AN INVESTIGATION OF THE GROUP SCHEDULING HEURISTICS IN A FLOW-LINE CELL

Huzayyin, A.S.*, Badr, M.A. and Helal, M.E. *****

* Professor and Dean, Benha High Inst. of Technology
** Researcher, Mechanical Eng. Dept., The National Research Centre – Cairo
*** Demonstrator; Mechanical Eng. Dept., Benha High Inst. of Technology

ABSTRACT

A comparative study of group scheduling (GS) in a flow line cell is presented. Three simple scheduling heuristics are compared with two iterative improvement heuristics. The objective is minimizing makespan and total flow time separately. A number of modifications are proposed in order to explore the performance of the heuristics and to investigate the characteristics of the GS model. In addition, a timetabling procedure for multi-family cells is proposed, considering the presence of the zero processing times. Results showed that the proposed modification could improve the performance of the heuristics under study. The iterative improvement techniques were found appropriate for GS not only because of their superiority over the simple methods but because they can handle the phases' interaction in GS as well. The tabu search heuristic is found preferable to the simulated annealing heuristic.

KEYWORDS

Group Scheduling, Heuristics, Flow-Line Cell, Tabu Search, Simulated Annealing.

1. INTRODUCTION

Group scheduling (GS) is applied when parts are classified into different part families based on setup and processing requirements. The machines are assumed grouped into a manufacturing cell. The creation of part families leads to the creation of a two-phase scheduling model: scheduling part families (family phase) and scheduling jobs within families (job phase). A typical application of GS is the scheduling of a static flow line manufacturing cell [1-4]. The scheduling task is greatly simplified with GS in addition to the reduction of the setup times. Results generally indicated that the GS approach yields superior performance over the corresponding traditional (single-phase) models [5,6].

GS problem is Non-Polynomial complete. An optimal solution based on permutation schedules with respect to makespan was obtained by Hitomi and Ham in 1976 using branch-and-bound. Their work is a two-stage application of the branch and bound model of Ignall and Scharage developed in 1965 for the general flow shop problem [3,4,7]. The main feature of the methodology is that schedules in both phases must be determined simultaneously in order to achieve optimality [1]. However, as optimization techniques are computationally burdensome, heuristic techniques are used instead to obtain near optimal solutions. The traditional flow line heuristics can be modified for the flow line GS problems where heuristics

are executed in two stages, family phase and job phase. Grasso et-al. [4] proposed a general framework to modify the traditional heuristics to GS applications. The problem according to them is simplified into the separated determination of job sequences within families, and the determination of family sequences. This does not take into account the possible interaction between the two phases and hence leads to sub-optimal results. Still, the simplification is useful in order to derive rapid and efficient GS heuristics from the traditional heuristics. Makespan is the criterion in consideration.

Modified versions of the known traditional flow line scheduling heuristics were developed in a way similar to Grasso's framework. Hitomi [8] developed a GS version of Petrov's method. Wemmerlov and Vakharia [7] developed GS versions for Campbell, Dudek and Smith' s heuristic (CDS) and for Nawaz, Enscore and Ham' s heuristic (NEH).

A drawback of the above simple GS heuristics is that they do not consider the phase's interaction. In addition, the number of generated solutions is small. With the increase of computer capabilities, researchers developed iterative improvement techniques to solve the GS problem. The iterative methods seem able to consider the phase's interaction, while enumerating a much larger number of feasible solutions. Two of the generic techniques applied are the simulated annealing (SA) and the tabu search (TS) approaches.

SA is a randomized iterative improvement technique that was originally developed as a simulation model for a physical annealing process. Basic concepts for the algorithm were developed by Kirkpatrick et-al. in [9]. SA starts off from an arbitrary initial configuration associated with a cost given by a relevant cost function. In each iteration, by slightly perturbing the current configuration a new configuration is generated. Difference in cost between the two configurations is compared with an acceptance criterion that tends to accept improvements but also admits, in a limited way, deteriorations in cost in order to avoid being traped in a local optimum. Initially, the acceptance criterion is taken such that deteriorations can be accepted at a high probability. As the process proceeds, the probability of accepting deteriorations decreases until zero [10].

