ISSN 1392-8619 print/ISSN 1822-3613 online
ŪKIO TECHNOLOGINIS IR EKONOMINIS VYSTYMAS
TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY
http://www.tede.vgtu.lt
2006, Vol XII, No 4, 307–313

OPTIMIZATION OF RESOURCE-CONSTRAINED PROJECT SCHEDULES BY SIMULATED ANNEALING AND VARIABLE NEIGHBORHOOD SEARCH

Leonidas Sakalauskas[1], Gražvydas Felinskas[2]

[1]*Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223 Vilnius, Lithuania*
E-mail: sakal@ktl.mii.lt
[2]*Šiauliai university, Faculty of Mathematics and Informatics,*
P. Višinskio g. 19, Šiauliai, Lithuania. Tel. +370 6 863 8156
E-mail: grazvis@splius.lt

Received 15 June 2006; accepted 20 November 2006

Abstract. Applications of information technologies are often related to making some schedules, timetables of tasks or jobs with constrained resources. In this paper we consider job scheduling and optimization algorithms related to resources, time and other constraints. Schedule optimization procedures, based on schedule coding by priority list of jobs, are created and investigated. Optimal priority list of jobs is found by approaching algorithms of local and global search, namely, random search and simulated annealing methods with the variable neighborhood, defined by the decoding procedure applied. Computational results with testing data from project scheduling Library are given.

Keywords: resource-constrainted project, schedule optimization, Monte Carlo method, simulated annealing, variable neighborhood.

1. Introduction

Information systems and their applications are often related to making some schedules, timetables of tasks or jobs with constrained resources. Input data for such problems is a set of jobs, their duration, priority rules (successors, predecessors), and necessary resources. The aim is to find such a schedule, which meets the requirements of job priority relations, resource constraints minimizing it by some criteria. In many cases, this criterion is project's finishing time.

In this paper, we consider job scheduling and optimization algorithms related to resources, time, and other constraints. In most cases, complexity of scheduling problems belongs to complexity class NP [1]. Optimal solution can be found using full binary recombination [2] or using methods based on ideas of branch and bound methods [3], etc. It is usually difficult to perform a full binary recombination in real computer time, therefore, in order to schedule and optimize jobs, we may apply heuristic methods based on priority rules [4], evolution process ideas [5, 6], local search [7–9], variable neighborhood methods [10, 11], etc. Test results show that development and investigation of these methods are the promising direction to create efficient scheduling methods. Schedule optimization procedure discussed in this paper is based on coding and decoding of schedule depending on job priority list. Optimal priority list of jobs can be found by composing global search procedures and algorithms of local search.

In this paper, we explore application of Monte Carlo [12], simulated annealing [13] methods to schedule optimization. Computational results are given using data sets from project scheduling library (PSPLib) [14, 15].

2. Formulation of the schedule optimization problem

Input data for schedule planning and optimizing is a set of jobs and their duration, definition of resource constraints, a set of priority relations.

Let us denote $J = \{0,1,...,n,n+1\}$ a set of jobs, jobs *No*. 0 and *No*. $(n + 1)$ are dummy and means the beginning and the end of the whole project, $d_j, j \in J$ – duration of j^{th} job, $d_j \geq 0, j \in J$ – non-negative numbers and $d_0 = d_{n+1} = 0$.

We can define priority relations in a set J as a set of

pairs $C = \{(i, j) \mid i$ must be executed before $j\}$. Let us denote a set of resources $K = \{1,...,m\}$. All resources are renewable and non-additive – at every moment we can use fixed amount of each type of resources, remains are gone. Let us assume that amounts of resources $R_k > 0,\ k \in K$ are constant. Let us denote the starting moment of j^{th} job by $s_j \geq 0$, and correspondingly $r_{jk} \geq 0$ – amount of k^{th} resource, needed for performing this job. Let us assume started jobs must be performed without breaks. Then the finishing time of j^{th} job can be defined as $c_j = s_j + d_j$. The problem of schedule making may be reduced to the problem of finding a vector $s = (s_0, ..., s_{n+1})$ of jobs' starting time which meets priority and resource requirements and minimizes a certain objective function. An objective function may reflect the economical outlay (outgoing), yield (incoming), finishing time, etc. Project finishing time is one of the most analyzed schedule optimality criteria which will be applied in this paper.

