Algorithms for scheduling of resource-constrained jobs

Ivaylo Penev[1], Milena Karova[2]

Abstract – **The paper presents two heuristic algorithms for scheduling resource constrained jobs in an environment, consisting of identical connected computers. The first algorithm uses sorted list of jobs and is applicable to the particular case when the number of jobs is not greater than the number of available computers. For the general case of the problem with arbitrary number of jobs and computers a genetic algorithm is proposed. The experimental results present comparison of solutions, obtained by the algorithms, performed on test data sets with the exact solutions, produced by a classical branch and bound algorithm.**

Keywords – **job-scheduling, resource constraints, heuristics, genetic algorithm.**

I. Introduction

The goal of the resource-constrained job-scheduling problem is constructing a schedule with starting moments for each job on an available computer from the environment, such that preliminary defined limiting criteria are satisfied and extreme value (minimum or maximum) of objective function is achieved.

The following common approaches and algorithms for solving the resource-constrained job-scheduling problem are known.

- Algorithms for producing exact solution;

The most frequently used exact method is the branch and bound algorithm. The algorithm performs exhaustive check of all branches of a tree with possible solutions (i.e. all possible schedules). For reducing the space of solutions the tree is usually pruned by the usage of a bound value for the examined branch.

The results of many realizations of the branch and bound algorithm are published, for example for solving the resource constrained project scheduling problem [1].

- Algorithms for producing good (near the exact) solution;

Most scheduling problems are NP-hard [2]. For finding good solution, which is close to the exact one, heuristic approaches and algorithms are applied. Multiple heuristic rules and algorithms for solving scheduling problems are suggested, summarized for example in [3]. Genetic algorithms are among the most often algorithms used. A genetic algorithm, solving common resource-constrained scheduling problem is presented for example in [5].

[1]Ivaylo Penev is with the Department of Computer Sciences and Technologies at Technical University - Varna, 1 Studentska str., Varna 9000, Bulgaria, E-mail: ivailopenev@yahoo.com.

[2]Milena Karova is with the Department of Computer Sciences and Technologies at Technical University - Varna, Bulgaria.

The paper presents two heuristic algorithms for scheduling of resource-constrained jobs. The first algorithm is based on sorted lists of jobs. The second one is genetic algorithm. The solutions, obtained by these algorithms are compared to the exact solution, produced by classical branch and bound algorithm. The authors demonstrate, that the proposed heuristic algorithms produce near the exact solutions for significantly reduced time.

II. Formal description of the resource-constrained job-scheduling problem

The resource-constrained job-scheduling problem could be formally defined by the following sets.

1. Jobs

$J = \{j_i \mid i \leq n\}$ - set of jobs

$j_i = \{v_i, q_i, w_i\}$ – set of sub jobs for each job

v_i – reading data from common resource

q_i – calculation operations

w_i – storing results into the common resource

$t(v_i), t(q_i), t(w_i)$ – completion time for each sub job of job j_i

C_i – completion time of job j_i

2. Resources

Sub jobs v_i and w_i uses common resource. Sub job v_i requires $r(v_i)$ units and sub job w_i requires $r(w_i)$ units of the common resource.

3. Temporal constraints

$S = \{S_i \mid i \leq n\}$ – set of starting moments along the time axis for each job

$S(v_i) = \{S(v_i) \mid i \leq n\}$ – set of starting moments for sub job v_i

$S(q_i) = \{S(q_i) \mid i \leq n\}$ – set of starting moments for sub job q_i

$S(w_i) = \{S(w_i) \mid i \leq n\}$ - set of starting moments for sub job w_i.

The sub jobs of each job are performed in the following order – reading data, calculation operations, storing data, i.e.

$\forall j_i \in J \ \exists \ S_i \rightarrow (S_i + t(v_i) \leq S(q_i)) \cap (S(q_i) + t(q_i) \leq S(w_i))$

The completion time of each job (which means completion of all sub jobs) in a schedule is defined as $C_i = S(w_i) + t(w_i)$. The completion time of the last job in a schedule is $C_{max} = max(C_i)$ and is called ***makespan***.

