Parallel Neuro-Tabu Search Algorithm for the Job Shop Scheduling Problem

Wojciech Bożejko[1], Mariusz Uchroński[2], and Mieczysław Wodecki[3]

[1] Institute of Computer Engineering, Control and Robotics
Wrocław University of Technology
Janiszewskiego 11-17, 50-372 Wrocław, Poland
wojciech.bozejko@pwr.wroc.pl
[2] Wrocław Centre of Networking and Supercomputing
Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
mariusz.uchronski@pwr.wroc.pl
[3] Institute of Computer Science, University of Wrocław
Joliot-Curie 15, 50-383 Wrocław, Poland
mwd@ii.uni.wroc.pl

Abstract. We propose two parallel algorithms based on neuro-tabu search method, designed to solve the jobs shop problem of scheduling. The fist algorithm is based on independent runs of the neuro-tabu with different starting points. The second one uses sophisticated diversification method based on path-relinking methodology applied to the set of elite solutions. Proposed approaches are especially effective for the instances of large size.

1 Introduction

We consider the job shop scheduling problem here, which can be described as follows (see [4]). There is a set of jobs and a set of machines. Each job consists of a number of operations which have to be processed in a given order, each one on a specified machine during a fixed time. The processing of an operation cannot be interrupted. Each machine can process at most one operation at a time. We want to find a schedule (the assignment of operations to time intervals on machines) that minimizes the *makespan*. The job shop scheduling problem, although relatively easily stated, is strongly NP-hard and it is considered as one of the hardest problems in the area of combinatorial optimization.

Because of NP-hardness of the problem heuristics and metaheuristics are recommended as 'the most reasonable' solution methods. The majority of these methods refer to the makespan minimization. We mention here a few recent studies: Jain, Rangaswamy, and Meeran [21]; Pezzella and Merelli [30]; Grabowski and Wodecki [17]; Nowicki and Smutnicki [27]; Bożejko and Uchroński [6]. Heuristics algorithms based on dispatching rules are also proposed in papers of Holthaus and Rajendran [19], Bushee and Svestka [10] for the problem under consideration. For the other regular criteria such as the total tardiness there are proposed

L. Rutkowski et al. (Eds.): ICAISC 2013, Part II, LNAI 7895, pp. 489–499, 2013.

metaheuristics based on various local search techniques: simulated annealing [34], tabu search [2] and genetic search [26].

Here we propose the new approach to the distributed tabu search metaheuristic designing to solve difficult discrete optimization problems, such as the job shop problem, using a multi-GPU cluster with distributed memory. We also determine theoretical number of processors, for which the speedup measure has a maximum value. We experimentally determine what parallel execution time T_p can be obtained in real-world installations of multi-GPU clusters (nVidia Tesla S2050 with a 6-cores CPU server) for Taillard benchmarks [32] of the job shop scheduling problem, and compare them with theoretically determined values.

2 Job Shop Problem

Job shop scheduling problems result from many real-world cases, which means that they have good practical applications as well as industrial significance. Let us consider a set of jobs $\mathcal{J} = \{1, 2, \ldots, n\}$, a set of machines $M = \{1, 2, \ldots, m\}$ and a set of operations $\mathcal{O} = \{1, 2, \ldots, o\}$. The set $\mathcal{O}$ is decomposed into subsets connected with jobs. A job j consists of a sequence of o_j operations indexed consecutively by $(l_{j-1}+1, l_{j-1}+2, \ldots, l_j)$, which have to be executed in this order, where $l_j = \sum_{i=1}^{j} o_i$ is the total number of operations of the first j jobs, $j = 1, 2, \ldots, n$, $l_0 = 0$, $\sum_{i=1}^{n} o_i = o$. An operation i has to be executed on machine $v_i \in M$ without any idleness in time $p_i > 0$, $i \in \mathcal{O}$. Each machine can execute at most one operation at a time. A feasible solution constitutes a vector of times of the operation execution beginning $S = (S_1, S_2, \ldots, S_o)$ such that the following constraints are fulfilled

$$S_{l_{j-1}+1} \geq 0, \quad j = 1, 2, \ldots, n, \tag{1}$$

$$S_i + p_i \leq S_{i+1}, \quad i = l_{j-1} + 1, \; l_{j-1} + 2, \ldots, l_j - 1, \; j = 1, 2, \ldots, n, \tag{2}$$

$$S_i + p_i \leq S_j \quad \text{or} \quad S_j + p_j \leq S_i, \quad i, j \in O, \quad v_i = v_j, \quad i \neq j. \tag{3}$$