Let us denote $A(t) = \{j \in J \mid s_j \leq t < c_j\}$ – a set of executable jobs at the time moment t, $T(s)$ – project finishing time, $T(s) = c_{n+1}$.

After these definitions we can formulate the problem minimizing the objective function $T(s)$.

Find $\min\limits_{s} T(s)$, subject to:

$$c_i \leq s_j, \quad (i, j) \in C,$$

$$s_j \geq 0,\ c_j = s_j + d_j,\ j \in J, \tag{1}$$

$$\sum_{j \in A(t)} r_{kj} \leq R_k,\ 0 \leq t \leq c_{n+1}. \tag{2}$$

The objective function defines the finishing time of the whole job project, inequality (1) describes priority relations, and inequality (2) requires to pay heed to resource constraints.

3. Schedule coding and decoding

Efficiency of schedule optimization algorithm depends on solution coding [10, 11, 16]. In this paper, we analyze job scheduling problems under resource constraints with solution coding in the shape of a priority list [4, 17]. The job priority list $b = (0, b_1, b_2, ..., b_n, b_{n+1})$ can be determined by jobs' starting time vector s, where $s_{b_i} \leq s_{b_j}$, if $i < j$. It is very important that this vector of starting moments must meet priority relations and resource constraints. On the other hand, for given priority list, we can find the vector of job starting time concerted with priority list and initial priority constraints. For this we use the serial decoder described in section 3.2.

At first, we must be sure that the priority list is concerted with priority constraints. Let us denominate the priority list from which we can find a vector of jobs' starting time concerted with priority relations as a feasible priority list. Using a set of priority relations we can check whether the solution is feasible or not. For any feasible job priority list, by applying the serial decoding procedure, we can determine jobs' starting time vector which may minimize the project finishing time, according to priority relations, resource constraints, and a given job priority list. Operating with job priority lists, the constructed algorithm must enable us to find such job priority list which corresponds to an optimal solution of the problem (1), (2).

3.1. Determination of admissibility of the job priority list

Let us introduce the binary matrix $V = (v_{ij},\ 1 \leq i, j \leq n)$, $v_{ij} = 1,\ if\ (i, j) \in C, v_{ij} = 0,\ if\ (i, j) \notin C$, related with a set of priority constraints and define a full priority relations matrix $G = (g_{ij}, 1 \leq i, j \leq n)$. This matrix describes all jobs which must be done corresponding to all chains of priority relations. So, $g_{kj} = 1$, if it is possible to find such a sequence of index pairs that $(k, k_1) \in C$, $(k_1, k_2) \in C$, ..., $(k_l, j) \in C$. The matrix V has the following feature: $v_{ij} = 1 \Rightarrow v_{ji} = 0$. The matrix G also has this feature. The priority list is feasible if b_i, b_j are the components of b, $i < j \Rightarrow g_{b_j, b_i} = 0$ applies for any pair (b_i, b_j).

3.2. The serial priority list decoder

This procedure computes the early starting moments of jobs, according to jobs' priority list concerted with priority relations, and resource constraints. In schedules obtained in such a way, none of jobs can be started earlier than calculated time, without break of priority relations or resource constraints. We can call such schedules active ones [11]. Let us describe the algorithm for determination of the active schedule.

Step 1. Set initial $s_0 = c_0 = 0$. Let $i = 1$.

Step 2. Let us agree that starting time of the first $(i-1)$ jobs from the priority list have already been determined. So, we know all $s_{b_j}, c_{b_j},\ j \leq i-1$.

