Int J Adv Manuf Technol (2006) 30: 1126–1131
DOI 10.1007/s00170-005-0134-6

ORIGINAL ARTICLE

G. Prabhaharan · B. Shahul Hamid Khan · L. Rakesh

Implementation of grasp in flow shop scheduling

Received: 6 August 2004 / Accepted: 22 April 2005 / Published online: 8 December 2005

Abstract In this paper the problem of permutation flow shop scheduling is considered with the objective of minimizing the makespan. An attempt is made to schedule the jobs using a new approach called greedy randomized adaptive search procedure (GRASP). It is a competitive algorithm and is a metaheuristic for combinatorial optimization. The performance of GRASP is then compared with that of NEH heuristics. The comparison is carried out with the benchmark problems taken from Carlier (1978), Reeves (1995) and Taillard (1993).

Keywords Flow shop · GRASP · Makespan · Metaheuristic

1 Introduction

Many heuristics have been proposed for flowshop scheduling with the objective of minimizing the makespan [1–4]. Many metaheutistics such as genetic algorithm and simulated annealing [5] are also used to solve flow shop scheduling problems.

In this paper an attempt has been made to solve problems by using a technique called (greedy randomized adaptive search procedure (GRASP). We investigate the problem of scheduling in flow shop with the objective of minimizing the makespan. A comparative study is made to test the performance of GRASP with that of NEH [4] heuristics by solving benchmark problems taken from Carlier [6], Reeves [7] and Taillard [8].

2 Formulation of the flow shop scheduling problem

The flow shop scheduling problem consists of scheduling 'n' jobs with the given processing times on 'm' machines. The sequence of processing of a job on all machines is identical and unidirectional.

2.1 Objective function

The objective of this problem is to minimize makespan
Minimize {C max} Let
t_{ij} be the processing time of job i on machine j.
n be the total no of jobs to be scheduled.
m be the total no of machines in the flow shop.
$C_{[i,j]}$ be the completion time of job in position i on machine j.
C max=$C_{[n,m]}$=Makespan.
$\sum$ be the ordered set of jobs already scheduled out of n jobs; partial sequence.

2.2 Assumptions

The following assumptions are made in this permutation flow shop scheduling problem

(1) A set of 'n' multiple-operation jobs are available for processing at time zero.
(2) Each job requires 'm' operations and each operation requires different machines.
(3) Setup times for jobs are independent of job sequence and can be included in processing time.
(4) Job descriptors are known in advance.
(5) Once processing begins on a job, it is processed to completion without interruption (No preemption).

3 Description of GRASP

The greedy algorithm solves problems by making the choice that seems best at the moment. A greedy algorithm exhibits two properties namely, greedy choice property and optimal sub structure. Let

Z	Objective function.
α	Greedy parameter.
RCL	Restricted candidate list.

G. Prabhaharan (✉) · B. S. H. Khan · L. Rakesh
Production Eng'g Department, National Institute of Tech,
Trichirappalli, Tamil,
Nadu, India
e-mail: prabha@nitt.edu

r (σ)	Rank of element.
bias [r(σ)]	Bias function.
$\prod$ (σ)	Probability of selecting job.

We first initialize the completion time of a job on a machine equal to zero. This indicates the time of availability of a job in the flow shop. The objective function that we have considered here is minimizing the Makespan. A greedy parameter "α" is determined experimentally and its value ranges from zero to one. In this paper we have taken the value of "α" as equal to 0.5. Among the candidates, the best candidates are arranged in the restricted candidate list (RCL) according to the range and width.

Range = (Maximum time of completion of a job
– Minimum time ofcompletion of a job)

$$\text{Width} = \text{Range} \times \alpha \tag{1}$$

$$\text{RCL} = \{\text{Minimum}, \text{Minimum} + \text{Width}) \tag{2}$$

It is to be noted that by using the above formulae the GRASP algorithm works.

3.1 Brief review on GRASP

GRASP is a general procedure and its basic concepts are customized for the problem being solved. Basically GRASP consists of two stages namely construction phase and local search phase. However, in this work only the construction phase is considered. The construction phase of GRASP is essentially a randomized greedy algorithm. GRASP tries to capture good features of greedy and random constructions. It iteratively samples solution space using a greedy probabilistic bias to construct a feasible solution.

