Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals of DAAAM International

A NEW "GREEDY" SELECTIVE ALGORITHM FOR SOLVING FLOWSHOP SCHEDULING PROBLEMS

PACURAR, R[azvan] I[oan]; RADU, S[ever] A[drian] & ANCAU, M[ircea]

Abstract: *This paper presents the design of the "Greedy" selective heuristic algorithm and its results that were obtained running an optimization program developed in Visual C++ by the authors at the Technical University of Cluj-Napoca, Romania within a national research contract. The program has been tested on several problems, such as E. Taillard series. Parts of the results are presented and compared with the ones obtained using the NEH algorithm.*
Key words: *flow shop scheduling problem, heuristic algorithm*

1. INTRODUCTION

The NEH heuristic algorithm is known as one of the most important algorithms applied in the flow shop scheduling problem field. Its strategy is different from that of Johnson's algorithm and does not need the assignment of some specific indexes in order to sort-out the jobs in a certain way. When the NEH algorithm is applied, the jobs are sorted out in a descending way, by using the total manufacturing time adequate to each job particularly considered as sorting key-control. The first two jobs are analyzed at the beginning, the total manufacturing time being evaluated in both possible alternatives. Finally, the best order alternative is hold over. Further on, the third job is selected from the list, and the total manufacturing time is evaluated once again by placing the third job in each possible alternative combination, first at the beginning, second, in the middle and finally, at the end. Once again, the best order alternative is hold over. In the same way, all the remaining jobs are analyzed, compared in all possible alternatives and finally sorted out, so that the manufacturing time would be minimum at the end. To reach this objective, $n \cdot (n-1)/2 - 1$ evaluating operations are needed.

There are several other heuristic methods in the field, constructed on the basis of the NEH algorithm, with some particular differences that come mainly from the starting sequences (Framinan et al, 2003). For example, in the case of the SPIRIT heuristic algorithm, the manufacturing order is constructed by using an Insertion type strategy. The first two jobs from the list J_i ,J_k are selected at the beginning, so as the total manufacturing time needed would be minimum at the end. Further on, the next jobs are introduced randomly, in that particular position that corresponds to a minimum increase of the total manufacturing time.

There are also different algorithms used for solving the flow shop scheduling problems, such as the so-called "*improvement heuristics*", that starts from a processing sequence already established and can be improved afterwards by using different other procedures. These techniques are frequently based on a simple commutation between neighbor jobs, by using either the *Rapid Access with Close Order Search* (RACS) or the *Rapid Access with Extensive Search* (RAES), the initial sequence being created with Rapid Access, or by using the minimization of $C_{max,}$ as the basic criterion, etc. (Gupta & Stafford, 2006), (Agarwal et al., 2006), (Chakraborty & Laha, 2007), (Ruiz & Morato, 2005), (Ponnambalam et al., 2000).

2. PRINCIPLE OF THE "GREEDY" SELECTIVE HEURISTIC ALGORITHM

At the beginning, we start from an initial order randomly established J = { j_1 , j_2 ,…., j_N } and then we evaluate the total manufacturing time of all the ordered pair of jobs (j_i, j_k). There is N·(N-1) possible combinations to be analyzed. Let us assume that the minimum processing time is obtained in this case for the (j_k , j_i) order (Figure 1).

```
For i = 1 TO N-1
{
   For k = i+1 TO N
   {
    select jobs  j_i and j_k
    determine manufacturing time t_Partial
   if (t_Partial < t_Partial Min)
      {
       t_Partial Min = t_Partial ;
memorize jobs order ;
      }
       inverse jobs order j_i and j_k ;
      determine t_Partial;
      if (t_Partial < t_Partial Min)
      {
      t_Partial Min = t_Partial ;
      memorize jobs order
      }
   }
}
```

Fig. 1. The algorithm used for determining the first two jobs

(N-2) jobs still need to be analyzed in order to find the optimum sequence of launching the jobs into the manufacturing process. In order to do so, each of the (N-2) remaining jobs will be tested one after the other in order to determine the proper job that leads to the optimum sequence. Let us consider j_x as being one of the (N-2) jobs that needs to be verified. For this particular job, there are three possibilities to be verified, such as the following:

$$(\mathbf{j_x}, j_k, j_i)\ ;\ (j_k, \mathbf{j_x}, j_i)\ ;\ (j_k, j_i, \mathbf{j_x}) \qquad (1)$$

Further on, the procedure is repeated for all the other remaining jobs, for each possible order alternative, the total manufacturing time being calculated. In total, at this stage 3☺(N-2) possible alternatives are analyzed. From all these combinations, it will be selected the one that provides the minimum time of manufacturing. Let us assume that the best job combination in this case is (j_k , j_x, j_i). One may notice that the two initial jobs j_k and j_i remained in the same relative order,

j_k the first job and j_i positioned after the j_k job, even if this two jobs are not successive, between them, job j_x being now included. At this new stage, we have an optimum sequence constructed from three jobs. (N-3) jobs still need to be introduced. What comes next is the selection of the best candidate from the (N-3) remaining jobs, which gives us the minimum manufacturing time. Let us assume that j_y is one of these jobs. In this case, in a similar way as presented above, we have to verify four possible combinations, such as the following:

$$(\mathbf{j_y}, j_k, j_x, j_i);\ (j_k, \mathbf{j_y}, j_x, j_i);\ (j_k, j_x, \mathbf{j_y}, j_i) \text{ şi } (j_k, j_x, j_i, \mathbf{j_y}) \qquad (2)$$

