Parallel Hybrid Metaheuristics for the Scheduling with Fuzzy Processing Times*

Wojciech Bożejko[1], Michał Czapiński[2], and Mieczysław Wodecki[2]

[1] Institute of Engineering, Wrocław University of Technology
Janiszewskiego 11-17, 50-372 Wrocław, Poland
wojciech.bozejko@pwr.wroc.pl
[2] Institute of Computer Science, University of Wrocław
Przesmyckiego 20, 51-151 Wrocław, Poland
mwd@ii.uni.wroc.pl

Abstract. In this paper, parallel simulated annealing with genetic enhancement algorithm (HSG) is presented and applied to permutation flow shop scheduling problem which has been proven to be $\mathcal{NP}$-complete in the strong sense. The metaheuristics is based on a clustering algorithm for simulated annealing but introduces a new mechanism for dynamic SA parameters adjustment based on genetic algorithms. The proposed parallel algorithm is based on the master-slave model with cooperation. Fuzzy arithmetic on fuzzy numbers is used to determine the minimum completion times $C_{\max}$. Finally, the computation results and discussion of the algorithms performance are presented.

1 Introduction

Practical machine scheduling problems are numerous and varied. They arise in diverse areas such as flexible manufacturing systems, production planning, communication, computer design, etc. A scheduling problem is to find sequences of jobs on given machines with the objective of minimizing some function. In a simpler version of the problem, flow shop scheduling, all jobs pass through all machines in the some order. In this paper, we deal with another special version of the problem called a permutation flow shop (PFS) scheduling problem where each machine processes the jobs in the same order. The PFS problem belongs to the NP-hard class problems, however a solution of such a problem is usually made using heuristic approach that converges to a locally optimal solution.

In recent studies, scheduling problems were fuzzificated by using the concept of fuzzy due date and processing times. In paper Dumitru and Luban [3] investigate the application of fuzzy sets on the problem of the production scheduling. Tsujimura et al. [12] present the branch and bound algorithm for the three machine flow shop problem when job processing times are described by triangular fuzzy numbers. Especially fuzzy logic application on the scheduling problems (by using fuzzy processing times) is presented in papers: Ishibuschi and Murata [6], Izzettin and Serpil [5] and Peng and Liu [9].

* The work was supported by MNiSW Poland, within the grant No. N N514 232237.

L. Rutkowski et al. (Eds.): ICAISC 2010, Part II, LNAI 6114, pp. 379–386, 2010.

In this study, flow shop scheduling problem of the typical situation of the flexible production systems which occupy a very important place in recent production systems are taken into consideration with fuzzy processing time.

2 Flow Shop Scheduling

The permutation flow shop problem can be formulated as follows. Each of n jobs from the set $J = \{1, 2, \ldots, n\}$ has to be processed on m machines $1, 2, \ldots, m$ in that order. Job $j \in J$, consists of a sequence of m operations; operation O_{jk} corresponds to the job j processing on machine k during an uninterrupted processing time p_{jk}. Assumptions:

(a) for each job only one operation can be processed on a machine,
(b) each machine can process only one job at a time,
(c) the processing order is the same on each machine
(d) all jobs are available for machine processing simultaneously at time zero.

We want to find a schedule such that the processing order is the same on each machine and the maximum completion time is minimal.

The flow shop problem is NP-complete and thus it is usually solved by approximation or heuristic methods. The use of simulated annealing is presented. e.g., in Osman and Potts [8], Bożejko and Wodecki [1] (parallel algorithm), tabu search in Nowicki and Smutnicki[7], Grabowski and Wodecki [4], and genetic algorithm in Reeves [11].

Each schedule of jobs can be represented by the permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ on the set J. Let Π denote the set of all such permutations. We wish to find such permutation $\pi^* \in \Pi$, that

$$C_{\max}(\pi^*) = \min_{\pi \in \Pi} C_{\max}(\pi),$$

where $C_{\max}(\pi)$ is the time required to complete all jobs on the machines.