3.2 General structure of GRASP

3.2.1 GRASP–construction phase

In this phase a feasible solution is iteratively constructed one element at a time. Feo and Resende [12] has explained regarding GRASP construction phase. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements in a candidate list with respect to greedy function. A greedy parameter (α) is experimentally determined and its value ranges from zero to one. The heuristic is adaptive because the benefits associated with every element are updated at each iteration to reflect the changes brought on by the selection of the previous element. The probabilistic component of GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The best candidate list is called the restricted candidate list (RCL).

The pseudo code describes a basic construction phase of GRASP algorithm.

Procedure construct (g (.), α, x)

1. x = 0
2. Initialize candidate set C
3. While C ≠ 0 do
4. S_{min} = Min{g(T)\T∈C};
5. S_{min} = Max{g(T)\T∈C};
6. RCL = {s belongs to c\g (s) ≤ (S_{min} + Alpha (S_{max} − S_{min})
7. Select s at random, from the RCL;
8. X = XU {s}
9. Update candidate set C;
10. End while;
11. End construct;

The pseudo code shows that α controls the amount of greediness and randomness in the algorithm.

3.2.2 Bias functions

A standard GRASP uses random bias function. Various other bias functions like linear bias, logarithmic bias, exponential bias, and polynomial bias can be used in GRASP algorithm. However any probability distribution could be used to bias the selection toward some particular candidates. In this paper we have considered random bias for the execution of GRASP.

3.2.3 Structure of GRASP algorithm

Step 1:
Let us consider an empty set for initialization as is ∑.
Step 2:
Find out ∑ of the operations.
Step 3:
Compute the objective function (makespan) of the operations.
Step 4:
Find out the minimum and maximum value of the objective function.
Step 5:
Find out the range, i.e., (MAX–MIN).
Step 6:
Choose the α parameter (greedy value) value say 0.2, 0, 4, 0.5, 0.8 etc.
Step 7:
Find out the width. (Range×α)
Step 8:
Choose candidate for entry in RCL , if the Z ($\sum \sigma$) ≤ [MIN[Z($\sum \sigma$) + WIDTH]
Step 9:
Define rank 'r' for each operation in RCL.

Step 10:
Calculate rank for random bias function.
Step 11:
Sample and update the solution.

3.2.4 GRASP–local search phase

The solutions generated in GRASP construction phase can not guaranteed to be globally optimal with respect to simple neighborhood definition. Hence it is always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative function by successively replacing the current solution by a better solution in the neighborhood of the current solution.

The pseudo code below describes a basic local search procedure:

Procedure local (f (.), n (.), x}

1. H = {y belongs to n(x) \ f(y) < f(x)};
2. While H > 0 do
3. Select x belongs to H;
4. H = {y belongs to n (x) \ f(y) < f(x)}
5. end while;
6. end local;

3.2.5 Numerical illustration using GRASP

For illustration purpose five machines and a five jobs problem is considered and processing time (including set up time) are furnished in Table 1.

First initialized with time zero for all machines

0	0	0	0	0

Now let us take job 1 and calculate the time taken in all machines to complete. Similarly it is calculated for all jobs.

Job 1

2	6	12	20	30

Job 2

3	5	9	14	21

Job 3

4	7	8	12	17

Table 1 Processing time in minutes

Machines Jobs	1	2	3	4	5
1	**2**	**4**	**6**	**8**	10
2	**3**	**2**	**4**	**5**	7
3	**4**	**3**	**1**	**4**	5
4	**3**	**4**	**3**	**8**	4
5	3	6	5	7	3

Job 4

3	7	10	18	22

Job 5

3	9	14	21	24

Maximum completion time = 30
Minimum completion time = 17
Range = Maximum Completion time – Minimum Completion time = 30 − 17 = 13
Let alpha = 0.5(considering the mid value between zero to one)
Width = Range×alpha
= 13×0.5
= 6.5
RCL = {Minimum Completion time, Minimum Completion time + width}
= {17,23.5}

The jobs have completion time in the RCL range in found to be eligible to consider in construction. Now the range is 17 to 23.5. As per this, jobs 2, 3, 4 are eligible.