TS is a meta-strategy that has its origins in combinatorial optimization procedures applied to some non-linear problems in the late 1970s. Strategic principles of TS in a broader sense have been laid out in [11,12]. It starts from a random initial solution and performs a set of alterations (moves) on it to move to neighbour solutions in a search for a better result. The basic components of TS are defined as follows. A tabu list is comprised of moves that are not allowed (tabu moves) to be performed at the current iteration. Function of the list is to prevent returning to previously visited solutions to avoid cycling. The aspiration level function whose role is to provide added flexibility to choose good moves, by allowing the tabu status of a move to be overridden if this aspiration level is fullfilled. A Long term memory (LTM) is employed to achive regional intensification and global divesivication of the search. LTM records and compares features of the best trial solutions generated during a particular period of search. Features that are common are taken to be regional attributes of a good solution. The method then seeks new solutions that exhibit these features (intensification). Or LTM may be used to guide the process to regions that contrast with those examined so far (diversification).

Vakharia and Chang [2] proposed a SA heuristic to minimize makespan in multi-family cells. They used an acceptance probability that is independent of the change in the objective function value. The heuristic spends 90% of the search efforts in the job phase and 10% in the family phase. It starts from an initial random solution and performs a pair-wise interchange of

jobs or families. Makespan is checked and the acceptance probability is then used to accept or reject an inferior solution if reached. Initial value of the acceptance probability is 50% and it is reduced by a constant amount each iteration until reaching zero.

Kapov and Vakharia [13] proposed a TS heuristic to minimize makespan in the multi-family cells. An initial random schedule is generated and moves are checked in a pair-wise interchange of jobs or families. A best move is identified and performed at each iteration given it is not a tabu move. The tabu list is a record of a predefined number of recent moves. It is updated each iteration by appending the last performed move and deleting the oldest one. A LTM is used to rerun the procedures five times by generating new initial solutions. LTM is FxF frequency matrix that contains records about the number of times a family occupied a certain position in the trial schedules during the search process, where F is the number of families. For a new start, a new family' schedule is generated by letting each family be in the position in the schedule it occupied for the largest number of times (intensification). Only a new schedule for part families is generated while jobs schedules are randomly reset. The aspiration level function allows a tabu move to be performed if an overall improvement is got.

2. OBJECTIVES OF THE CURRENT STUDY

2.1. Proposed timetabling procedure

Sridhar and Rajendran [14-16] notified that a feature of cells is the presence of some zero processing times for some jobs[1]. They showed that disregarding this yields erroneous jobs start and finish dates, incorrect information about machine availability, and overestimates of makespan and total flow time. They presented a modified timetabling formulation that accounts for the zero times. Their work considers a single-family cell.

For the multi-family situation, Hitomi [8] and Kapov and Vakharia [13] proposed two different timetabling procedures. Both neglected the zero times. The procedure proposed by Kapov and Vakharia could not be implemented. It contained errors such that some valuse calculated in some latest seteps of calculations are needed in eariler steps [18].

The following timetabling procedure is proposed for the multi-family cells considering the presence of the zero times. Assuming a flow cell of M machines for the processing of F part families each contains n_i jobs, let the family index be i (i =1,2...F) and jobs in family i indexed by j (j =1,2...n_i). The setup time of family i on machine k (k =1,2...M) is S_{ik} and the processing time of job j in family i on machine k is P_{ijk}. Let $Start_{i,j,k}$ be the start time for job j in family i on machine k and $SetStart_{i,k}$ setup time start for family i on machine k.

For i = 1, 2,... F , For j = 1, For k = 1, 2, ... , M

$$Start_{i,1,k} = \begin{cases} \max \begin{cases} Start_{i,1,kk} + P_{i,1,kk} \\ Start_{ii,jj,k} + P_{ii,jj,k} + S_{i,k} \times Z_1 \end{cases} & \text{If } P_{i,1,k} > 0 \\ 0 & \text{Otherwise} \end{cases}$$

Where : kk Last machine that job 1 in family i was processed on.
jj The job preceeds job 1 in family i on machine k.

[1] In 1974, Baker [17] mentioned the possibility that a job may not be processed on some machines in the general flow shop and he assigned zero processing times for such cases.

ii Family containing job jj.

Z_1 Binary variable such that $= \begin{cases} 1 & \text{If } P_{i,1,k} > 0 \\ 0 & \text{If } P_{i,1,k} = 0 \end{cases}$

For i = 1, 2,... F , For j = 2,3,…,n_i , For k = 1,…, M

$$Start_{i,j,k} = \begin{cases} \max \begin{cases} Start_{i,j,kkk} + P_{i,j,kkk} \\ Start_{iii,\,jjj,k} + P_{iii,\,jjj,k} + S_{i,k} \times Z_2 \end{cases} & \text{If } P_{ijk} > 0 \\ 0 & \text{Otherwise} \end{cases}$$

$$SetStart_{i,k} = Start_{i,jjjj,k} - S_{i,k}$$

Where :

kkk Last machine job j in family i visited.

jjj The job preceeds job j in family i on machine k.

iii Family containing job jjj.

jjjj First job in family i having a non-zero time on machine k.