3 Flow Shop Scheduling with Fuzzy Processing Times

Let us suppose that processing times of the jobs on machines are not deterministic but they are given by fuzzy numbers.

In this paper the fuzzy processing times $p_{i,j}$ $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$ are represented by a triangular membership function μ (i.e. 3-tuple $\widetilde{p}_{i,j} = (p_{i,j}^{\min}, p_{i,j}^{\text{med}}, p_{i,j}^{\max})$ $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$ with the following properties:

i) $(p_{i,j}^{\min} \leq p_{i,j}^{\text{med}} \leq p_{i,j}^{\max})$,
ii) $\mu(a) = 0$ for $a \leq p_{i,j}^{\min}$ or $a \geq p_{i,j}^{\max}$,
iii) $\mu(p_{i,j}^{\text{med}}) = 1$,
iv) μ is increasing on $[p_{i,j}^{\min}, p_{i,j}^{\text{med}}]$ and decreasing on $[p_{i,j}^{\text{med}}, p_{i,j}^{\max}]$.

The addition of fuzzy numbers $\widetilde{a} = (a_1, a_2, a_3)$ and $\widetilde{b} = (b_1, b_2, b_3)$, can be derived from the extension principle and it is as follows (see [2])

$$\widetilde{a} + \widetilde{b} = (a_1 + b_1, a_2 + b_2, a_3 + b_3).$$

Similarly

$$\max\{\widetilde{a}, \widetilde{b}\} = (\max\{a_1, b_1\}, \max\{a_2, b_2\}, \max\{a_3, b_3\}).$$

If the time of the job execution $\pi(i)$ ($\pi \in \Pi$) on the machines j is determined by a fuzzy number

$$\tilde{p}_{\pi(i),j} = (p^{\min}_{\pi(i),j},\ p^{\text{med}}_{\pi(i),j},\ p^{\max}_{\pi(i),j}),$$

then its finishing time is a fuzzy number in the form of:
$\tilde{C}_{\pi(i),j} = \left(C^{\min}_{\pi(i),j}, C^{\text{med}}_{\pi(i),j}, C^{\max}_{\pi(i),j}\right)$, where $C^{\min}_{\pi(i),j}$, $C^{\text{med}}_{\pi(i),j}$ and $C^{\max}_{\pi(i),j}$ can be determined from the following recurrent formulas:

$$C^{\delta}_{\pi(i),j} = \max\{C^{\delta}_{\pi(i-1),j},\ C^{\delta}_{\pi(i),j-1}\} + p^{\delta}_{\pi(i),j}, \quad \delta \in \{\min, \text{med}, \max\},$$

with the initial conditions

$$C^{\delta}_{\pi(0),j} = 0 \ \ j = 1, 2, \ldots, m, \ \ C^{\delta}_{\pi(i),0} = 0 \ \ i = 1, 2, \ldots, n, \quad \delta \in \{\min, \text{med}, \max\}$$

The time of all jobs execution (in the π sequence) is also a fuzzy number

$$\tilde{C}_{\max}(\pi) = (C^{\min}_{\pi(n),m},\ C^{\text{med}}_{\pi(n),m},\ C^{\max}_{\pi(n),m}).$$

The ranking function defined as follows is to compare fuzzy cost function values:

$$\Im(\tilde{C}_{\max}(\pi)) = \frac{1}{4}\left(C^{\min}_{\pi(n),m} + C^{\text{med}}_{\pi(n),m} + C^{\text{med}}_{\pi(n),m} + C^{\max}_{\pi(n),m}\right) \tag{1}$$

The permutation flow shop scheduling problem with fuzzy processing times (FPFS) consists in determining a permutation $\pi^* \in \Pi$ such that

$$\Im(\tilde{C}_{\max}(\pi)) = \min\{\Im(\tilde{C}_{\max}(\beta)) : \ \beta \in \Pi\},$$

which fulfills constrains (a)–(d).

