Parallel Genetic Algorithm for Minimizing Total Weighted Completion Time

Wojciech Bożejko[1] and Mieczysław Wodecki[2]

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

Abstract. We have considered the problem of job scheduling on a single machine with deadlines. The objective is to find a feasible job sequence (satisfying the deadlines) to minimize the sum of weighted completion times. Since the problem is NP-hard, heuristics have to be used. Methods of artificial intelligence: simulated annealing, neural networks and genetic algorithms, are some of the recent approaches. We propose a very effective parallel genetic algorithm PGA and methods of determining lower and upper bounds of the objective function. Since there are difficulties with determining the initial population of PGA for this scheduling problem, therefore the algorithm also adds random generated unfeasible solutions to the population. We announce a method of elimination of these kind of solutions. The examined algorithms are implemented in Ada95 and MPI. Results of computational experiments are reported for a set of randomly generated test problems. *

1 Introduction

Nowadays artificial intelligence methods: neural networks, simulated annealing and genetic algorithms, are successfully used to solve the most difficult (strongly *NP*-hard) problems of combinatorial optimization. Implementation of such an algorithm in multiprocessor environment, if cleverly done, considerably accelerates calculations and makes it possible to solve large instances of the problem relatively fast.

Many scheduling problems are an important part of short-term production planning. We present a problem of sequencing set of jobs on a single machine. Each job has to be done before its deadline. The objective is to find a schedule of jobs that minimizes the sum of weighted completion times.

This problem was defined by Smith [8] and is denoted as $1|d_i|\sum w_i C_i$ in literature. Lenstra et al. [4] shows, that it is strongly *NP*-hard which indicates that the existence of a polynomial bounded algorithm is unlikely.

* The work was supported by KBN Poland, within the grant No. T11A01624

An optimal branch and bound algorithm was proposed by Bensal [1], Posner [6] and Yupeng Pan [5]. The algorithm of Posner solve all instances of up to 40 jobs and most instances of 50-60 jobs, see [5]. In the paper [5], Yupeng Pan presents a branch and bound algorithm (based on an idea by Posner) that extensively utilizes the principle of optimality to reduce the size of the search tree. This algorithm can solve instances of up to 90 jobs. In this paper we announce a parallel genetic algorithm based on the island model of migration with a new crossover operator (modified PMX) and deadline flowing technique. Through lack of reference instances, we compare the obtained solutions (for random instances) with calculated upper bound. This work is a continuation of [9].

2 Problem definition

The single machine total weighted completion time problem (SWCT) may be stated as follows. In a given instance, each job i has a due date (deadline) d_i, a processing time p_i and weight w_i. For a given schedule, if a job is completed before its due date, its feasible. The schedule is called feasible if any job in this schedule is feasible. Our goal is to find a feasible schedule that minimizes the total weighted completion time $\sum_{i=1}^{n} w_i C_i$, where C_i is the completion time of job i.

Let $N = \{1, 2, \ \dots \ , n\}$ be a set of n jobs. By Φ_n we mean a set of all permutations of elements of the set N. The permutation $\pi \in \Phi_n$ is feasible, if

$$\forall i \in N, \ \ d_{\pi(i)} \geq C_{\pi(i)} \text{ where } C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)}.$$

The SWCT problem consist in determining optimal element π^* such, that

$$F(\pi^*) = \min\{F(\pi) : \ \ \pi \in \Pi\},$$

where $\Pi \subseteq \Phi_n$ is a set of feasible solutions and the goal function is

$$F(\pi) = \sum_{i=1}^{n} w_{\pi(i)} * C_{\pi(i)}, \quad \pi \in \Pi.$$

For a sequence of jobs $\pi = (\pi(1), \pi(2), \ \dots \ , \pi(n))$, if $d_{\pi(1)} \leq d_{\pi(2)} \leq, \ \dots \ , \leq d_{\pi(n)}$, then such a schedule (permutation) is called Earliest Due Date (EDD).

Property 1. If for a certain instance of SWCT problem, EDD jobs' schedule is feasible, then the set of solutions $\Pi \neq \emptyset$, otherwise $\Pi = \emptyset$.

Using this property we can easily check if an instance has any feasible solution.

3 Genetic algorithm

Genetic algorithms are inspired by natural selection and evolution. Search strategy based on a theme borrowed from genetics was introduced by Holland [3]. Goldberg [2] described this method as a search algorithm for optimization. He used concepts from population genetics and the evolution theory to construct algorithms that try to optimize the fitness of a population of elements through recombination and mutation of their genes. Key components of a genetic algorithm are: a mechanism to generate an initial population, a measure of solution fitness, and genetic operators that combine and alter current solutions to new solution forms.

We have applied a partially mapped crossover (PMX), and cycle crossover (CX) operators in our algorithm. The generated descendant (permutation) is modified in such a way that the distance between it and one of the parents is minimal. The distance between permutations $\alpha, \beta \in \Phi_n$ is denoted as a minimal number of transpositions of adjacent elements that should be performed to obtain β from α.