Certainly, $C_j = S_j + p_j$. An appropriate criterion function has to be added to the above constraints. The most frequent are the following two criteria: minimization of the time of finishing all the jobs and minimization of the sum of job finishing times. From the formulation of the problem we obtain $\mathcal{C}_j \equiv C_{l_j}$, $j \in \mathcal{J}$.

The first criterion, the time of finishing all the jobs

$$C_{\max}(S) = \max_{1 \leq j \leq n} C_{l_j}. \tag{4}$$

corresponds to the problem denoted as $J||C_{\max}$ in the literature. The second criterion, the sum of job finishing times

$$C(S) = \sum_{j=1}^{n} C_{l_j}. \tag{5}$$

corresponds to the problem denoted as $J||\sum C_i$ in the literature.

Both problems described are strongly NP-hard and although they are similarly modelled, the second one is found to be harder because of the lack of some specific properties (so-called block properties, see [27]) used in optimization of execution time of solution algorithms.

3 Tabu Search Mechanism with Neural Network Application

In the considered neuro-tabu search algorithm NTS each move is represented by its neuron. For the neighborhood considered in [28] a network of neurons formed of $o-1$ neurons. Let i-th neuron represents a move consisting in swap of two adjacent elements on the positions i and $i+1$ in a solution π.

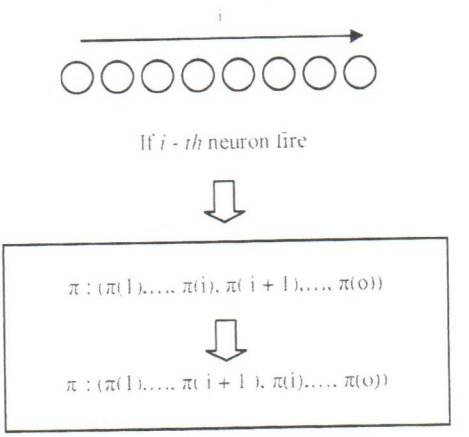

Fig. 1. Dependencies between activation of the neuron and a move

In a proposed neural network architecture a history of each neuron is stored as its internal state (*tabu effect*). If in an iteration neuron is activated, then the value 1 is fixed on its output and values 0 are fixed on the outputs of other neurons. The neuron activated in an iteration must not be activated once again for the next s iterations. Each neuron is defined by the following equations:

$$\eta_i(t+1) = \alpha \Delta_i(t), \tag{6}$$

$$\Delta_i(t) = \frac{C_{max}(\pi_v^{(t)}) - C^*_{max}}{C^*_{max}}, \tag{7}$$

$$\gamma_i(t+1) = \sum_{d=0}^{s-1} k^d x_i(t-d), \tag{8}$$

where $x_i(t)$ is an output of the neuron i in the iteration t. Symbol $C_{max}(\pi_v^{(t)})$ means the value of the goal function for the permutation obtained after executing

a move v in the iteration t, i.e. $\pi^{(t)}$. Symbol $\Delta_i(t)$ means a normalized, current value of the goal function, and C^*_{max} is the value of the best solution found so far. Parameters α i k are scale factors. A symbol $\eta_i(t+1)$ (*gain effect*) defines quality of a move v. A variable $\gamma_i(t+1)$ (*tabu effect*) stores a history of the neuron i for the last s iterations. Neuron is activated if it has a low value of the tabu effect and it gives a better reduction of the C_{max}. More detailed a neuron i is activated if it has the lowest $\{\eta_i(t+1) + \gamma_i(t+1)\}$ value of all the neurons.