4 Parallel Hybrid Algorithm

In this section simulated annealing (SA) with the genetic enhancement algorithm (HSG)is used for the permutation flow shop problem with $C_{\max}$. The classic SA algorithm and all modifications leading to HSG are described below. We shall present methods of algorithms parallelization as well as its modifications for the flow shop problem in which execution times are fuzzy numbers.

Classic simulated annealing algorithm
In classic simulated annealing in each iteration a new solution is generated and evaluated. If it is better than the original solution it is accepted, and if it is worse then it is accepted with probability equals to $\exp(-\delta/T)$, where δ is the difference between values of the original and the new solution, and T is a control parameter corresponding to temperature in annealing process in metallurgy. The SA algorithm general scheme is presented on the listing:

```
T := start temperature current := generate initial solution
evaluate current
best:=current
repeat
    count:=0
    repeat
       candidate:=generate candidate from current
       evaluate candidate
       if candidate is better than best then update best
       accept candidate as current with probability
          equal to min(1, exp((F(current) - F(candidate))/T)
       count++
    until (count == number of iterations at temperature T)
    decrease temperature according to cooling scheme
until (stopping criteria)
```

To generate new candidate solutions both transposition and insertion moves are applied with equal probability. In HSG basic geometric cooling scheme is used, so the temperature decreases according to the formula $T_{new} = \alpha \times T_{old}$.

Clustering algorithm for simulated annealing
In their paper, Ram et al. [10] propose a parallel Clustering algorithm for simulated annealing (CASA). In this algorithm master and worker nodes are distinguished. CASA is divided into generations in which the master node distributes the initial solution (or solutions) along worker nodes so they can start running simulated annealing independently. Then, each fixed number of iterations of SA, best solutions found by worker nodes are gathered by the master node. Next the generation starts with the best solution found so far as the initial solution. This model of parallelism is adopted into the HSG algorithm.

Simulated annealing with genetic enhancement
As it has been mentioned before, a mechanism to dynamically adjusted SA's configuration during the runtime is introduced in order to reduce its influence on performance. The SA configuration includes start temperature, minimal temperature and cooling ratio. Number of SA iterations between consecutive temperature reductions is computed in such a way that at the end of the generation temperature is equal to the minimal temperature.

Algorithm description, configuration and complexity
The master node is responsible for generating initial solution. Then at beginning of each *generation*, the master node broadcasts the best solution found so far if necessary (i.e. in first generation or when new best solution was found in a previous generation). After that, worker nodes start SA algorithm for a fixed number of iterations and master node waits until they finish. Finally, the master node gathers values of the best solutions found by worker nodes and selects the best among them. If this best value is better than the best value stored by the

master node, respective permutation is received from a worker node which had found it, and this solution is considered as the best in the next generation. Below listing contains pseudo-code for the the master node in the HSG algorithm.

```
best := generate initial solution
for i := 1 to number of generations do
    broadcast best to worker nodes (if necessary)
    gather values of the best solutions found by worker nodes
    select best solution and receive permutation
    from a worker node that found it (if necessary)
    update best
end.
```

At each worker node a fixed number of generations is performed. Each of these generations starts with receiving the best solution from the master node (if necessary). This solution is then used as an initial solution to execute SA algorithm for a fixed number of iterations. Each worker node starts with different SA configuration (referred to as *individual*, in analogy to genetic algorithms) which is generated randomly with uniform distribution from following ranges: $[1, 200]$ for start temperature, $[0.1, 1]$ for minimal temperature and $[0.9, 1)$ for cooling ratio. After SA algorithm is finished, value of the best solution is returned to the master node. If this value is the best among other worker nodes, and better than the best value stored by the master node, respective permutation is sent back to the master node.