Z_2 Binary variable such that $= \begin{cases} 1 & \text{If iii} < \text{i} \\ 0 & \text{otherwise} \end{cases}$

$$\text{Makespan} = \max_{k=1,2,\ldots,M} \{ Start_{v,t,k} + P_{v,t,k} \}$$

$$\text{Total Flow Time} = \sum_{i=1}^{F} \sum_{j=1}^{n_i} Finish_{i,j,lk}$$

Where :

t Last job processed on machine k

v Family containing job t

lk Last machine in the cell that job J_{ij} was processed on.

2.2. Modifying the GS heuristics

The three simple GS heuristics of Hitomi [8], CDS and NEH [7], and the two iterative improvement techniques of SA [2] and TS [4] are selected for the study. Modified versions of each of them are proposed. The details of the proposed modifications and description of the modified versions are found in [18].

Basically, the modifications to the simple methods were to test Rajendran's modification [15,16] that suggests dividing the scheduling indices by the number of the non-zero times for the job. This is to account for the zero processing time but no explanation is provided in [15,16]. This was tested and found ineffective [18]. In addition, an iterative version of CDS was proposed. It was found to be remarkably superior to the original CDS, which indicates the importance of taking the two-phase nature of GS in consideration. However, its superiority is limited by the finite number of solutions enumerated by CDS [18].

The iterative improvement techniques were found superior to the simple methods. They are considered in the following analysis. Three modifications to SA and two to the TS are proposed. The modified heuristics are described below.

In **SA-M-1,** a change-dependent acceptance probability version of the original SA [2] is proposed. The standard form of the acceptance probability [9,10] is employed. The advantage of using change-dependent acceptance probability as defined in the listing below, is that solutions which cause drastic changes in the objective function value are avoided as iterations proceed [14]. In **SA-M-2,** slight modifications in Steps 6 and 7 in the original SA is suggested to increase the efficiency of the search process. The idea is to prevent the use of two successive random numbers of the same value so that to avoid reversing (cancellation) of the last performed perturbation operation during the current operation, hence to avoid wasting efforts. In **SA-M-3,** the control on the random number behaviour in SA-M-2 is added to Steps 6 and 7 in SA-M-1. Being the best SA version (as shown later) SA-M-3 is listed below [18].

Step 1. Set X = 25, Y = 50, AP_o = 0.5, r = 0.9, and set the GP value.

Step 2. Generate a random initial schedule. This includes a complete sequence for all jobs (Ω^o), a family sequence (τ) and a sequence for jobs in each family (μ_i); i = 1, 2 , ... , F. Let this be current solution with total flow time $Flow^o$. Let Ω^* be the incumbent solution with total flow time $Flow^*$. Set $\Omega^* = \Omega^o$ and $Flow^* = Flow^o$.

Step 3. Let X = rX. If X ≤ 1.62 then stop, else set y = 0 and continue.

Step 4. Set y = y + 1. If y > Y then go to Step 3, else go to Step 5.

Step 5. Generate a random number v (0 ≤ v ≤ 1). If v ≥ GP go to Step 7, else go to Step 6

Step 6. Generate a random number v1 (1 ≤ v1 ≤ F). **If** v1 = last v1 and previous perturbation was a families interchange, then regenerate v1, **Else if** v1 = last v1 and previous perturbation was a jobs interchange and no change in current sequence has occurred in that perturbation, then regenerate v1, **Else** interchange the families in positions v1 and v1+1 (if v1 = F, interchange the families in positions F and 1) and get a new family sequence τ^1. Based on τ^1 specify a new complete job sequence Ω^1 and calculate its total flow time $Flow^1$.

(a) **If** $Flow^1 > Flow^*$ then go to (b). **Else** let $\Omega^* = \Omega^1$, $Flow^* = Flow^1$, go to (b).

(b) If $Flow^1 > Flow^o$ then let $\Delta = Flow^1 - Flow^o$, calculate the acceptance probability $AP_x = EXP(-\Delta / X)$ and go to (c). **Else** let $\tau = \tau^1$ and $\Omega^o = \Omega^1$ in the current solution, and set $Flow^o = Flow1$ and go to Step 4.