If $0 < k < 1$ i $s = t$ then an equation(8) takes the form of:

$$\gamma_i(t+1) = k\gamma_i(t) + x_i(t). \tag{9}$$

where $\gamma_i(0) = 0$ and $x_i(0) = 0$ for each i. From the equation (9) it follows, that the value of $\gamma_i(t)$ of each neuron decreases exponentially (Fig. 3 and 4).

In many algorithms proposed in the literature which are based on the tabu search method a so-called aspiration criterion is implemented. It consists in executing of the forbidden move if it follows to the base solution with the goal function value lower than the best found so far.

In the proposed neuro-tabu search such a function can be implemented by ignoring tabu effect for a move v for which $\Delta_i < 0$. However during computational experiments we have observed that it does not give good effects. Therefore a proposed neuro-tabu search has not got such a function. The skeleton of the neuro-tabu search algorithm is given on the Fig. 2.

4 Parallel Neuro-Tabu Search Algorithm *pNTS*

Here we propose a solution method to the job shop problem in the distributed computing environments, such as multi-GPU clusters. Tabu search algorithm is executed in concurrent working threads, as in *multiple-walk* model of parallelization [1] (MPDS,*Multiple starting Point Different Strategies* in the Voß [33] classification of parallel tabu search metaheuristic). Additionally, MPI library is used to distribute calculations among GPUs (see Fig. 4).

Now let us consider a single cycle of the MPI data broadcasting, multi-GPU computations and batching up of the results obtained. Let us assume that the single communication procedure between two nodes of a cluster takes the time T_{comm}, the time of sequential tabu search computations is T_{seq} and the computations time of parallel tabu search is $T_{calc} = \frac{T_{seq}}{p}$ (p is the number of GPUs). Therefore, the total parallel computations time of the single cycle is

$$T_p = 2T_{comm}\log_2 p + T_{calc} = 2T_{comm}\log_2 p + \frac{T_{seq}}{p}.$$

In case of using more processors, the parallel computing time $\left(\frac{T_{seq}}{p}\right)$ decreases, whereas the time of communication ($2T_{comm}\log p$) increases. We are looking for such a number of processors p (let us call it p^*) for which T_p is minimal. By calculating $\frac{\partial T_p}{\partial p} = 0$ we obtain