The selection procedure is repeated in the same way with all the other jobs from the total (N-3) remaining ones. For each combination, the minimum manufacturing time is calculated. At this stage, 4☺(N-3) possible alternatives are analyzed. Once again, from all these combinations, it will be selected that combination constructed by four jobs, which corresponds to the minimum time of manufacturing. Further on, there still remain (N-4) jobs to be launched into manufacturing. In a similar way, the procedure will be repeated as presented above. Finally, for the last job that needs to be added to the optimum sequence of launching into manufacturing, N possible alternatives will be evaluated, after this stage, the algorithm being considered as reaching the end.

The principle of the "Greedy" selective heuristic algorithm has the highest complexity grade - $\Theta(n^3)$ - as compared to the other known algorithms.

3. RESULTS OBTAINED RUNNING THE OPTIMIZATION COMPUTER PROGRAM

The performance of the "Greedy" selective heuristic algorithm has been verified on several test problems, such as 90 problems from E. Taillard series. The obtained results are presented in Tables 1 and 2.

Test problem	Upper bound	Results		Calculated error [%]	
		NEH	GS	NEH	GS
Ta002-20x5	1359	1365	1367	0.44	0.58
Ta011-20x10	1582	1680	1600	6.19	1.13
Ta029-20x20	2237	2320	2289	3.71	2.32
Ta040-50x5	2782	2790	2787	0.28	0.17
Ta050-50x10	3091	3257	3206	5.37	3.72
Ta057-50x20	3672	3952	3963	7.62	7.18
Ta069-100x5	5454	5489	5485	0.64	0.56
Ta092-200x10	10494	10716	10625	2.11	1.24

Tab. 1. Best results obtained with "Greedy" selective algorithm

Test problem	CPU average time per problem [sec]	Average error [%]	
	SG	NEH	SG
20 x 5	0.062	3.246	3.365
20 x 10	0.095	4.587	5.643
20 x 20	0.203	3.721	5.460
50 x 5	3.12	0.723	2.047
50 x 10	6.09	4.567	5.485
50 x 20	14.8	8.329	8.606
100 x 5	86.1	0.476	0.945
200 x 10	3698	1.282	1.550

Tab. 2. Average CPU time and average error(after running 10x)

Each Taillard problem has been run 10 times on the computer. Table 1 and 2 present not only the best results we have obtained, but also the"upper bound" value and the error that has been calculated using equation (3), as compared to the "upper bound" value and the error that has been calculated for each Taillard problem, by using the NEH algorithm (Nawaz, M., et al,1983).

$$\text{Error} = \frac{\text{Result} - \text{UB}}{\text{UB}} \cdot 100\% \qquad (3)$$

The test problems were solved on a PC computer equiped with an Intel Pentium CPU of 1.6 GHz.

4. CONCLUSIONS

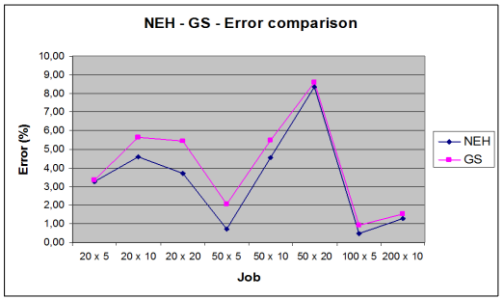

Fig. 2. Average error calculated for jobs solved by using NEH / GS algorithm

As one may observe from the results plotted in Figure 2, there are small differences between the performances obtained using the designed "Greedy" selective heuristic algorithm, as compared to the ones obtained by using the NEH algorithm, which is being considered as "the champion" of the heuristic algorithms. Some improvements are still needed to be done in the future, focusing mainly on the aim of reaching the same manufacturing process time at one specific stage, when a new job is required to be launched into manufacturing, with the constraint of maintaining the global optimum launching manufacturing time.

5. REFERENCES

Agarwal, A., Colak, S., Eryarsoy, E. (2006) Improvement heuristic for the flow-shop scheduling problem: An adaptive-learning approach. *European Journal of Operational Research* 169, p.801-815, ISSN: 03772217

Chakraborty, U.K., Laha, D. (2007) An improved heuristic for permutation flowshop scheduling. *Int. J. Information & Communication Techn.*, 1 (1), p.89-97, ISSN:1466-6642.

Framinan, J.M., Leisten, R., Rajendran, C. (2003). Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. *Internat. Journal of Production Research* 41 (1), p.121-148, ISSN 0020-7543.

Gupta, J.N.D., Stafford Jr., E.F. (2006) Flowshop scheduling research after five decades. *European Journal of Operational Research* 169, p.699-711, ISSN: 0377-2217

Nawaz, M., Enscore Jr., E.E., Ham, I. (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA, *The International Journal of Management Science*, 11 (1), p.91-95

Ponnambalam, S.G., Aravindan, P., Chandrasekaran, S. (2001) Constructive and improvement flow shop scheduling heuristics: Production Planning&Control 12(4), p.335-344, ISSN 1366-5871.

Ruiz, R., Maroto, C. (2005), A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research* 165, p.479-494, ISSN: 0377-2217