(c) Generate a random number v2 (0 ≤ v2 ≤ 1). If v2 ≥ AP_x go to Step 4, **Else** let $\Omega^o = \Omega^1$, $\tau = \tau^1$ in current solution, set $Flow^o = Flow^1$ and go to step 4.

Step 7. Generate a random number v3 (1 ≤ v3 ≤ N) where N is total number of jobs. **If** v3 = last v3 and previous perturbation was a jobs interchange, regenerate v3, **Else if** v3 = last v3 and previous perturbation was a families interchange and no change occurred in that perturbation, then regenerate v3, **Else** let f_1 be the family in which job v3 is included. Interchange job in positions v3 and v3+1 (if v3 is the last in f_1, interchange jobs in positions v3 and 1) in Ω^o. Let the new sequence be $\mu^1_{f_1}$ for family f_1 and new complete sequence be Ω^1 with total flow time $Flow^1$.

(a) **If** $Flow^1 \geq Flow^*$ then go to (b), **Else** let $\Omega^* = \Omega^1$ in the incumbent solution, set $Flow^* = Flow^1$ and go to (b).

(b) **If** $Flow^1 \geq Flow^o$ then let $\Delta = Flow^1 - Flow^o$, calculate the acceptance probability $AP_x = EXP(-\Delta / X)$ and go to (c). **Else** let $\mu_{f_1} = \mu^1_{f_1}$, $\Omega^o = \Omega^1$ in the current solution, and set $Flow^o = Flow^1$ and go to Step 4.

(c) Generate a random number v2 (0 ≤ v2 ≤ 1). If v2 ≥ AP_x go to Step 4, **Else** $\mu_{f_1} = \mu^1_{f_1}$, $\Omega^o = \Omega^1$ in current solution, set $Flow^o = Flow^1$, and go to Step 4.

Tne GP parameter controls the amount of search efforts given to each scheduling phase. Its value of 0.1 in [2] leads to spending 10% of efforts to the family phase and 90% to the job phase. Alison [3] stated that the family phase is more worthy. To investigate this GP will be given values of 0.1, 0.3, 0.5, 0.7, and 0.9. This is used for all SA versions.

In **TS-M-1,** when generating the new restart schedule, the current jobs sequences within families are kept instead of being randomly regenerated. The concept is to make use of the search efforts in the job phase. LTM is used for the same purpose in the family phase. In **TS-M-2,** long term memories for jobs within the families are developed and used to complete the new restart schedule by generating restart jobs sequences within families as done in the family phase. Actually, original TS is completed rather than being modified. Being best TS version (as shown later) TS-M-1 is described below. Variable list sizes are used. Two types of tabu list are needed for the two phases. In the family phase, the initial f-tabu list size is Int(F/2), decreased size is Int(F/3) and the increased size Int(F/0.5). In the job phase the initial j-tabu list size is Int(N/F), decreased size is Int(N/2F) and the increased size Int(N/0.5F). N is the total number of jobs.

Step 1. Initialize the f-tabu-size (tabu list for family phase) and j-tabu-size (tabu list for job phase). Set the number of LTM restarts = 5. Set LTM matrix = **0**.

Step 2. **If** LTM = **0**, generate a random families sequence, **Else** generate a families sequence using LTM, and for each family, generate a jobs sequence using LTM_i. Let this be current solution Ω^o with a total flow times $Flow^o$. Let Ω^* represent the incumbent solution with total flow time $Flow^*$. Set $\Omega^* = \Omega^o$, $Flow^* = Flow^o$. Set LTM = LTM + 1 and $LTM_i = LTM_i + 1$ for all i = 1,2,..., F.

Step 3. Start counting iterations for family exchange. Set f-iter = 0.

Step 4. Stopping criterion for family exchanges:

(a) **If** no improvment in the last 5F itaerations with the initial f-list size then decrease the f-list size and go to (b).

(b) **If** no improvment in the last 2F itaerations with the decreased f-list size then increase the f-list size and go to (c).

(c) **If** no improvment in the last 3F itaerations with the increased Size then set the list size to its initial value and go to Step 6.

If at any poine therse an improvement then go to Step 5

Step 5. **Family exchange phase of search.** Evaluate completely the neighbourhood N_i (Ω^o) and select the best exchange of families. Denote the new complete sequence by Ω^1 and its makespan by $Flow^1$. If $Flow^1 < Flow^*$ then set $\Omega^* = \Omega^1$ and $Flow^* = Flow^1$. Set $\Omega^o = \Omega^1$, $Flow^o = Flow^1$ and go to Step 4.