$$\frac{2T_{comm}}{p\ln 2} - \frac{T_{seq}}{p^2} = 0 \tag{10}$$

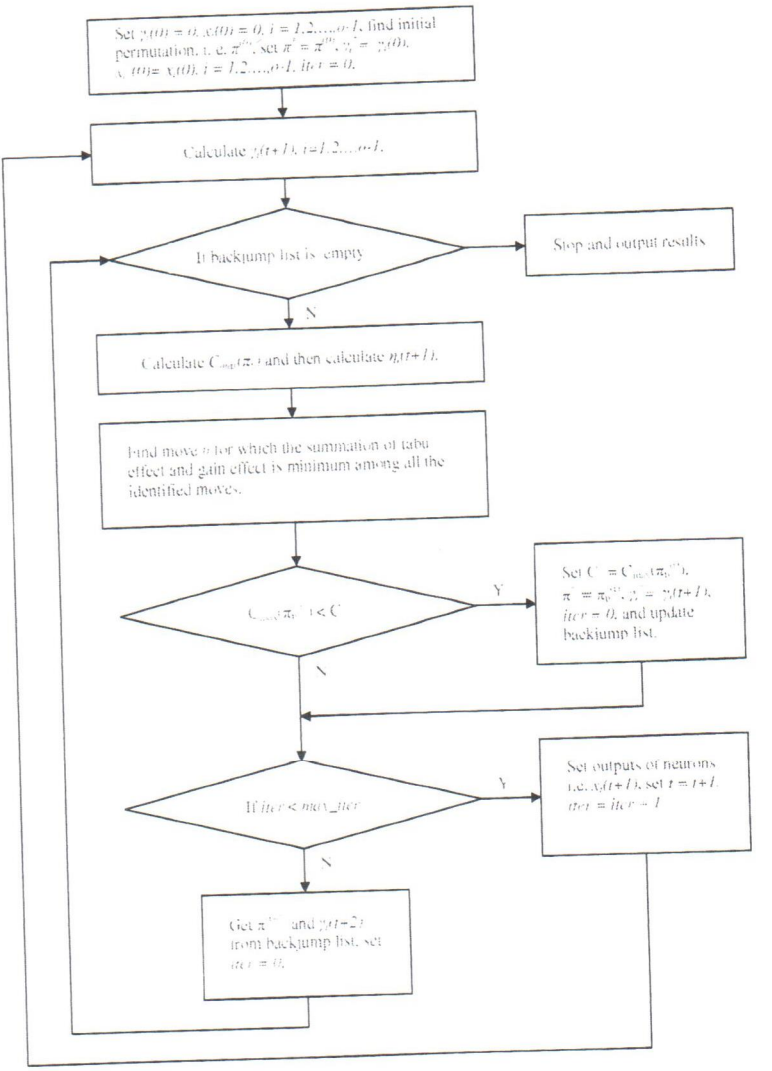

Fig. 2. A skeleton of the neuro-tabu search algorithm

and then

$$p = p^* = \frac{T_{seq} \ln 2}{2T_{comm}}, \tag{11}$$

which provides us with an optimal number of processors p^* which minimizes the value of the parallel running time T_p.

The fraction of communication time is $O(\log_2 p)$ in this tree-based data broadcasting method, therefore this is another situation than for linear-time broadcasting (discussed in [3]), for which the overall communication and calculation

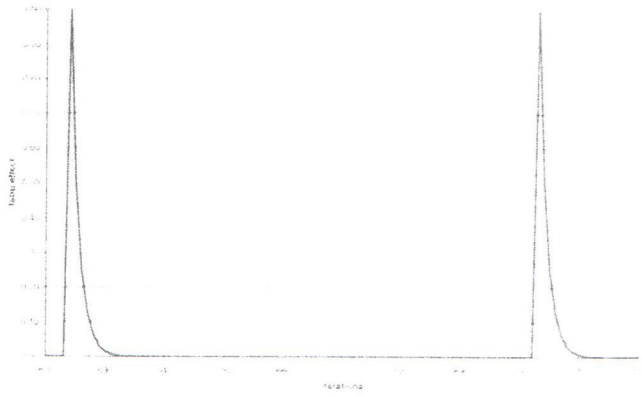

Fig. 3. Changes of the $\gamma(t)$ value for $k = 0.5$

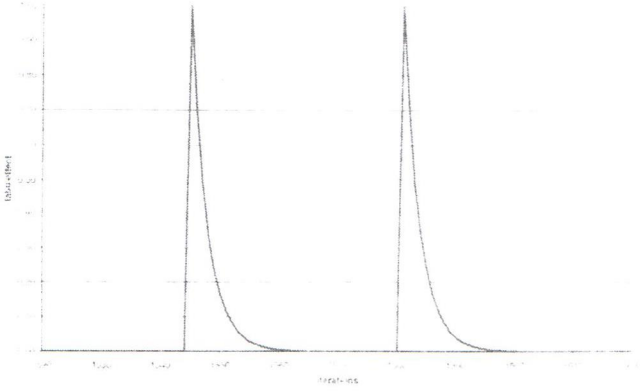

Fig. 4. Changes of the $\gamma(t)$ value for $k = 0.7$

efficiency is much lower. On the other hand the linear-time broadcasting is similar to described by Brooks' Law [9] for project management, i.e. the expected advantage from splitting development work among n programmers is $O(n)$ but the communications cost associated with coordinating and then merging their work is $O(n^2)$.

5 Advanced Neuro-Tabu Search Algorithm *iNTS*

As the second solution method we propose an approach introduced by Nowicki and Smutnicki [27] with using neuro-tabu instead of classic tabu search algorithm, as in the original paper. The proposed $iNTS$ algorithm operates on the set of dispersed (elite) solution obtained with using $NIS(\gamma, \delta, C^R)$ function (see Fig. 6), where γ, δ are two processing orders and C^R is the reference makespan (goal function value). Inside, the neighborhood generation function is used based