Let us denote the moment:

$$T_i = \max\Big(\max_{\substack{g_{b_l, b_i} = 1 \\ 0 \leq l \leq i-1}} c_{b_l},\ \max_{0 \leq l \leq i-1} s_{b_l}\Big).$$

The starting time of the next job is equal to this moment if resource constraints are met:

$$s_{b_i} = T_i, \text{ if } \sum_{j \in A(T_i)} r_{k_j} \leq R_k, 1 \leq k \leq K.$$

If resource constraints are not met, then the starting time of this job is equal to finishing time of the first finished job, when resource constraints are met:

$$s_{b_i} = \min_{\sum_{j\in A(c_{b_j})} r_{kj} \le R_k, 1\le k\le K,\ c_{b_j} > T_i, 1\le j\le i-1} c_{b_j}.$$

The finishing time of this job can be calculated simply:

$$c_{b_i} = s_{b_i} + d_{b_i}.$$

Step 3. If $i = n$, then $\{T(S) = c_{b_n} + d_{n+1}$, end of algorithm$\}$, or else $\{i = i+1$, go to step 2$\}$.

It is easy to see that it is necessary to make $O(n \cdot m)$ elementary operations for performing step 2 of the algorithm [11]. Therefore, the complexity of decoding procedure from the viewpoint of computer operations number is $O(n^2 \cdot m)$.

4. Schedule optimization algorithms

Using the serial decoding procedure of the job priority list we can construct schedule optimization algorithms, applying several random heuristic procedures. We investigate the efficiency of these procedures by statistical simulation.

4.1. Optimization by crude random search

In the simplest way, we can apply a crude random search by Monte Carlo method – to generate some number of random priority lists, select from them the feasible list with minimal finishing time using proper amount of resources. Let us describe the statistical simulation algorithm of schedule optimization by Monte Carlo method.

Let us denote $MAXIT$ – a number of iterations, $MAXD$ – a number of descents by random search, $AVGZ$, $AVGZF$ – an average of reached objective function values after $MAXD$ descents, T – objective function value, $STEP$ – number of iterations for fixing intermediate reached objective function values.

Step 1. $SUMZF = 0$, $SUMZ = 0$, $M = 0$.
Step 2. $IT = 0$, $Z^* = 1e10$.
Step 3. $IT = IT + 1$. Generate feasible priority list $b = (b_1, \ldots, b_N)$, N – number of jobs: Generate sequentially numbers of jobs b_i in such a way that $b_i \ne b_j$, $1 \le i, j \le N$, $g_{b_j, b_i} = 0$, $i < j$.
Step 4. Calculate the value of objective function $Z = T(b)$. If $Z < Z^*$, then $Z^* = Z$.
Step 5. If $(IT \bmod STEP = 0)$ then $SUMZ[IT \operatorname{div} STEP] = SUMZ[IT \operatorname{div} STEP] + Z^*$.
Step 6. If $IT < MAXIT$, then go to **Step 3**, otherwise go to **Step 7**.
Step 7. $SUMZF = SUMZF + Z^*$.
Step 8. $M = M + 1$. If $M \le MAXD$, then go to **Step 2**, otherwise go to **Step 9**.
Step 9. $AVGZ[i] = SUMZ[i] / MAXD$, for all $i = 1..(MAXIT \operatorname{div} STEP)$; $AVGZF = SUMZF / MAXD$.

Step 5 allows us to fix intermediate average values of the objective function and, later, to compare them with the average of best reached value of the objective function after $MAXD$ descents.

4.2. Optimization by simulated annealing method

Let us consider simulated annealing (SA) algorithm based on the priority list and the serial decoding procedure [4, 17]. The main idea of algorithms such as SA, is the solutions generation method and special rules for accepting new randomly generated solutions [13]. New solutions are generated from the current solution environment, while calculating the values of the objective function. Usually depth (radius) of environment is decreasing through optimization procedure, starting from the fixed value. In order to move from the current solution to a new one we apply Metropolis rule, which allows, with a certain probability, to accept solutions with a worse objective function value (accept worse solution with the probability $p = \exp(\frac{T(b) - T^*(b)}{t_k})$, here $T(b)$ is current solution, $T^*(b)$ – new solution).