Step 6. Start counting iterations for job exchanges. Set j-iter = 0 and go to Step 7.

Step 7. Stopping criteria for job exchanges. Set j-iter = j-iter + 1.

(a) **If** no improvment in the last Int(N/3) iterations with the initial j-list size then decrease the j-list size and go to (b).

(b) **If** no improvment in the last Int(N/3) itaerations with the decreased j-list size then increase the j-list size and go to (c).

(c) **If** no improvment in last Int(N/3) itaerations with the increased size then set the list size its initial value and go to Step 9.

If at any point there is an improvement then go to Step (8)

Step 8. **Job exchange phase of search.** Evaluate completely the neighbourhood N_j (Ω^o) and select the best exchange of jobs. Denote the new complete sequence by Ω^1 and its makespan by $Flow^1$. If $Flow^1 < Flow^*$ then set $\Omega^* = \Omega^1$, $Flow^* = Flow^1$. Set $\Omega^o = \Omega^1$, $Flow^o = Flow^1$ ang go to step 7.

Step 9. **If** the incumbent solution was changed during job phase, then go to Step 3. **If** the required number of LTM restarts was performed then stop, **Else** go to Step 2.

3. COMPARISON OF THE SCHDULING HEURISTICS

For Carrying out the comparison of the GS heuristics, GS problems of various sizes are randomly generated. Data configuration is similar to that used in [13,2]. 30 problems for each of 8 problem sizes were generated. Written as (F × M × n_i), problem sizes are (3×3×3), (3×4×5), (4×4×4), (6×5×4), (5×5×5), (6×6×6), (5×6×8), and (8×8×8).

Jobs processing times are integer random variables uniformly distributed in U(1,10). To generate the zero times a uniform random number is sampled, if it is less than or equal to 0.2 a zero time is used. Hence 20% of job processing times are set to zero. This percentage is used in [14,15,16]. The family setup times are integer random variables uniformly distributed in the three sets U(1,20), U(1,50) and U(1,100) so that to study the impact of the different values for the family setup time to job processing time ratios (S/R) of approximately 2, 5 and 10 respectively.

Scheduling objectives are minimizing total flow time (sum of completion times of jobs [14,15]) and minimizing makespan separately. Makespan is commonly used however, Rajendran and Sridhar [14,15,16] notified that total flow time is more relevant criterion in cells. Better reduction in total scheduling costs is achieved by minimizing total flow time than minimizing makespan [16,17].

A measure of performance is established as follows. The total flow time and makespan obtained by the original Hitomi's heuristic are standardized to be 100. Then the average total flow time or the average makespan for each heuristic is compared with respect to that of Hitomi. For instance, let F_{Hitmoi} and F_X represent the average total flow times obtained by Hitomi and the X heuristic, respectively, then the relative total flow time for X is denoted by $RELF_x$ (similary relative makespan is $RELM_X$) and is given by: $RELF_x = (F_x / F_{Hitomi}) \times 100$. A value below 100 will indicate that X outperforms Hitomi and is preferred to it. And generally lower values are for better performance. Heuristics are coded in Quick Basic 4.5 and expermental calculations are performed on a Pentium 100 MHz PC.

4. RESULTS

The complete set of results of the comparison is found in [18]. Results showed that the iterative improvement methods are superior to the simple methods for all conditions. The iterative methods investigate a much larger number of solutions and are able to account for the phases' interaction.

The simulated annealing heuristics. Figs.1 and 2 show the performance of SA-M-2 at the different values of GP, at S/R = 5 for total flow time and makespan respectively. It is observed there are no significant differences for the different GP values. However GP of 0.5 and 0.7 gives relatively better results in most cases. GP of 0.9 is then preferred to 0.3. This means that giving the majority of the search efforts to the family phase is preferable. This is true for all S/R and is true for the other SA versions at less differences for SA-M-1 and SA-M-3. This is also true for minimizing makespan at less important differences than for total flow time in general. In all, GP of 0.1 used in [2] gives the least performance.