4 Parallel genetic algorithm

There are three basic types of parallelization strategies which can be applied to the genetic algorithm: global, diffusion model and island (migration) model.

Algorithms based on the island model divide the population into a few subpopulations. Each of them is assigned to a different processor which performs a sequential genetic algorithm based on its own subpopulation. The crossover involves only individuals within the same population. Occasionally, the processor exchanges individuals through a migration operator. The main determinants of this model are: (1) size of the subpopulations, (2) topology of the connection network, (3) number of individuals to be exchanged, (4) frequency of exchanging.

The island model is characterized by a significant reduction of the communication time, compared to previous models. Shared memory is not required, so this model is more flexible too.

Below, a parallel genetic algorithm is proposed. The algorithm is based on the island model of parallelizm. Let P be a population, P' a set of parents and P'' a set of offspring of the random parents from P'.

Algorithm 1. Parallel genetic algorithm

```
parfor j = 1,2,...,p   { p is number of processors }
  i ← 0; P_j ← random subpopulation connected with processor j;
  p_j ← number of individuals in subpopulation j;
  repeat
    Selection(P_j, P'_j);   Crossover(P'_j, P''_j);   Mutation(P''_j);
    if (k mod S = 0) then {Migration}
      r := random(1,p);
      Remove β = 25 percentage of individuals in subpopulation P_j;
      Replenish P_j by the best individuals from subpopulation P_r
      taken from processor r;
    end if;
  until Stop_Condition;
end parfor
```

The frequency of communication between processors (migration) is very important for the parallel algorithm performance. It must not be too frequent because of the long time of communication between processors. In this implementation the processor gets new individuals quite rarely, every $S = 20$ iterations.

5 Lower bound and upper bound

For each instance the lower bound (LB) and upper bound (UB) of the optimal goal function value is calculated.

A Lower Bound from the Assignment Problem.

Let us suppose that we have found that there exists an optimal schedule whereby, for the job $i \in N$, the set of jobs $\Gamma_i^- \subseteq N$ precedes i and the set of jobs $\Gamma_i^+ \subseteq N$ follows i. If in any established relation job i precedes j, than $i \in \Gamma_j^-$ and $j \in \Gamma_i^+$.

Theorem 1. *(Elimination rule). If at least one of the following conditions holds*

$$p_i < p_j,\ w_i \geq w_j,\ d_i \leq P(\Gamma_j^-) + p_j \ \text{ or } w_i \geq w_j,\ d_j \geq d_i,\ d_j \geq P(N \backslash \Gamma_i^+) - p_j.$$

then job i precedes j, $(i, j \in N)$.

Proof. The proof is based on the more general results of Rinnoy Kan et al. [7].

For any $Q \subseteq N$, $\quad P(Q) = \sum_{\pi(i) \in Q} p_{\pi(i)}$.

$$\text{Let } T_i(q) = \min\{P(Q) : Q \subset N \backslash \left\{\Gamma_i^- \cup \Gamma_i^+ \cup \{i\}\right\}\},\ \text{where } q = |Q|.$$

$$\text{Next } t_{ij} = P(\Gamma_j^-) + p_i + T_i\left(j - \left|\Gamma_j^-\right| + 1\right), \quad \left|\Gamma_i^-\right| \leq j \leq \left|N \backslash \Gamma_i^+\right|,$$

and

$$c_{ij} = \begin{cases} w_i t_{ij} & \text{for } \ |\Gamma_i^-| \leq j \leq |N \setminus \Gamma_i^+|, \\ \infty & \text{for } \ 1 \leq j < |\Gamma_i^-| \text{ and } |N \setminus \Gamma_i^+| < j \leq n. \end{cases}$$

The lower bound is equal to the optimal solution of the following assignment problem: Minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$, for

$$\sum_{i=1}^{n} x_{ij} = 1,\ \ j = 1, 2, \ldots, n,\ \ \sum_{j=1}^{n} x_{ij} = 1,\ \ i = 1, 2, \ldots, n \text{ and } x_{ij} \in \{0, 1\}$$

In the paper of Rinnoy Kan et al. [7] the lower bound for TWTS problem is calculated similarly.

The Upper bound.

The upper bound is the value of a solution calculated by the heuristic algorithm WBH:

Step 1: Order n jobs by decreasing of due date d_i (EDD schedule);

Step 2: Take the first two jobs and schedule them in order to minimize the partial goal function as if there were only these two jobs;

Step 3: For k=3 **to** n **do**

insert the k-th job in the place, among k possible ones, which minimizes (for k jobs) the partial goal function.

The complexity of algorithm is $O(n^2 m)$.