Each individual has its TTL (time to live) with initial value which is HSG's parameter. If a solution returned by SA algorithm at the end of generation is not better than initial one, TTL of respective worker's individual is reduced by one. Otherwise, TTL is reset to the initial value. If individual survives (i.e. its TTL is positive), limit for minimal temperature is lifted and worker continues next generation with a temperature, which it has finished last generation with. If TTL of any individual reaches zero, it is replaced with a new one, generated randomly. Listing contains pseudo-code for each worker node in the HSG algorithm and Figure 1 presents co-operation between nodes in the HSG algorithm.

```
Generate inital individual (SA configuration)
   for i:=1 to number of generations do
      receive best solution from the master node (if necessary)
      execute SA algorithm for a fixed number of iterations
      send back the best value found to the master node
      send back the best solution's permutation
         to the master node (if necessary)
      update individual's TTL and replace it if TTL reaches zero
   end.
```

The HSG's configuration includes number of worker nodes, number of generations and number of SA iterations to be performed in each generation and initial TTL for each individual.

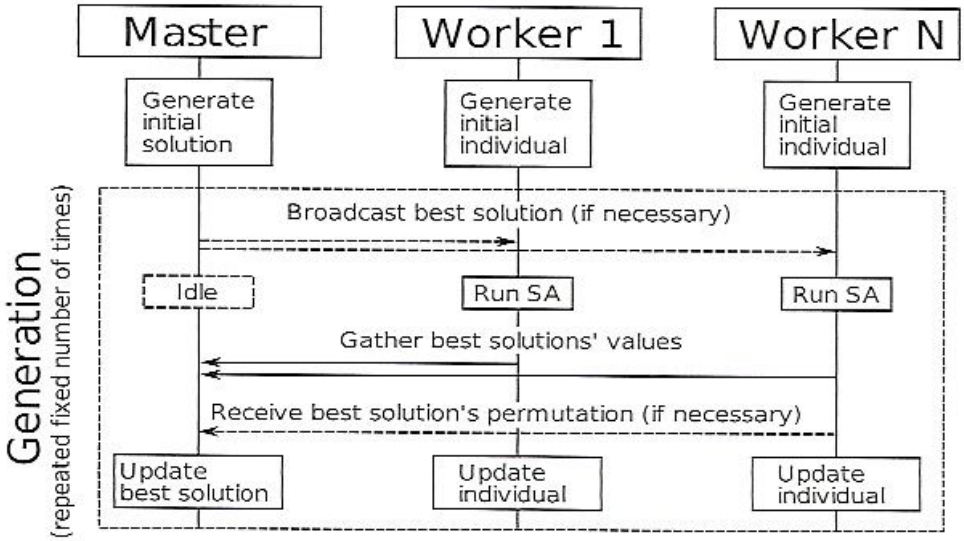

Fig. 1. Co-operation between nodes in HSG algorithm

5 Hybrid Algorithm with Fuzzy Processing Times

In the Section 3 the method of cost function value calculation for the permutation flow shop problem with fuzzy jobs execution times was described. The adequately modified HSG algorithm (in which fuzzy jobs execution times was included) will be represented as FzHSG.

5.1 Algorithms Stability

Let $\mathbf{p} = [p_{i,j}]_{nxm}$ be (deterministic) jobs execution times for an instance of the PFS problem. By $\mathcal{D}(\mathbf{p})$ we describe a set of examlpes of data generated from $\mathbf{p}$ by the disturbance of jobs execution times (i.e. elements from $\mathbf{p}$). The disturbance consists in random changes of $p_{i,j}$ values, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$. This procedure is detailed described in the Section 5.2. We use the following notion:

$\mathcal{A}$ - an algorithm of solving PFS problem,
$\mathbf{p} = [p_{i,j}]_{nxm}$ - an instance of data (execution times) for the PFS problem,
$\pi_{\mathbf{p}}^{\mathcal{A}}$ - a solution determined by the algorithm $\mathcal{A}$ for the data $\mathbf{p}$,
$C_{\max}(\pi_{\mathbf{p}}^{\mathcal{A}}, \mathbf{d})$ - a value of the cost function for the data $\mathbf{d}$ and a sequence of jobs execution (permutation) $\pi_{\mathbf{p}}^{\mathcal{A}}$.