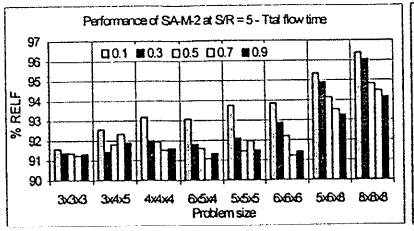


Fig.1 Effect of GP values on SA-M-2 Total flow time

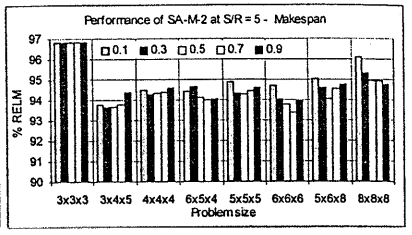
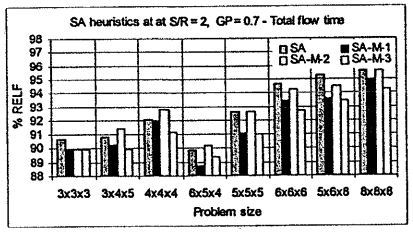
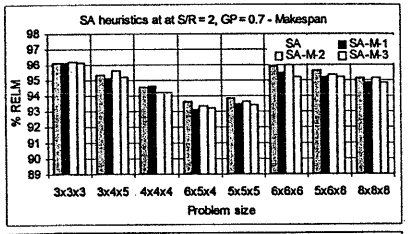
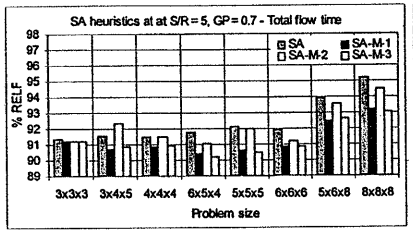
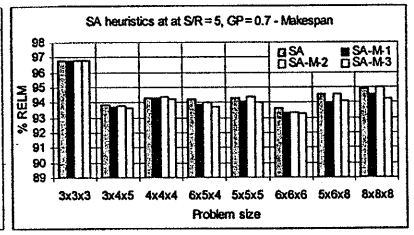


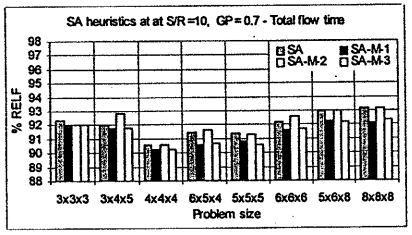
Fig.2 Effect of GP values on SA-M-2 makespan











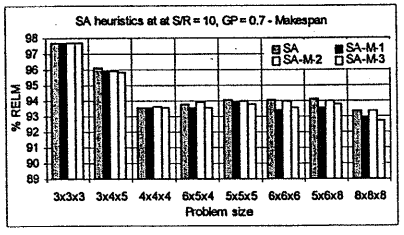


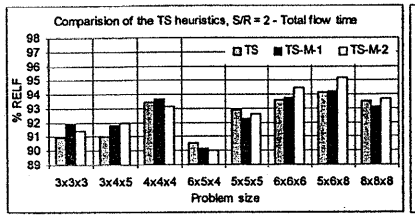
Fig.3 Performance of SA methods GP = 0.7, total flow time

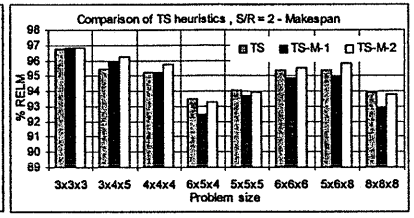
Fig.4 Performance of SA methods GP = 0.7, makespan

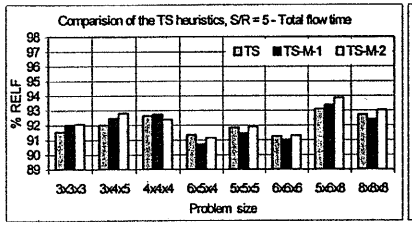
Figures.3 and 4 compare the four versions of SA for total flow time and makespan respectively at GP = 0.7 for all S/R. It can be observed that SA-M-1 and SA-M-3 are better than the original SA and SA-M-2 for the two objectives. This shows that the change-dependent acceptance probability is more efficient than the change-independent probability used in [2]. It is found also that SA-M-2 is slightly better than the original SA for about 58% of cases (for all GP)[18]. Similarly SA-M-3 is better than SA-M-1 for about 60% of cases

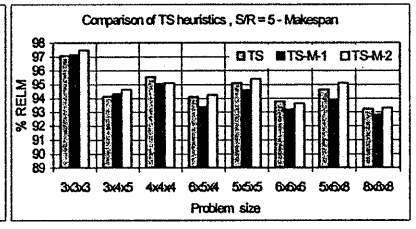
[18]. That is some control on the effect of the random numbers behaviour in SA is recommended. The proposed SA-M-3 is thus the preferred SA version. Nevertheless, the performance of the SA versions is generally tending to be inferior at the larger problem sizes. This is clearer for total flow time and at lower S/R values.