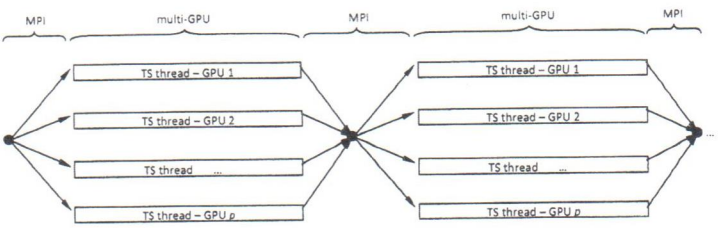

Fig. 5. Skeleton of the Multi-Level Tabu Search metaheuristic

on swap moves of adjacent operations (see Bożejko [4]). We denote by $N(\pi)$ the set of moves obtained from a solution π by swapping any two adjacent operations lying on critical path (to obtain feasible solution, see [17]). The fundamental aim of the NIS function is to provide the solution $\varphi = NIS(\gamma, \delta, C^R)$ located 'between' γ and δ reference solutions to run a neuro-tabu search explaration in the main $iNTS$ algorithm (see Fig. 7). To control the distance between any two solution α,β, the Kendall's tau measure $D(\alpha, \beta)$ is used to represent the minimal number of adjacent swap moves (inversions) to obtain permutation β^{-1} from α^{-1} (for more measures on permutations see e.g. Diaconis [13]). Values of tuning parameters were following: $maxE = 8$ (number of elite solutions), $maxD = 5$ (maximal distance used in the loop), $maxV = 0.5$.

Algorithm 1. $NIS(\gamma, \delta, C^R)$
Input: γ, δ – two processing orders; C^R – reference makespan;
Output: φ – processing order; update reference makespan C^R;
  $\pi \leftarrow \gamma$; $iter \leftarrow 0$. **Find** δ^{-1} **and** $D(\gamma, \delta)$
  repeat
    $iter \leftarrow iter + 1$; **Find** $N(\pi)$;
    For any $v \in N(\pi)$ **calculate and store** $C_{max}(\pi_{(v)})$;
    Find $N^+ = \{v = (x, y) \in N(\pi) : \delta^{-1}(y) < \delta^{-1}(x)\}$;
      if $N^+ \neq \emptyset$ **than** $K \leftarrow N^+$ **else** $K \leftarrow N(\pi)$;
      Select the move $w \in K$ **such, that**
        $C_{max}(\pi_{(w)}) = \min_{v \in K} C_{max}(\pi_{(v)})$;
      Denote $\pi_{(w)}$ **by** α;
    $\pi \leftarrow \alpha$; $\varphi \leftarrow \pi$;
    if $C_{max}(\pi) < C^R$ **than** $C^R \leftarrow C_{max}(\pi)$ **and exit**;
  until $iter \geq maxV \cdot D(\gamma, \delta)$; {$maxV \in (0, 1)$ - **parameter**}

Fig. 6. $NIS(\gamma, \delta, C^R)$ function

6 Computational Experiments

Proposed algorithms were ran on the server based on 6-cores Intel Core i7 CPU X980 (3.33GHz) processor equipped with nVidia Tesla S2050 GPU (1792 cores) working under 64-bit Linux Ubuntu 10.04.4 LTS operating system and tested on the benchmark problems taken from Taillard [32].

Algorithm 2. $iNTS$
Input: π^0 – processing orders provided by INSA;
Output: π^* – the best found processing order
and its makespan C^*;
Set $(\pi^1, C^1) \leftarrow NTS(\pi^0)$ **and** $C^* \leftarrow C^1$;
for $i \leftarrow 2, \ldots, maxE$ **do**
$\varphi \leftarrow NIS(\pi^{i-1}, \pi^0, C^*)$; $(\pi^i, C^i) \leftarrow NTS(\varphi)$;
$C^* = \min\{C^*, C^i\}$;
repeat
Find $1 \leq l \leq maxE$ **so that**
$D(\pi^k, \pi^l) = \max\{D(\pi^k, \pi^i) : 1 \leq i \leq maxE\}$;
Set $\varphi \leftarrow NIS(\pi^k, \pi^l, C^*)$ **and** $(\pi^l, C^l) \leftarrow NTS(\varphi)$;
if $C^l < C^k$ **than set** $(\pi^*, C^*) \leftarrow (\pi^l, C^l)$ **and** $k \leftarrow l$;
until $\max\{D(\pi^k, \pi^i) : 1 \leq i \leq maxE\} < maxD$.