Let $\mathbf{p}$ be an instance of deterministic data and $\mathcal{D}(\mathbf{p})$ a set of disturbed data. For an algorithm $\mathcal{A}$ and an instance of disturbed data $\mathbf{d}$

$$\delta(\mathcal{A}, \mathbf{p}, \mathbf{d}) = \frac{C_{\max}(\pi_{\mathbf{p}}^{\mathcal{A}}, \mathbf{d}) - C_{\max}(\pi_{\mathbf{d}}^{\mathcal{A}}, \mathbf{d})}{C_{\max}(\pi_{\mathbf{d}}^{\mathcal{A}}, \mathbf{d})} \cdot 100\%. \tag{2}$$

This formula defines a percentage relative deviation of the cost function value for the $\mathbf{d}$ if jobs are executed in the sequence $\pi_{\mathbf{p}}^{\mathcal{A}}$ and $\pi_{\mathbf{d}}^{\mathcal{A}}$. By

$$\Delta(\mathcal{A}, \mathbf{p}, \mathcal{D}(\mathbf{p})) = \frac{1}{|\mathcal{D}(\mathbf{p})|} \sum_{\mathbf{d} \in \mathcal{D}(\mathbf{p})} \delta(\mathcal{A}, \mathbf{p}, \mathbf{d}) \tag{3}$$

we define the ***stability of the best solution*** of an instance $\mathbf{p}$ determined by an algorithm $\mathcal{A}$ on the set of disturbed data $\mathcal{D}(\mathbf{p})$. Permutations $\pi_{\mathbf{p}}^{\mathcal{A}}$ and $\pi_{\mathbf{d}}^{\mathcal{A}}$ are the best solutions determined by the algorithm $\mathcal{A}$ for the data $\mathbf{p}$ and $\mathbf{d} \in \mathcal{D}(\mathbf{p})$, respectively.

Let Ω be a set of some (deterministic) data instances for the PFS problem. The ***algorithm stability*** $\mathcal{A}$ on the data set Ω

$$\mathbf{S}(\mathcal{A}, \Omega) = \frac{1}{|\Omega|} \sum_{\mathbf{p} \in \Omega} \Delta(\mathcal{A}, \mathbf{p}, \mathcal{D}(\mathbf{p}))). \tag{4}$$

5.2 Computational Experiments

The algorithms HSG and FzHSG were coded in C++ using MPICH2 implementation of MPI standard for communication between nodes. All experiments were ran on Cranfield Univerity's Astral cluster, which is equipped with 856 Xeon 3 GHz processors, 2 GB memory for each, and Infiniband network. HSG and FzHSG tested on the first 6 groups of benchmark instances (see OR Library: http://mscmga.ms.ic.uk/info.html). The benchmark set contains 120 particulary hard instances of 12 size. For each size (group, ta001-ta060) $n \times m$: 20×5, 20×10, 20×20, 50×5, 50×10, 50×20.

Fuzzy jobs execution times generation

If $p_{i,j}$ $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$ is an instance of deterministic data for the PFS problem, then fuzzy jobs execution times $\widetilde{p}_{i,j}$ are represented by a triple $(p_{i,j}^{\min}, p_{i,j}^{\text{med}}, p_{i,j}^{\max})$, where

$$p_{i,j}^{\min} = \max\{1, \lceil p_{i,j} - p_{i,j}/3 \rceil\},\ p_{i,j}^{\text{med}} = p_{i,j},\ \text{and}\ p_{i,j}^{\max} = \lceil p_{i,j} + p_{i,j}/6 \rceil.$$

Disturbed data generation

For each instance of deterministic data $\mathbf{p} = \{p_{i,j}\}_{m \times n}$ there were 100 instances generated - elements of the set $\mathcal{D}(\mathbf{p})$. If an instance of the data $\mathbf{d} \in \mathcal{D}(\mathbf{p})$ $(\mathbf{d} = \{d_{i,j}\}_{m \times n})$ than jobs execution times were drawn (due to the uniform distribution) from the range $[\max\{1, \lceil p_{i,j} - p_{i,j}/3 \rceil\}, \lceil p_{i,j} + p_{i,j}/6 \rceil]$.