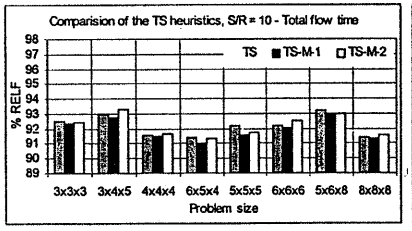
The tabu search heuristics. No much difference is observed among the TS versions. Still, the proposed TS-M-1 is generally better than the original TS and TS-M-2 for about 66% of cases. At S/R of 10 TS-M-1 is better all the time. This can be observed in Figs. 5 and 6. The TS methods are relatively more robust than SA when increasing problem sizes.











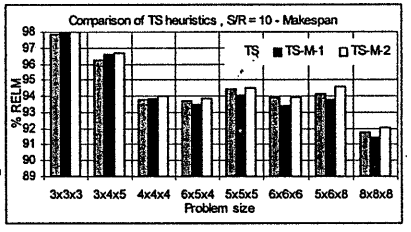


Fig.5 Comparison of TS methods S/R = 5 , total flow time

Fig.6 Comparison of TS methods S/R = 5 , makespan

By counting the number of times a TS version is better than the others it was found that [18], the original TS ranks second after TS-M-1, and TS-M-2 is the last. Using a relatively good initial solution, TS-M-2 could outperform the original TS. Still, TS-M-1 is better than both of them for most of time [18]. Hence, it is the complete LTM; in the two phases in TS-M-2; is the reason that it came forward to the original TS when using a relatively good initial solution. Nevertheless, TS-M-1 is always better than TS and TS-M-2, while using a partial LTM in the family phase and making use of the search efforts in the job phase by simply keeping the current jobs sequences during restarts. It thus possible to conclude that LTM is needed in both scheduling phases, however, including frequency information only in LTM as in [11] is not

sufficiently effective. Other search-based information may be concerning phases' interaction, should be included in the LTM.

Thus the SA-M-3 for GP of 0.7 (SA-M-3.7) and TS-M-1 are identified to be preferable to the other heuristic versions for the current study. Figs.7 and 8 presents a comparison between the two methods. A better level of improvement over the reference value is observed for total flow time than for makespan for both methods. No significant difference is observed between the two methods. However it can be seen that SA-M-3.7 is slightly better for total flow time. For makespan, the two methods are approximately equivalent. Nevertheless, TS-M-1 is generally better than SA-M-3.7 at the larger problem sizes for the two objectives. This is true for all S/R values [18].

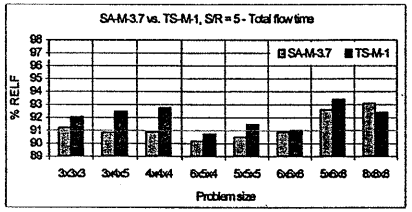


Fig.7 SA-M-3.7 vs. TS-M-1 , S/R = 5 total flow time

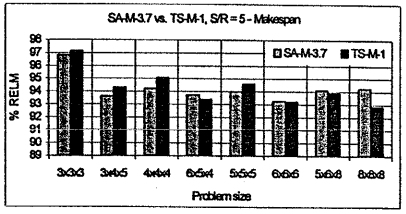


Fig.8 SA-M-3.7 vs. TS-M-1 , S/R = 5 makespan

TS-M-1 seems to be more robust to increasing problem sizes. It offers the ability to redefine its components (LTM, tabu lists and the aspiration level function) so as to improve its performance incorporating search-based information, which is not available for SA-M-3. Consequently as TS-M-1 is more robust and flexible, it is generally preferred to the SA-M-3.7 in this study. SA-M-3.7 may be used for scheduling with respect to total flow time at the relatively smaller problems.