Theoretical recommendations for choice of appropriate parameters of SA in continuous optimization, that gain the algorithm's convergence to global optimum, has been considered in [16, 18]. We applied these suggestions to scheduling problems introducing the following rules:

$$\rho_k = \rho_0 / k^{\alpha}, \quad t_k = t_0 / k^{\beta}, \quad 0 < \alpha, \beta < 1, \qquad (3)$$

here k – iteration number (variable), the parameter ρ_k models depth of environment, t_k corresponds to the parameter of temperature in SA algorithm. We used the initial values $\rho_0 = N / const$ and $t_0 = MJD / const$ (MJD – maximum job duration, $const$ – simple constant which depends on problem dimension). During optimization this method accepts not only the better solutions; while applying Metropolis acceptance criterion, this method with a little probability (which also depends on the temperature parameter t) also accepts worse schedules than the current ones. Influence of adjusting rules for the parameters ρ_k and t_k on convergence speed and accuracy of the solution was considered in papers [16, 18].

Below we discuss about constructing of the SA algorithm for schedule optimization. Let us call two priority lists as neighboring, if we can get one from another by applying only one elementary operation, such as moving a

job from one priority list's position to another (Shift) or replacing two jobs positions in priority list (Swap) [19].

Let us consider l and q as holding two positions in priority list, $l \neq q$, $1 \le l, q \le n$.

We can get the priority list b' from the list b by applying Shift operation, accordingly renumbering the job priority list: if $q > l$, then $b'_x = b_{x+1}$, $l \le x < q$, if $q < l$, then $b'_x = b_{x-1}$, $q < x \le l$. After this let us set $b'_q = b_l$. Other elements of the list b are the same as in the list b. We can get the priority list b' from the list b by applying Swap operation in the following way: $b'_q = b_l$, $b'_l = b_q$. Other elements of b' are the same as in b. We can get swap operation by applying shift operation twice.

The SA algorithm is as follows:

Step1. $k = 0$.

Step 2. Let us assume that k steps of the algorithm are already done. The current job priority list is b, objective function value with this list is $Z1 = T(b)$ (calculated using the serial priority list decoder described in section 3.2). Calculate step parameters ρ_k and t_k applying (3).

Step 3. Randomly generate numbers q and l, $l \neq q$, $1 \le l$, $q \le n$. With the probability $p_e = 0.5$, perform either shift or swap operation.

Step 4. If priority list obtained in such a way is not feasible then repeat step 3 until obtained priority list is feasible.

Step 5. Repeat steps 3 and 4 no less than ρ_k times. Let us denote b' priority list obtained from current b after performing no less than ρ_k allowable elementary operations.

Step 6. Calculate the objective function value $Z2 = T(b')$ for the obtained priority list using the serial decoding procedure.

Step 7. Change the current priority list to another by adapting Metropolis rule:

if $\eta < \exp((Z1 - Z2)/t_k)$, then $b = b'$ (η – randomly generated number from $(0,1)$)

Step 8. $k = k + 1$. If $k < k_{\max}$, then repeat step 2.

Several modifications of this basic algorithm can be studied. In this algorithm numbers q and l were generated with equal probabilities. However, we can set special probabilities for obtaining numbers q and l, depending on jobs positions in the priority list, the priority relations matrix, etc.

5. Computational results

Constructed algorithms were tested with data sets from project scheduling library PSPLib. In PSPLib one can find a lot of data set instances of various classes of RCPSP [14, 15].

5.1. Optimization by using crude random search and simulated annealing methods

Several test data sets (also the best known solutions and in cases of 30 jobs – global optimums are given) were chosen with various numbers of jobs, their priority relations, various needs, and resources. Convergence of objective function was compared while applying random search (marked "M-C") as well as the method of simulated annealing (SA) (consider Figures 1, 2, 3 and 4, accordingly with 30, 60, 90 and 120 jobs). Due to the method of random search, 100 descents each were performed and average of the objective function value after 100, 200, 300, 500, and 1000 iterations each was fixed. Feasible priority list was generated in every iteration; while comparing with the previous reached value, improvement of objective function value was searched.