4. Mathematical model of the resource-constrained job-scheduling problem

Assuming the above notations the problem for scheduling **n** jobs in an environment of **m** identical connected computers, using constrained common resource could be formally described:

$min(Cmax),\ i = 1, \ldots, n$ – minimum makespan - objective function (1)

$(S_i + t(v_i) \le S(q_i)) \cap (S(q_i) + t(q_i) \le S(w_i))$ - temporal constraints (2)

$$\sum_{i=1}^{n} (r(v_i) \cup r(w_i)) \le 1 \text{ – resource constraints} \qquad (3)$$

A feasible solution of the scheduling problem is assignment of starting moments to all sub jobs, i.e. *$(S(v_1), S(q_1), S(w_1), \ldots, S(v_n), S(q_n), S(w_n))$*, which satisfies the temporal constraints (2) and the resource constraints (3). The **exact solution** is the minimum value of all possible values of the completion time of the last job in the schedule, i.e. the exact solution guarantees minimum makespan *Cmax* in the schedule.

III. Heuristic algorithms for solving the resource-constrained job-scheduling problem

A. Algorithm with sorted lists of jobs for the particular case $n \le m$ of the resource-constrained scheduling problem

The algorithm uses two lists with jobs. It is presented by the following pseudo code:

```
List1 ← $j_i$, for $\forall i<j \rightarrow t(v_i) + t(q_i) \ge t(v_j) + t(q_j)$
$S_1 = 0$
for each job in List1 do
    $S_i = S_{i-1} + t(v_{i-1})$
endfor
List2 ← List1, for $\forall i<j \rightarrow S_i + t(v_i) + t(q_i) \le S_j + t(v_j) + t(q_j)$
$S_1 = 0$
$j_1$ ← List2 – Get first job from List2
$t_1 = S_1 + t(v_1) + t(q_1) + t(w_1)$
for each job in List2 do
    if $S_i+t(v_i)+t(q_i)>=S_{i-1}+t(v_{i-1})+t(q_{i-1})+t(w_{i-1})$
        $t_i = S_i+t(v_i)+t(q_i)+t(w_i)$
    else
        begin
            $\Delta t(w_i) = t_{i-1} - (S_i + t(v_i) + t(q_i))$
            $t_i = S_i + t(v_i) + t(q_i) + \Delta t(w_i) + t(w_i)$
        end
    endif
endfor
j ← Get last job from List2
$C_{max} = j$
```

B. Genetic algorithm for the general case $n \le m$ of the resource-constrained scheduling problem

The genetic algorithm iteratively chooses the best candidates (individuals) of a set of possible solutions. The choice is based on the calculation of an objective function of the genetic algorithm, which presents the number of time units, in which two or more jobs access the common resource concurrently. At each iteration the algorithm shifts an arbitrary job on the time axis and calculates the objective function. If the shifting decreases the objective function's value, the solution is saved in the population.

B. 1. Distribution of the tasks between the computers

To assign jobs to a computer a list with the jobs, sorted by execution time is used (i.e. t(vi) + t(qi) + t(wi)). Each job from the list is afterwards assigned to a computer. After the first m jobs are assigned to m computers, the assignment of jobs continues in reverse order.

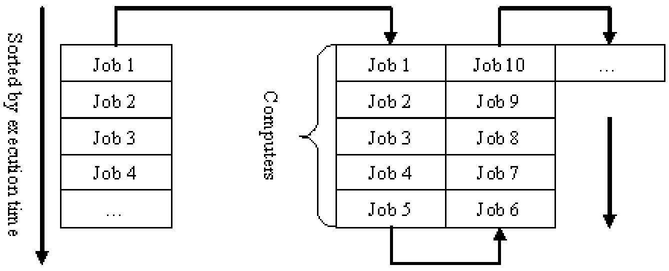

Fig. 1. Distribution of jobs between the computers

B.2. Coding of the chromosomes

Each chromosome presents a list with all available computers. Each computer is assigned to a job with starting moment and known times for reading data, calculation operations and storing results.