Choice 1:
We may choose any eligible job at random or by using the bias function and probability of select.
Choice 2:
Let ‘r’ be the rank of the eligible jobs assigned as per the completion time.

R[2] = 2
R[3] = 1
R[4] = 3

Accordingly the fitness of job 2 is 0.5, jobs 3 is 1, and job 4 is 0.3.

The linear bias is given by = 1/r.

Accordingly the fitness of job 2 is 0.5, job 3 is 1 and job 4 is 0.3.

Probability of job 2 = 0.5/1.8 = 0.28.
Probability of job 3 = 1/1.8 = 0.55.
Probability of job 4 = 0.3/1.8 = 0.17.

Now generate a random number ‘u’ such that
If u ≤ 0.28 then choose job 2.
If 0.28 ≤ u ≤ 0.55 then choose job 4 else choose job 3.

Assume job 2 is chosen.

Job 2

3	5	9	14	21

Job 21

5	9	15	23	33

Job (23)

7	10	11	15	20

Job (24)

6	10	13	21	25

Job (25)

6	12	17	24	27

Maximum completion time = 33
Minimum completion time = 20
Range = 13
When alpha = 0.5
Width = 13×0.5 = 6.5
RCL = {20, 26.5}
Job (23), (24) falls in this range
Assume job (23) is chosen.

Now keep job (23) as fixed. Try with various other combinations and follow the same procedure.

Job (23)

7	10	11	15	20

Job (231)

9	13	19	27	37

Job (234)

10	14	17	25	29

Job (235)

10	16	21	28	31

Range = 37−29 = 8
When alpha = 0.5
Width = 8×0.5 = 4
RCL = {29, 33}
Jobs (234), (2 3 5) falls in this range
Assume job (2 3 4) is chosen
Keep job (2 3 4) as fixed and try various other combinations.

Job (234)

10	14	17	25	29

Table 2 Makespan time

S.No	*n	**m	GRASP	NEH
1	11	5	8603	9053
2	13	4	8936	9185
3	12	5	9954	10584
4	14	4	10366	10261
5	10	6	10198	11010
6	7	7	8043	8378
7	8	8	9661	10029
8	20	5	1648	1754
9	20	5	1625	1598
10	8	9	11416	11589

Job (2341)

12	16	22	30	40

Job (2345)

13	19	24	31	34

Range = 40−34 = 6
Width = 6/0.5 = 3
RCL = {34, 37}
Job (2 3 4 5) falls in this range
Choose job (2 3 4 5)

Keeping job (2 3 4 5) as fixed and trying various other combinations,

Job (2345)

13	19	24	31	34

Table 3 Performance of GRASP Algorithm

S.No	*n	**m	Upper bound	% Deviation from upper bound value	
				NEH	GRASP
1	20	5	1278	25.35	18.5
2	20	5	1359	20.16	15.15
3	20	5	1081	51.9	45.5
4	20	5	1293	26.99	18.87
5	20	5	1236	24.8	10.68
6	20	5	1195	34.1	15.14
7	20	5	1239	29.05	16.78
8	20	5	1206	27.7	15.75
9	20	5	1230	32.35	13.98
10	20	5	1108	37.27	15.43
11	20	10	1582	37.5	22.75
12	20	10	1659	26.64	19.59
13	20	10	1496	37.16	20.05

*n=No. of jobs
**m=No. of Machines

Job (23451)

13	19	24	31	34

The sequence as per GRASP algorithm is 2-3- 4-5-1 and the makespan is 43.

3.2.6 Application of GRASP

There are various applications of GRASP. Some of them are in manufacture equipment selection, bus driver scheduling, vehicle routing, and flight scheduling.

4 NEH algorithm

The NEH algorithm (Nawaz, Enscore & Ham 1983) is widely regarded as the best performing heuristic for permutation flow shop problem [17]. The NEH algorithm can be summarized as follows:

(1) Order the *n* jobs by decreasing sums of total job processing time on machines.
(2) Take the first two jobs and schedule them so as to minimize the partial makespan as if there were only two jobs.
(3) For k = 3 to *n* do

Insert the *k*th job into the location in the partial schedule, among the k possible, which minimizes the partial makespan.