The same comparative investigations were carried out while applying the method of simulated annealing (in Figures marked "SA"). In every iteration the next priority list is being formed after application of elementary operations (according to SA method) to the current priority list. It is being accepted or rejected. Since the computing time for performance of a single iteration is similar for all studied algorithms, the best achieved value of objective function may be used as the criteria of algorithm quality. Thus, the best reached objective function value (marked "best") was fixed at 100, 200, 300, 500, and 1000 iterations each. Optimization while applying SA method was started from random priority list. Figures 1, 2, 3, 4 display graphs which show that SA method's advantages are clearly highlighted after a small number of iterations. The average of objective function values (100 descents) achieved by SA method differs little from the best value achieved. This proves the reliability of the method because deviation (dispersion) from the best value is not significant.

Hence, advantage of simulated annealing obviously becomes evident already after a small number of iterations; therefore, adjustment of environment depth was investigated while optimizing by SA method.

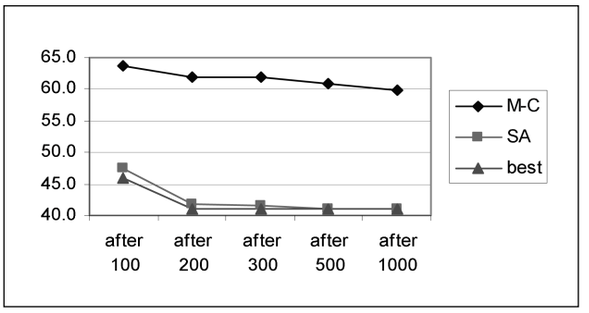

Fig 1. Convergence of the objective function value to global minimum (Number of jobs n = 30, Optimum of the objective function Opt = 41)

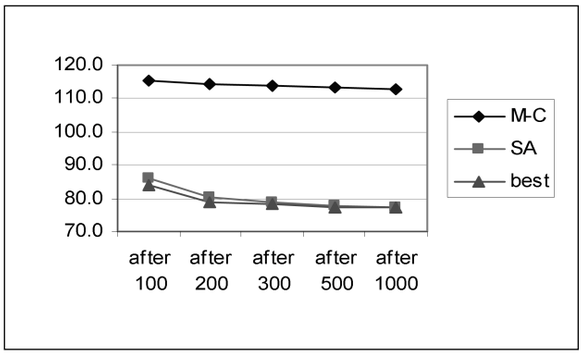

Fig 2. Convergence of the objective function value to global minimum (Number of jobs n = 60, Optimum of the objective function Opt = 77)

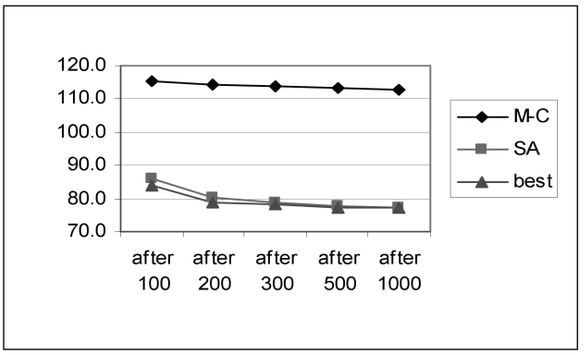

Fig 3. Convergence of the objective function value to global minimum (Number of jobs n = 90, Optimum of the objective function Opt = 67)

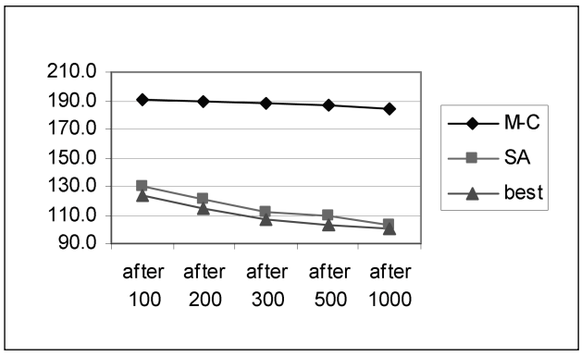

Fig 4. Convergence of the objective function value to global minimum (Number of jobs n = 120, Optimum of the objective function Opt = 99)