6 Computer simulations

We have tested the proposed algorithm on a set of randomly generated problems on a Sun Enterprise 8x400MHz using Ada95 language and MPI library. For each job i, an integer processing time p_i was generated from the uniform distribution [1, 100] and an integer weight w_i was generated from the uniform distribution [1, 10]. Let $P = \sum_{i=1}^{n} p_i$. The deadlines d_i distribution depends on P and two additional parameters L and R, where L takes on values from 0.2 to 1.0 in increments of 0.1, and R takes on values from 0.2 to 1.6 in increments of 0.2. An integer deadline d_i was generated from the uniform distribution $[P(L - R/2), P(L + R/2)]$. Ten instances of the problem were generated for each of the 40 pairs of R and L values, creating 400 problems for each value of n. Number of jobs was n=20, 40, 60, 80, 100, 120. Many of the generated instances have no feasible sequence. This fact can be quickly determined using the EDD rule.

There are computational results for two heuristics EDD and WBH in Table 1.

Table 1. Heuristic algorithms.

	n=20		n=40		n=60		n=80		n=100		n=120	
	δ_{avg}^1	$\delta_{\max}^2$	δ_{avg}	$\delta_{\max}$	δ_{avg}	$\delta_{\max}$	δ_{avg}	$\delta_{\max}$	δ_{avg}	$\delta_{\max}$	δ_{avg}	$\delta_{\max}$
EDD	36.2	49.6	43.5	55.1	49.7	61.7	48.6	68.6	53.7	72.4	56.9	93.7
WBH	18.9	26.6	21.4	32.7	25.8	35.4	27.1	39.7	26.2	33.6	31.4	41.8

[1]average percentage deviation to the lower bound.

[2]maximal percentage deviation to the lower bound.

Initial population:

Property 1 follows if in permutation $\pi \in \Phi_n$ all jobs are feasible ($C_{\pi(i)} \leq d_{\pi(i)}$, $i \in N$), then π is also a feasible solution, $\pi \in \Pi$. In a classical genetic algorithm the initial population P_0 is a randomly generated subset of a feasible set of solutions, ($P_0 \subset \Phi_n$). For many instances of the SWCT problem it is very hard to determine this kind of subset. Therefore, we generate the initial population, which also includes unfeasible solutions. In further iterations of the algorithm unfeasible solutions are eliminated using the decreasing deadline technique.

For each of the test instances of the problem, we compute the following values: F^{UB} – upper bound and F^* – the value of objective function found by the genetic algorithm. Having these values, we define a relative measure for the algorithm quality: $PRD^{UB} = ((F^{UB} - F^*)/F^*) \cdot 100\%$ – the mean value of the percentage relative deviation of the objective function F^{UB} with respect to F^*.

The computational results can be found in Table 2. The number of iterations was counted as a sum of iterations on processors, and was permanently set to 3,200. For example, 4-processor implementations make 800 iterations on each of the 4 processors, 8-processor implementation – 400 iterations on every processor.

As we can see in Table 2, results of the algorithms are the best for the 8-processor implementation of parallel genetic algorithm. The improvement of the relative distance to reference solutions was at the level of 22.6%, compared to solutions of the sequential algorithm, with the same number of iterations equal to 3,200 for the sequential and parallel algorithm. The time of the computing amount of a few seconds up to a few dozen seconds, depends on the size n of the problem instance.

Table 2. Percentage relative deviation of PGA solutions compared to the solutions of WBH constructive algorithm. NFI – number of feasible instances.

n	NFI	processors		
		1	4	8
20	229	-4.796%	-6.020%	-6.110%
40	245	-5.685%	-5.615%	-7.902%
60	249	-5.700%	-6.010%	-8.474%
80	251	-6.901%	-7.903%	-9.025%
100	250	-10.103%	-10.496%	-11.595%
120	251	-9.816%	-10.004%	-9.608%
average		-7.167%	-7.675%	-8.786%

7 Conclusions

We have discussed a new approach to the single machine total weighted completion time problem based on the parallel asynchronous genetic algorithm. The advantage is especially visible for large problems. As compared to the sequential algorithm, parallelization increases the quality of the solutions obtained. The idea of dynamic floating deadlines and the best individual migration was used. Computer experiments show, that the parallel algorithm is considerably more efficient with relation to the sequential algorithm.

References

1. Bensal S.: Schingle machine scheduling to minimize weighted sum of completion times with secondary criterion – a branch and bound approach. European J. Oper. Res. **5** (1980) 177–181
2. Goldberg D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Inc., Massachusetts (1989)
3. Holland J.H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press (1975)
4. Lenstra J.J., Rinnoy Kan A.H.G., Brucker P.: Complexity of machine scheduling problems. Ann. Discrete Math. (1979) 287–326
5. Pan Y.: An improved branch and bound algorithm for single machine scheduling with deadlines to minimize total weighted completion time, Operation Research Letters **31** (2003) 492–496
6. Posner M.: Minimizing weighted completion times with deadlines, Oper. Res. **33** (1985) 562–574
7. Rinnoy Kan A.G.H, Lageweg B.J., Lenstra J.K.: Minimizing total cost one-machine scheduling. Operations Research **26** (1975) 908–972
8. Smith W.E.: Various optimizers for single-stage production, Naval Res. Logist. Quart. **3** (1956) 59–66
9. Wodecki M., Bożejko W., Solving the flow shop problem by parallel simulated annealing, LNCS No. **2328**, Springer Verlag (2002) 236–247