Pseudo-code for a Tabu search framework

List-Schedule(JSSP)

```
1 .  JSSP is an instance of the Job Shop Scheduling problem
2 .  L is a list, t is an operation, µt is the machine on which t
     must run
3    for each Job Ji Є JSSP
4    do L :  L Ủ first[Ji]
5    for each Machine Mi Є JSSP
6    do avail[Mi]  :  0;
7    while L # γ ;
8    do t : bestOperation(L)
9    µt[avail[µt] : t
10   avail[µt]   avail[µt] + 1
11   L :  L \ t
12   if t # last[Jt]
13   then L :  L Ủ jobNext(t)
```

Figure 4.2: Pseudo-code for a List Scheduling algorithm

7. RESULTS

In this project, Tabu framework were developed and the developed Tabu were tested with the standard benchmark problem Lawrence problem which were taken from the OR library. The obtained result and the benchmark result were compared and is given in the table 5.1

Table 5.1 Comparison of the result obtained from Tabu and the Benchmark Dataset

Problem	Best Value	Obtained value	Time (best)	No. of iterations
LA1	666	666	725 Sec	900
LA2	655	655	700 Sec	900
LA3	597	597	750 Sec	900
LA4	590	590	825 Sec	900
LA5	593	593	854 Sec	900
LA6	926	926	893 Sec	1500
LA7	890	890	920 Sec	1500
LA8	863	863	1060 Sec	1500
LA9	951	951	1125 Sec	1500
LA10	958	958	1150 Sec	1500

Form the table it is quite clear that the value obtained is matching the benchmark problem dataset. So the developed algorithm is suitable for the job shop scheduling problems. However the time taken for achieving the result is slightly higher when compared to the standard value by the other algorithms, as the tabu searching is time.

8. CONCLUSION

In this project, TS has been developed to solve not only single objective but also for multi-objective and multi-constrained JSSP. This research work has taken up the problem of scheduling jobs to various machines of different processing time. Lawrence problem from 01 to 10 were taken and solved. To arrive with a feasible pattern, major constraints faced such as each machine has different processing time, one job can be processed at a time in a machine, holding time also included in the processing time, job should complete all the process, each machine was capable of doing all the operations.

The tabu parameters have been set by conducting various test cases and the operators have also been modified to suit the need. Swapping has been eliminated to reduce the computational complexity when compared to the traditional methodology. Exploitation and exploration operator has been enhanced to improve the convergence rate and thereby reducing the computational time. It was found that the

Volume : 3 | Issue : 12 | Dec 2013 | ISSN - 2249-555X

performance was better compared to the steady state tabu.

Global memory updating operator used to avoid stagnation and to check the probability of various patterns. Global memory updating operator has been developed to change the local stagnation at random, thereby it avoids the stagnation, increases the search space and also identifies the best job. Local memory updating operator has been applied to increase the search space for identifying the better solution. Various fitness functions like makespan minimization function, penalty function, idle time minimization function have also been tried and the finally end up with the completion make span minimization fitness function to calculate the best job shop sequencing.

This project is not to prove that the TS algorithm is better. But, it proved that the TS can be an effective algorithm for JSP with related constraints.

9. FUTURE SCOPE

It is possible to reasonably extend these algorithms to even broader classes of job shop problems (e.g. A wider class of objective functions). Someother neighbourhood functions which are better suited to solving problem instances with sequence dependent setup times. Some other heuristics that are better suited to providing good initial solutions to problem instances with sequence dependent setup times.

REFERENCE

[1] E.H.L. Aarts, P.J.M. van Laarhoven, J.K. Lenstra, and N.L.J. Ulder, \A Computational Study of Local Search Algorithms for Job Shop Scheduling", ORSA Journal on Computing 6, (1994)118-125. | [2] E. Balas and A. Vazacopoulos, \Guided Local Search with Shifting Bottleneck for Job Shop Scheduling", Management Science Research Report, Graduate School of Industrial Administration, Carnegie Mellon University (1994). | [3] J.W. Barnes and J.B. Chambers, \Solving the Job Shop Scheduling Problem Using Tabu Search", IIE Transactions 27, (1994)257-263. | [4] M. Dell'Amico and M. Trubian, \Applying Tabu search to the job-shop scheduling problem", Annals of Operations Research, 41(1993)231-252. | [5] F. Della Croce, R. Tadei, and G. Volta, \A Genetic Algorithm for the Job Shop Problem", Computers and Operations Research, 22(1995)15-24. | [6] M.R. Garey, D.S. Johnson, and R. Sethi, \The complexity of flowshop and jobshop scheduling", Mathematics of Operations Research, 1(1976)117-129. | [7] J.P. Hart and A.W. Shogan, \Semi-greedy heuristics: an empirical study", Operations Research Letters 6(1987)107-114. | [8] A.S. Jain and S. Meeran, \Deterministic job-shop scheduling: Past, present, and future", European Journal of Operational Research, 113(1999)390-434. | [9] H. Matsuo, C.J. Suh, and R.S. Sullivan, \A Controlled Search Simulated Annealing Method for the General Jobshop Scheduling Problem", Working Paper 03-04-88, Graduate School of Business, University of Texas, Austin. | [10] E. Nowicki and C. Smutnicki, \A Fast Taboo Search Algorithm for the Job Shop Problem", Management Science, 6(1996)797-813. | [11] B. Roy and B. Sussmann, \Les problems d'ordonnancement avec constraintes disjonctives", Node DS n.9 bis, SEMA, Montrouge (1964). | [12] E. Taillard, \Parallel Taboo Search Techniques for the Job Shop Scheduling Problem", ORSA Journal on Computing 6, (1994)108-117. | [13] P.J.M. van Laarhoven, E.H.L. Aarts, and J.K. Lenstra, \Job shop scheduling with simulated annealing", Report OS-R8809, Centre for Mathematics and Computer Science, Amsterdam (1988). | [14] R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra, \Job Shop Scheduling by Local Search", INFORMS Journal on Computing, 3(1996)302-317. | [15] T. Yamada and R. Nakano, \A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems", Parallel Problem Solving from Nature 2, R. M¨anner, B. Mandrick (eds.),North-Holland, Amsterdam, (1992)281-290.n |