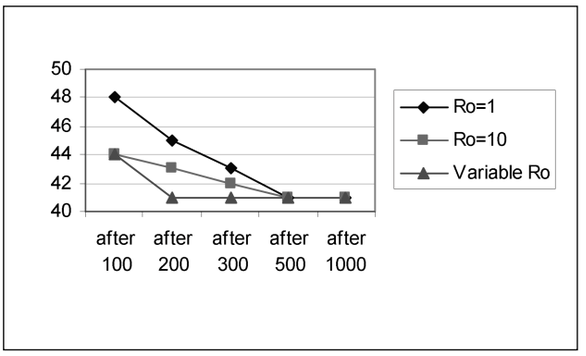

Fig 5. Convergence of the objective function value to global minimum using three types of environments (Number of jobs n = 30, Optimum of the objective function Opt = 41)

5.2. Optimization by using simulated annealing method with variable neighborhood

In section 4.2, we described in detail the SA algorithm. Number of elementary operations in every iteration determines what space of possible solutions may be generated. Optimization with a fixed number of elementary operations was performed in every iteration of optimization procedure. (Ro = 1 and Ro = 10. Consider Figures 5, 6, 7, 8). A small number of elementary operations which allow forming neighborhoods of the solution, form new solutions in a "close distance" from the current one. Therefore search for optimum takes more time – a big number of iterations is required in order to come close to the optimal solution. On the other hand, a fixed big number of elementary operations in every iteration even at the beginning provides a possibility for generation of possible solutions in a "bigger distance" from the current one (more chances to "cover" the space of solutions). However, during the process of optimization, when moving towards optimum, a too big change often only prolongs the search for optimum. Moreover, checking of generated lists takes more time. Adjustment of environment depth, according to a number of iterations, allowed us to improve speed of convergence (in Figures 5, 6, 7, 8 marked "Variable Ro"). This adjustment of environment depth in some cases allowed us to reach the maximum result (to find a known optimum) already after a small number of iterations.

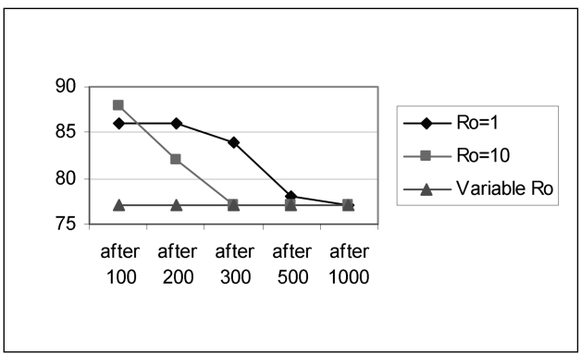

Fig 6. Convergence of the objective function value to global minimum using three types of environments (Number of jobs n = 60, Optimum of the objective function Opt = 77)

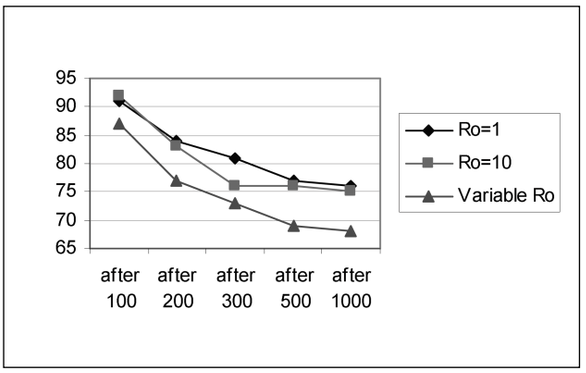

Fig 7. Convergence of the objective function value to global minimum using three types of environments (Number of jobs n = 90, Optimum of the objective function Opt = 67)