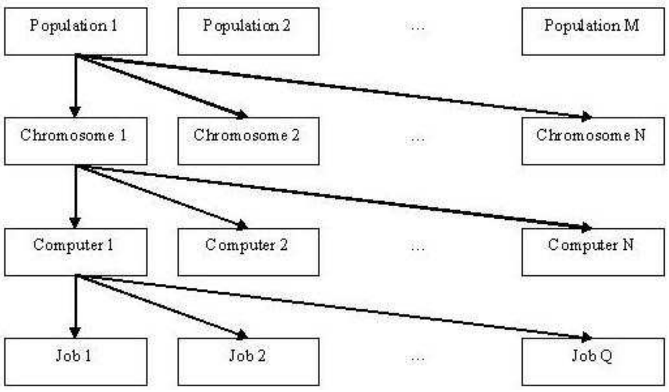

Fig. 2.Structure of the genetic algorithm

Each chromosome is presented as a string. Each element of the string is coded with one of two possible values – 0 or 1. The time units, in which the job uses the common resource is presented with 1, the others are presented with 0.

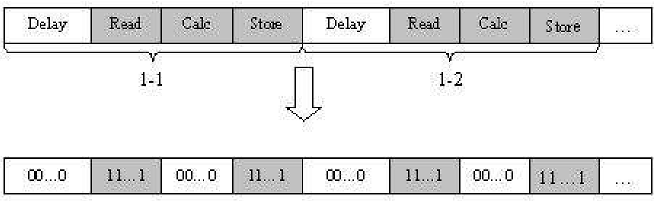

Fig. 3. Structure of a chromosome

B.3. Objective function of the genetic algorithm

The objective function of the chromosome is calculated by intersection of the strings of each pair of computers from the chromosome. The number of "1"-s is added to the sum of time units with concurrent access to the constrained resource.

Algorithm for calculating the objective function's value of the genetic algorithm:

```
foreach chromosome
  sum_of_time_units_with_concurrent_access = 0
  for i = 1 to m-1
    for j = i + 1 to m
      result_string = $string_i \cap string_j$
      sum_of_time_units_with_concurrent_access = sum_of_time_units_with_concurrent_access + number of “1”-s in result_string
    endfor
  endfor
end foreach
```

B.4. Genetic operators

Mutation is performed by choosing an arbitrary job from an arbitrary computer of the chromosome. The job is then shifted on the time axis. The objective function is calculated. If the number of time units with concurrent access to the common resource is decreased, the shifted job is saved in the population.

IV. Experimental results

The branch and bound algorithm and the two presented algorithms are tested for constructing schedules with different number of jobs. Times for reading data, calculation operations and storing results for each job are generated. For each number of jobs 100 attempts are performed. The minimum makespan of the schedule and the necessary time for constructing the schedule are measured.

Table I shows the deviation of the makespans, produced by the algorithm with sorting and the genetic algorithm from the corresponding exact makespan, obtained by the classical branch and bound algorithm. The deviations are measured in percents.

TABLE I

DEVIATIONS OF THE HEURISTIC ALGORITHMS’ MAKESPANS FROM THE BRANCH AND BOUND’S MAKESPAN

Number of jobs	Sorting (%)	Genetic algorithm (%)
2	0	0
3	0,74	0,95
4	1,92	3,07
5	1,19	0,89

Table II shows comparison of the average time, necessary to the algorithms to produce the makespan. The following notations are used:

BB – Branch and bound algorithm
Sorting – Algorithm with sorting
GA – Genetic algorithm

TABLE II

COMPARISON OF TIMES FOR PRODUCING THE MAKESPAN

Jobs	BB (ms)	Sorting (ms)	GA (ms)
3	3	3,29	23,10
4	63,85	2,68	288,67
5	305,67	2,85	275,56
7	11437	2,15	6645
8	110554	6,69	880
9	6459285,35	65545	10880

The next figures demonstrate the trend of the time, necessary to all the algorithms for constructing the schedule with the minimum makespan for different number of jobs.