Fig. 7. Algorithm $iNTS$

Computational experiments results are compared in the Table 1. Particular columns have the following notion:

- $sNTS$ – sequential Neuro Tabu Search algorithm of Bożejko and Uchroński [6],
- $pNTS$ – parallel (for $p = 16$) Neuro Tabu Search algorithm. MPSS model (due to the Voß [33] classification) without communication (proposed in the Section 4); for each processor the starting solution was generated by the NTS algorithm executed for $process\_id * 100$ iterations.
- $iNTS$ – advanced NTS algorithm based on the diversification and intensification methodology proposed in the Section 5.

Processing times for all benchmark instances were: (sequential) NTS – 132m27.319s, (parallel) $pNTS$ ($p = 4$) – 169m45.748s (longer time of parallel algorithm work causes of the method of starting solutions generation and synchronization), $iNTS$ – about 60 hours. As we can observe the proposed $iNTS$ algorithm managed to obtain the average relative percentage deviation from the

Table 1. Percentage relative deviations (PRD) to the best known solutions

problem	$n \times m$	$sNTS$	$pNTS(p = 16)$	$iNTS$
TA01-10	15 × 15	0.4948	0.3141	0.0652
TA11-20	20 × 15	1.1691	0.9129	0.4412
TA21-30	20 × 20	1.2486	0.7033	0.4803
TA31-40	30 × 15	1.0592	0.7965	0.3764
TA41-50	30 × 20	1.8565	1.5634	0.8328
TA51-60	50 × 15	0.0915	0.0915	0.0520
TA61-70	50 × 20	0.1479	0.0210	0.0140
TA71-80	100 × 20	0.0090	0.0090	0.0090
average		**0.7596**	**0.5515**	**0.2839**

best known solution of the Taillard instances on the level of 0.28%. Average PRD values for test instances of the group TA61-TA70 (1,000 operations) are 10 times lower than for the proposed $iNTS$ algorithm. Comparing results for $sNTS$ and $pNTS$ algorithm we can notice that using parallel algorithm results in obtaining much lower PRD values. Based on this we plan to implement parallel version of $iNTS$ algorithm as a future work.

7 Conclusions

In this paper we propose an approach designed to solve difficult problems of combinatorial optimization in distributed parallel architectures without shared memory, such as clusters of nodes equipped with GPU units (i.e. multi-GPU clusters). The methodology can be especially effective for large instances of hard to solve optimization problems, such as flexible scheduling problems as well as discrete routing and assignment problems.

References

1. Alba, E.: Parallel Metaheuristics. A New Class of Algorithms. Wiley & Sons Inc (2005)
2. Armentano, V.A., Scrich, C.R.: Tabu search for minimizing total tardiness in a job shop. International Journal of Production Economics 63(2), 131–140 (2000)
3. Bożejko, W.: A new class of parallel scheduling algorithms, pp. 1–280. Wroclaw University of Technology Publishing House (2010)
4. Bożejko, W.: On single-walk parallelization of the job shop problem solving algorithms. Computers & Operations Research 39, 2258–2264 (2012)
5. Bożejko, W., Uchroński: Distributed Tabu Search Algorithm for the Job Shop Problem. In: Proceedings of the 14th International Asia Pacific Conference on Computer Aided System Theory 6th to 8th Sydney, Sydney, Australia, February 6-8 (2012)
6. Bożejko, W., Uchroński, M.: A Neuro-Tabu Search Algorithm for the Job Shop Problem. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part II. LNCS, vol. 6114, pp. 387–394. Springer, Heidelberg (2010)
7. Bożejko, W., Uchroński, M., Wodecki, M.: Parallel Meta^{2}heuristics for the Flexible Job Shop Problem. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part II. LNCS, vol. 6114, pp. 395–402. Springer, Heidelberg (2010)
8. Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 41, 157–183 (1993)
9. Brooks Jr., F.P.: The Mythical Man-Month, anniversary edition. Addison-Wesley, Reading (1995)
10. Bushee, D.C., Svestka, J.A.: A bi-directional scheduling approach for job shops. International Journal of Production Research 37(16), 3823–3837 (1999)
11. Crainic, T.G., Toulouse, M., Gendreau, M.: Parallel asynchronous tabu search in multicommodity locationallocation with balancing requirements. Annals of Operations Research 63, 277–299 (1995)