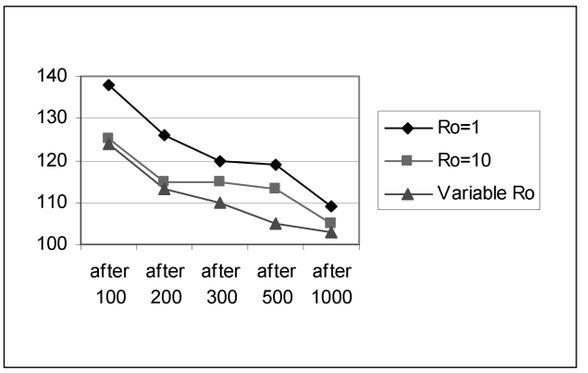

Fig 8. Convergence of the objective function value to global minimum using three types of environments (Number of jobs n = 120, Optimum of the objective function Opt = 99)

6. Conclusions and further research

1. Algorithms of planning and optimizing schedules of jobs with resource constraints based on schedule coding with the job priority list and the serial priority list decoding procedure were discussed in this paper. Such schedule coding allows us to apply several combinatorial optimization methods.

2. Random search and simulated annealing algorithms for RCPSP optimization were created. These algorithms were investigated and compared by statistical modeling, while using data sets and known solutions from PSP Library.

3. It was determined that the simulated annealing method is more efficient than random search and it allows reliably solve schedule optimization problems due to setting proper parameters of the method.

4. Minimization of the objective function, while applying several dynamic environment depth adjustment methods, showed that change of solution environment depth allows to accelerate convergence to global optimum.

5. Test results showed that created algorithms are able to find best known solutions of PSP Library while using practically acceptable calculating resources.

6. Nearest further plans: selection and adjustment of methods' parameters and application of genetic algorithms for optimization of a solution. This should improve convergence to global optimum.

References

1. Lassaigne, R.; Rougemont, M.; de. Logic and Complexity (Logika ir algoritmų sudėtingumas). Vilnius: Zara, 1999 (in Lithuanian).
2. Savage, Carla. A Survey of Combinatorial Gray Codes. *Society of Industrial and Applied Mathematics Review*, 39, 1997, p. 605–629.
3. Brucker, P.; Knust, S.; Schoo, A.; Thiele O. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 1998, Vol 107, No 2, p. 272–288.
4. Kolisch, R. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 1996, No 14, p. 179–192.
5. Genetic algorithms http://www.pcai.com/web/ai_info/genetic_algorithms.html, 2000.
6. Glibovec, N. N.; Medvidj, S.A. Genetic algorithms and their application to project scheduling problem solving. *Cybernetics and system analysis* (Кибернетика и системный анализ), 2003, No 1, p. 95–108 (in Russian).
7. Hoos, H. H.; Stutzle, Th. Stochastic Local Search. Foundations and Applications. Morgan Kaufmann / Elsevier, 2004.
8. Kolisch, R.; Hartmann, S. Project scheduling: Recent models, algorithms and applications. Kluwer, Amsterdam, 1999, p. 147–178.
9. Voss, St. Meta-heuristics: The State of the Art. *Local Search for Planning and Scheduling*, LNAI 2148, 2001, p. 1–23.
10. Hansen, P.; Mladenovic, N. An introduction to variable neighborhood search. In: Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, 1999, p. 433–458.
11. Kocetov, J. A.; Stoliar, A. A. Application of alternating environments to approximate solving of RCPSP. *Discrete analysis and operational research* (Дискретный анализ и исследование операций), 2003, Ser. 2, Vol 10, No 2, p. 29–55 (in Russian).
12. Wittwer, J. W. Monte Carlo Simulation Basics from Vertex42.com, June 1, 2004, http://vertex42.com/Excel Articles/mc/MonteCarloSimulation.html.
13. Kirkpatrick, S.; Gelatt, Jr. C. D.; Vecchi, M. P. Optimization by simulated annealing. *Science*, 1983, 220, p. 671–680.
14. Kolisch, R.; Sprecher, A. PSPLIB – A project scheduling library. *European Journal of Operational Research*, 1996, Vol 96, p. 205–216.
15. PSPLIB – A project scheduling library. http://www.bwl.uni-kiel.de/Prod/psplib/, 1996.
16. Felinskas, G.; Sakalauskas, L. Pareto type models in simulated annealing algorithms. *Lithuanian Mathematical Journal* (Lietuvos matematikos rinkinys), 2003, T. 43, special volume, p. 573–578. ISSN 0132-2818 (in Lithuanian).
17. Kolisch, R. Serial and parallel resource-constrained project

scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 1996, Vol 90, No 2, p. 320–333.

18. Yang, R. L. Convergence of the Simulated Annealing Algorithm for Continuous Global Optimization. *Journal of Optimization Theory and Applications*, 2000, Vol 104, No 3, p. 691–716.
19. Klein, R. Scheduling of Resource-Constrained Projects. Kluwer Academic Publishers, 2000. 369 p.

RIBOJAMŲ TVARKARAŠČIŲ IŠTEKLIŲ OPTIMIZAVIMAS MODELIUOJAMOJO ATKAITINIMO IR KINTAMOS APLINKOS PAIEŠKOS METODAIS

L. Sakalauskas, G. Felinskas

Santrauka

Taikant informacines technologijas dažnai susiduriama su įvairiomis darbų ar užduočių tvarkaraščių sudarymo problemomis, kai ištekliai, reikalingi užduotims atlikti, yra riboti. Šiame straipsnyje yra nagrinėjami darbų tvarkaraščių sudarymo ir optimizavimo algoritmai, atsižvelgiant į išteklių, laiko bei kitokius ribojimus. Sukurtos ir tiriamos tvarkaraščio optimizavimo procedūros, paremtos tvarkaraščio kodavimu pagal darbų pirmumo sąrašą. Optimalus darbų pirmumo sąrašas randamas derinant lokaliosios ir globaliosios paieškos algoritmus – atsitiktinės paieškos ir modeliuojamojo atkaitinimo metodus su kintama aplinka. Pateikti skaičiavimo rezultatai, naudojant testų duomenis iš tvarkaraščių sudarymo uždavinių bibliotekos.

Reikšminiai žodžiai: ribotų išteklių projektai, tvarkaraščio optimizavimas, Monte Karlo metodas, modeliuojamasis atkaitinimas, kintamos aplinkos metodai.

Leonidas SAKALAUSKAS. Doctor Habil, Professor. Department of Operational Research. Institute of Mathematics and Informatics, Vilnius, Lithuania.

Doctor (1974), Kaunas University of Technology. Doctor Habil (2000), Institute of Mathematics and Informatics. Professor (2005). Research visits to International Centre of Theoretical Physics (ICTP) (Italy, 1996, 1998).

L. Sakalauskas is a member of the New-York Academy of Sciences (1997), vice-president of the Lithuanian Operational Research society (2001), Elected Member of the International Statistical Institute (2002), member of International Association of Official Statistic (2001), member of European Working Groups on Continuous Optimization, Financial Modelling and Multicriterial Decisions. Author of more than 100 scientific articles.

Research interests: continuous optimization, stochastic approximation, data mining, Monte-Carlo method, optimal design.

Gražvydas FELINSKAS. PhD student. Department of Operational Research, Institute of Mathematics and Informatics. First degree (higher education) – in mathematics and informatics, Šiauliai Pedagogical Institute (1996). Master of Science (1997). Teacher and Assistant at Department of Informatics, Šiauliai University. Head of young computer users school, established in Šiauliai University. Research visits to Jyvaskyla University (JSS) (Finland, 2002), Hamburg University, Institute of information systems (Germany, 2004). Author of about 8 scientific articles.

Research interests: continuous and discrete optimization, scheduling, stochastic heuristic search (Simulated annealing, Tabu search, genetic algorithms), extreme value theory, Monte-Carlo method, optimal design.