5 Performance analysis of GRASP

We present the performance evaluation of GRASP with respect to minimization of makespan. The test problems for evaluating GRASP and making a comparative study are taken from Carlier [6] and Reeves [7].These test problems have varying sizes with number of jobs varying from 5 to 20 and the number of machines varying from 5 to 20. The problems are solved using GRASP and NEH algorithm [4]. The results of evaluation are tabulated in Table 2.

Relative performance of GRASP, NEH for the benchmark problems given by carlier and Reeves for the makespan objective.

In Table 3 we have presented the relative performance of GRASP and NEH for various instances given by Taillard [8] for the objective of makespan.

6 Conclusions

In this paper, the GRASP algorithm for minimizing makespan in permutation flow shop has been investigated. This algorithm is tested using various benchmark problems then compared with the results of NEH algorithm and tabulated. The percentage deviation of makespan with upper bound value is also tabulated for the benchmark problems taken from Taillard [8]. These computational experiments indicate that the GRASP algorithm outperforms the traditional NEH algorithm, which is widely acknowledged as giving good quality solutions to permutation flowshop problems. Repeated application of the construction phase will yield diverse starting solutions for the local search, which may help us to seek any further improved solutions.

7 Summary

Attempts were being made to solve problems using GRASP for job shop scheduling problem. In the case of flow shops scheduling relatively few attempts have been made to solve problems using GRASP. GRASP may give optimal solution in the construction phase itself, but sometimes we may have to zero in for improved solution, i.e, to go for local search phase. The results clearly show that, in the problems investigated, GRASP is a much better and simpler technique and gives satisfactory results. This paper gives an insight into how GRASP works.

Acknowledgements We gratefully acknowledge the help, support, encouragement and valuable suggestions given by Dr. C. Rajendran, Professor, Department of Humanities and Social science, Indian Institute of Technology, Madras, India, for helping us to carry out this work in a successful manner.

References

1. Johnson SM (1954) Optimal two and three stage production schedules. Noval Res Logist Q 1:61–68
2. Ignall E, Schrage L (1965) Application of branch and bound technique to some flow shop scheduling problems. Oper Res 13:400–412
3. Campbell HG, Dudek RA, Smith ML (1970) A heuristic algorithm for the *n*-job, *m*–machine sequencing problem. Manage Sci 16:B630–B637
4. Nawaz M, Enscore EH (1983) A heuristic algorithm for the *m* machine, *n* job flow shop sequencing problem. OMEGA11, pp 91–95
5. Ischibuchi H, Tanaka SH (1995) Modified simulated annealing algorithm for the flow shop sequencing problems. Eur J Oper Res 81:388–398
6. Carlier J (1978) Benchmark problems for flow shop scheduling. Oper Res 12:333–351
7. Reeves CR (1995) A genetic algorithm for flow shop scheduling. Oper Res 22:5–13
8. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285
9. Holthaus O, Rajendran C (1997) Efficient dispatching rules for scheduling in a job shop. Int J Prod Econ 48:87–105
10. Rajendran C, Chaudhuri D (1991) An efficient heuristic approach to the scheduling of jobs in a flow shop. Eur J Oper Res 61:318–325
11. Paradolous PM, Pitsoulis LS, Resende MGC (1995) A parallel graps implementation for the quadractic assignment problem. pp 111–130

12. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
13. Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York
14. Gowrishankar K, Rajendran C, Srinivasan G (2001) Flow shop scheduling algorithms for minimizing the completion time variance and the sum of squares of completion time deviations from a common due date. Eur J Oper Res 132:643–665
15. Hart JP, Shogun AW (1987) Semi greedy heuristics an empirical study. Oper Res Lett 6:107–114
16. Rajendran C, Ziegler H (1997) An efficient heuristic for scheduling in a flow shop to minimize total weighted flow time of jobs. Eur J Oper Res 103:129–138
17. Taillard E (1990) Some efficient heuristic methods for the flow shop sequencing problem. Eur J Oper Res 47(1):65–74
18. Pinedo M (1995) SCHEDULING theory, algorithms and systems. Prentice- Hall, Englewood Cliffs, NJ
19. French S (1982) Sequencing and scheduling. Halsted, New York