12. Dauzère-Pérès, S., Pauli, J.: An integrated approach for modeling and solving the general multiprocessor job shop scheduling problem using tabu search. Annals of Operations Research 70(3), 281–306 (1997)
13. Diaconis, P.: Group Representations in Probability and Statistics. Lecture Notes - Mono-graph Series, vol. 11. Institute of Mathematical Statistics, Harvard University (1988)
14. Flynn, M.J.: Very highspeed computing systems. Proceedings of the IEEE 54, 1901–1909 (1966)
15. Gao, J., Sun, L., Gen, M.: A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operations Research 35, 2892–2907 (2008)
16. Grabowski, J.: Generalized problems of operations sequencing in the discrete production systems, Monographs 9, Scientific Papers of the Institute of Technical Cybernetics of Wrocław Technical University (1979)
17. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm for the job shop problem. In: Rego, C., Alidaee, B. (eds.) Adaptive Memory and Evolution, Tabu Search and Scatter Search, Kluwer Academic Publishers, Dordrecht (2005)
18. Hanafi, S.: On the Convergence of Tabu Search. Journal of Heuristics 7, 47–58 (2000)
19. Holthaus, O., Rajendran, C.: Efficient jobshop dispatching rules: further developments. Production Planning and Control 11, 171–178 (2000)
20. Hurink, E., Jurisch, B., Thole, M.: Tabu search for the job shop scheduling problem with Multi-purpose machine, Oper. Res. Spektrum 15, 205–215 (1994)
21. Jain, A.S., Rangaswamy, B., Meeran, S.: New and stronger job-shop neighborhoods: A focus on the method of Nowicki and Smutnicki (1996). Journal of Heuristics 6(4), 457–480 (2000)
22. Jia, H.Z., Nee, A.Y.C., Fuh, J.Y.H., Zhang, Y.F.: A modified genetic algorithm for distributed scheduling problems. International Journal of Intelligent Manufacturing 14, 351–362 (2003)
23. Kacem, I., Hammadi, S., Borne, P.: Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C 32(1), 1–13 (2002)
24. Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the Flexible Job-schop Scheduling Problem. Computers & Operations Research 35, 3202–3212 (2008)
25. Mastrolilli, M., Gambardella, L.M.: Effective neighborhood functions for the flexible job shop problem. Journal of Scheduling 3(1), 3–20 (2000)
26. Mattfeld, D.C., Bierwirth, C.: An efficient genetic algorithm for job shop scheduling with tardiness objectives. European Journal of Operational Research 155(3), 616–630 (2004)
27. Nowicki, E., Smutnicki, C.: An advanced tabu search algorithm for the job shop problem. Journal of Scheduling 8(2), 145–159 (2005)
28. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop problem. Management Science 42, 797–813 (1996)
29. Pauli, J.: A hierarchical approach for the FMS schduling problem. European Journal of Operational Research 86(1), 32–42 (1995)
30. Pezzella, F., Merelli, E.: A tabu search method guided by shifting bottleneck for the job-shop scheduling problem. European Journal of Operational Research 120, 297–310 (2000)

31. Pinedo, M.: Scheduling: theory, algorithms and systems. Prentice-Hall, Englewood Cliffs (2002)
32. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285 (1993)
33. Voß, S.: Tabu search: Applications and prospects. In: Du, D.Z., Pardalos, P.M. (eds.) Network Optimization Problems, pp. 333–353. World Scientific Publishing Co., Singapore (1993)
34. Wang, T.Y., Wu, K.B.: An eficient configuration generation mechanism to solve job shop scheduling problems by the simulated annealing. International Journal of Systems Science 30(5), 527–532 (1999)