5. CONCLUSIONS

GS model was investigated while studying the relative performance of selected simple and iterative improvement GS heuristics in a static flow line cell. Besides, a time-tabling procedure that can account for the presence of zero processing times in a multi-family cell, is proposed. The major conclusions of the study are:

- The two-phase nature of GS should be considered in the heuristic methods in order to compensate for the possible phase's interaction.
- The iterative improvement techniques were found preferable to the simple methods, not only because of their superior performance but because they can consider the phases' interaction in GS as well.
- The proposed modifications to the iterative GS heuristics studied provided slight, yet observable improvements over the original formulations of SA, and TS.
- SA-M-3.7 and TS-M-1 are relatively better than the other methods under study. TS is found to be more robust than SA while offering the possibility to redefine its components to include more relevant search-based information thus to increase its efficiency.
- The change-dependent acceptance probability in the SA is more efficient than the change-independent acceptance probability. The change-dependent acceptance probability can avoid solutions that results in drastic changes in the objective function value thus to force the procedure to converge, while avoiding local optimality. The probability of accepting such solutions (non-improving) would be very low due to the large value of the change in the objective function value.
- The possibility of the zero processing times in cells should be in consideration during time tabling calculations. Otherwise, erroneous time tabling information and overestimation of makespan and total flow time would be obtained. The proposed time tabling procedure for the multi-family cell was found effective in accounting for the zero times and removing their effects.

REFERENCES

1. Hitomi, Ham, Yoshida, "Group Technology", Kluwer-Nijhoff, Hingham, MA, (1985).
2. Vakharia A. J., Chang Y., "A Simulated Annealing Approach to Scheduling a Manufacturing Cell", Naval Research Logistics, Vol. 37, pp. 559-577, (1990).
3. Allison J. D., "Combining Petrov's Heuristic and The CDS Heuristic in a Group Scheduling Problems", Computers and Industrial Engineering, Vol. 23, pp. 457-461,(1990).
4. Grasso, Masanta, Piacentini, "Heuristic Procedures for Group Scheduling"; 4th Cairo Univ. MDP Conf., Dec. pp. 629-636, (1988).
5. Mahmoodi, Dooley K. J., "A Comparison of Exhaustive and Non-exhaustive Group Scheduling Heuristics in a Manufacturing Cell", Int. J. of Prod. Res., Vol. 29, No. 9, (1991).
6. Ruben R. A., Mosier C. T., Mahmoodi F., "A Comprehensive Analysis of Group Scheduling Heuristics in a Job Shop Cell", Int. J. of Prod. Res., Vol. 31, No. 6, (1993).
7. Wemmerlov U., Vakharia A. J., "Job and Family Scheduling of a Flow-Line Manufacturing Cell: A Simulation Study", IIE Transactions, Vol. 23, No. 4, (1991).
8. Hitomi K., "Group Scheduling with Interactive Computer Graphics", Memories of the Faculty of Eng., Kyoto Univ., Vol. 50, No. 3, (1988).
9. Kirkpatrick S., Gelatt C. D., Vecchi M. P., "Optimization by Simulated Annealing", Science, Vol. 220, No. 4598, 13 May (1983).
10. Schuur P. C., "Classification of Acceptance Criteria for the Simulated Annealing Algorithm", Mathematics of Operations Research, Vol. 22, No. 2, May (1997).
11. Glover F., "Tabu Search: Part I", ORSA J. of Computing, Vol. 1, No. 3, pp. 190-206, (1989).
12. Glover F., "Tabu Search: Part II", ORSA J. of Computing, Vol. 2, No. 1, pp 4-32, (1990).
13. Kapov, Vakharia A. J., "Scheduling a Flow-Line Manufacturing Cell : a Tabu Search Approach", Int. J. of Prod. Res., Vol. 31, No. 7, (1993).
14. Sridhar J., Rajendran C., "Scheduling in a Cellular Manufacturing System: a Simulated Annealing Approach", Int. J. of Prod. Res., Vol. 31, No. 3, (1993).
15. Sridhar J., Rajendran C., "A Genetic Algorithm for Family and Job Scheduling in a Flowline-Based Manufacturing Cell", Computers and Industrial Engineering, Vol. 27, No. 1-4, pp. 469-472, (1994).
16. Rajendran C., "A Heuristic for Scheduling in Flow Shop and Flowline-Based Manufacturing Cell with Multi Criteria", Int. J. of Prod. Res., Vol. 32, No. 11, (1994).
17. Baker, K. R., "Introduction to Sequencing and Scheduling", John Wiely and Sons, New York, (1974).
18. Helal, M., "Investigating Group Scheduling in a Flow Line Based Manufacturing Cell", Unpublished Master thesis, Benha High Inst. of Technology, (1999).