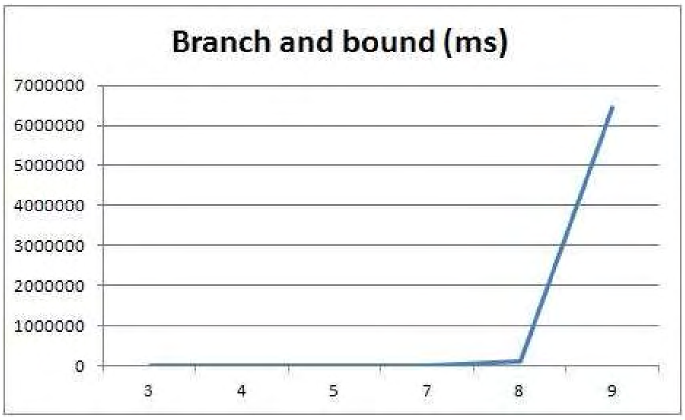

Fig. 4. Trend of time, necessary to the branch and bound algorithm for producing exact minimum makespan

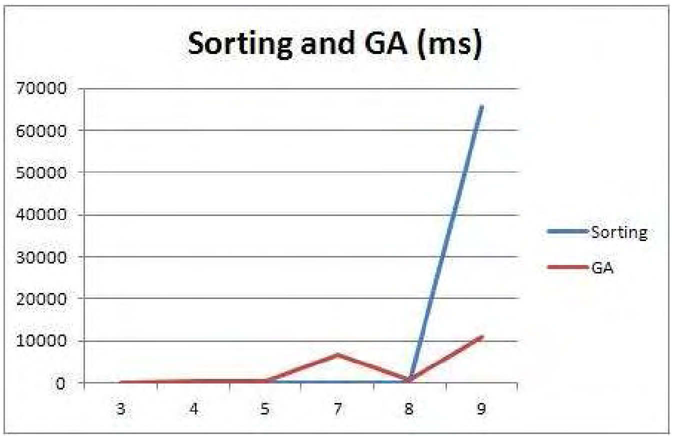

Fig. 5. Trend of time, necessary to the algorithm with sorting and the genetic algorithm for producing near the exact minimum makespan

V. Conclusions

The experimental results prove, that both the proposed algorithms produce near the exact minimum makespan of the constructed schedule. The algorithm with sorted lists of jobs is very easy to implement and produce makespan for significantly reduced time in comparison with the branch and bound algorithm. The genetic algorithm constructs schedule for least time in comparison with the other algorithms and is applicable to the general case of the scheduling problem with arbitrary number of jobs and identical connected computers.

The presented algorithms will be used in a real portfolio management system for evaluation of various financial objects' risk. Simulation analyses must be performed with historical data of portfolios. Typically there are no data dependencies between the data of the portfolios, which makes possible parallel execution of the estimations. The historical data are stored into common resource. The concurrent access to the common resource decreases the efficiency of the parallel execution [4]. The proposed heuristic algorithms will be used to construct schedule with jobs, estimating portfolios, which ensures completing of all jobs for polynomial time.

REFERENCES

[1] Dorndorf, U., W. Pesch, T. Phan-Huy. A Time-Oriented Branch-and-Bound Algorithm for Resource-Constrained Project Scheduling with Generalised Precedence Constraints.

[2] Garey, M., D. Johnson. Computers and intractability: A Guide to the Theory of NP-Completeness. Freeman. 1979.

[3] Hartmann, S., D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research, vol. 207, iss. 1, pp. 1 – 14, 2010.

[4] Penev, I. Realization of Portfolio Management System in a Distributed Computing Environment. Proceeings of Fifth International Scientific Conference Computer Science'2009, ISBN: 978-954-438-853-9, pp. 291 ÷ 296.

[5] Wall, M., B., A Genetic Algorithm for Resource-Constrained Scheduling, partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mechanical Engineering, Massachusetts Institute of Technology, 1996.