There were 6 000 instances generated of disturbed data in total. For the each group of instances the values parameters (2) and (3) were calculated and they are shown in the Table 1. Values $\delta_{\min}$ i $\delta_{\max}$ are values of the minimal and maximal deviation defined by (2), respectively, for the group of instances, and δ_{aprd} – an average value. The stability of both algorithm was also calculated on the set Ω including groups of instances shown in the Table 1. For the HSG algorithm, $\mathbf{S}(\text{HSG}, \Omega) = 2.02\%$, and for the algorithm with fuzzy jobs execution times FzHSG, $\mathbf{S}(\text{FzHSG}, \Omega) = 2.00\%$. Therefore, the stability of both algorithms is almost identical. It follows among others from this that fuzzy jobs finishing times (after defuzzification) were insignificantly different from the times calculated for the input deterministic times. The disturbance process of the jobs execution times makes the times of jobs shorter or longer, but it has small influence on the time of jobs finishing. Therefore, values of the cost function are almost identical. The application of parallelism makes possible to execute all the calculations in about 1 hour.

Group	*Parallel hybrid algorithm* HSG			*Fuzzy parallel hybrid algorithm* FzHSG		
	$\delta_{\min}$	δ_{aprd}	$\delta_{\max}$	$\delta_{\min}$	δ_{aprd}	$\delta_{\max}$
20×5	0.022	1.782	5.487	0.022	1.892	5.471
20×10	0.157	2.466	6.013	0.102	2.326	5.964
20×20	0.270	2.246	5.263	0.389	2.275	5.254
50×5	0.044	1.238	4.088	0.030	1.240	4.150
50×10	0.222	2.172	4.577	0.255	2.196	5.122
50×20	0.528	2.198	4.155	0.541	2.178	4.219

6 Conclusions

We present a parallel simulated annealing algorithm with genetic enhancement for a permutation flow shop problem with fuzzy processing times. The algorithm introduces a dynamic parameters adjustment for simulated annealing algorithm. The parallel algorithm stability was also defined and researched for the deterministic and fuzzy jobs execution times.

References

1. Bożejko, W., Wodecki, M.: Solving the Flow Shop Problem by Parallel Simulated Annealing. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2001. LNCS, vol. 2328, pp. 236–244. Springer, Heidelberg (2002)
2. Dubois, D., Prade, H.: Theorie des Possibilites. In: Applications a la representation des connaissances en informatique. MASSON, Paris (1988)
3. Dumitru, V., Luban, F.: Membership functions, some mathematical programming models and production scheduling. Fuzzy Sets and Systems 8, 19–33 (1982)
4. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Computers & Operations Research 31, 1891–1909 (2004)
5. Izzettin, T., Serpil, E.: Fuzzy branch-and-bound algorithm for flow shop scheduling. Journal of Intelligent Manufacturing 15, 449–454 (2004)
6. Ishibuschi, H., Murata, T.: Scheduling with Fuzzy Duedate and Fuzzy Processing Time. In: Słowiński, R., Hapke, M. (eds.) Scheduling Under Fuzziness, pp. 113–143. Springer, Heidelberg (2000)
7. Nowicki, E., Smutnicki, C.: A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research 91, 160–175 (1996)
8. Osman, I., Potts, C.: Simulated Annealing for Permutation Flow-Shop Scheduling. OMEGA 17(6), 551–557 (1989)
9. Peng, J., Liu, B.: Parallel machine scheduling models with fuzzy processing times. Information Sciences 166, 49–66 (2004)
10. Ram, J.D., Sreenivas, T.H., Subramaniam, G.K.: Parallel simulated annealing algorithms. Journal of Parallel and Distributed Computing 37(2), 207–212 (1996)
11. Reeves, C.: A Genetic Algorithm for Flowshop Sequencing. Computers & Operations Research 22(1), 5–13 (1995)
12. Tsujimura, Y., Park, S.H., Change, I.S., Gen, M.: An effective method for solving flow shop problems with fuzzy processing times. Computers and Industrial Engineering 25(1-4), 239–242 (1993)