Agent-Based Gene Expression Programming for Solving the RCPSP/max Problem

Piotr Jędrzejowicz and Ewa Ratajczak-Ropel

Department of Information Systems
Gdynia Maritime University, Poland
{pj,ewra}@am.gdynia.pl

Abstract. The paper proposes combining a multi-agent system paradigm with the gene expression programming (GEP) to obtain solutions to the resource constrained project scheduling problem with time lags. The idea is to increase efficiency of the GEP algorithm through parallelization and distribution of the computational effort. The paper includes the problem formulation, the description of the proposed GEP algorithm and details of its implementation using the JABAT platform. To validate the approach computational experiment has been carried out. Its results confirm that the agent based gene expression programming can be considered as a promising tool for solving difficult combinatorial optimization problems.

1 Introduction

The paper proposes combining a multi-agent system paradigm with the gene expression programming (GEP) to obtain solutions to the resource constrained project scheduling problem with time lags (RCPSP/max). In recent years the RCPSP/max problem has attracted a lot of attention and many exact and heuristic algorithms have been proposed for solving it [2], [12], [16]. There have been several multi-agent approaches proposed to solve different types of optimization problems. One of them is the concept of an asynchronous team (A-Team), originally introduced in [15]. On the other hand the gene expression programming proposed in [6] as a method for solving different kinds of optimization problems seems to be a population-based approach, which can easily yield to parallelization and decentralization. GEP is a kind of evolutionary algorithm that evolves computer programs, which can take many forms: mathematical expressions, neural networks, decision trees, polynomial constructs, logical expressions, etc. GEP has been successfully used for solving many optimization and especially combinatorial optimization problems ([8], [14]).

In this paper we propose a JABAT-based implementation of GEP algorithm intended for solving instances of the RCPSP/max problem. JABAT is a multi-agent platform which has been designed as a tool enabling design of the A-Team solutions [10], [3]. The algorithm based on GEP was implemented as optimization agent in JABAT. The approach has been validated experimentally. Section 2 of the paper contains the RCPSP/max problem formulation. Section 3 gives some details on gene expression programing. Section 4 provides details of the GEP implementation for solving the RCPSP/max problem and describes fine-tuning phase of the proposed algorithm. Section 5 provides a brief description of the platform used to implement GEP algorithm as an A-Team. Section 6 describes computational experiment carried-out. Section 7 contains conclusions and suggestions for future research.

2 Problem Formulation

The resource-constrained project scheduling problem with minimal and maximal time lags (RCPSP/max) consists of a set of $n+2$ activities $V=\{0, 1, \ldots, n, n+1\}$, where each activity has to be processed without interruption to complete the project. The dummy activities 0 and $n+1$ represent the beginning and the end of the project. The duration of an activity j, $j = 1, \ldots, n$ is denoted by d_j where $d_0 = d_{n+1} = 0$. There are r renewable resource types. The availability of each resource type k in each time period is r_k units, $k = 1, \ldots, r$. Each activity j requires r_{jk} units of resource k during each period of its duration where $r_{1k} = r_{nk} = 0$, $k = 1, ..., r$. Each activity $j \in V$ has a start time s_j which is a decision variable. There are generalised precedence relations (temporal constraints) of the start-to-start type with time lags $s_j - s_i \geq \delta_{ij}$, $\delta_{ij} \in Z$, defined between the activities.

The structure of a project can be represented by an activity-on-node network $G = (V, A)$, where V is the set of activities and A is the set of precedence relationships. The objective is to find a schedule of activities starting times $S = [s_0, \ldots, s_{n+1}]$, where $s_0 = 0$ (project always begins at time zero) and resource constraints are satisfied, such that the schedule duration $T(S) = s_n + 1$ is minimized.

The RCPSP/max as an extension of the RCPSP belongs to the class of NP-hard optimization problems [1], [5]. The objective is to find a makespan minimal schedule that meets the constraints imposed by the precedence relations and the limited resource availabilities.

3 Gene Expression Programming

Gene expression programming (GEP) as a kind of the genetic programming algorithm was proposed by Candida Ferreira in [6]. The fundamental difference between gene expression programming (GEP), genetic algorithms (GAs) and genetic programming (GP) resides in the nature of the individuals. In GAs the individuals are linear strings of fixed length (chromosomes); in GP the individuals are nonlinear entities of different sizes and shapes (parse trees); and in GEP the individuals are encoded as linear strings of fixed length (the genome or chromosomes) which are afterwords expressed as nonlinear entities of different sizes and shapes (i.e., simple diagram representations or expression trees).

The genes in GEP have a special structure. The genome or chromosome consists of a linear, symbolic string of fixed length composed of one or more genes.

Each gene may be composed of two parts: the head and the tail or one part: the tail only. The head contains symbols that represent both functions and terminals, whereas the tail contains only terminals. GEP chromosomes are usually composed of more than one gene of equal length but different classes (multigenic chromosomes). For each problem, the number of genes, as well as the length of the head, are chosen a priori. It is important, that the genes must be built such that any modification made in them always results in a structurally correct genes. There are several special gene operators proposed for GEP.

Using GEP for solving different combinatorial optimization problems was discussed in [6] and [7], also for scheduling problems: traveling salesperson problem and task assignment problem. The main components of the GEP algorithm used for solving combinatorial optimization problems include:

- gene structure,
- population of individuals,
- fitness function and selection rules,
- reproduction operators.

In the combinatorial optimization problems the multigenic chromosomes are considered where each gene are composed of the tail only. In one chromosome different classes of genes may be used. Genes are not subject to any additional restrictions - a gene may be created as a set of any terminals combination from the set fixed for the class to which the gene belongs.

The chromosomes which are parts of individuals from the initial population are generated randomly. Most of them or even all may not represent correct solutions. The next generation is formed basing on a simple elitism rule - the best individual (or one of the best) from each generation is cloned into the next generation.

In GEP it is important to use suitable fitness function to control the fitness of solutions. To make the evolution efficient some different fitness functions were proposed in [7]. In most cases they are based on the absolute or relative error. For the scheduling problems, where minimal solution is the best, the function $f_x = T_g - t_x + 1$ was proposed, where x is the individual, t_x is the length of the schedule and T_g is the the length of the largest schedule encoded in the chromosomes of the current population. Individuals in GEP are selected according to fitness by the roulette-wheel sampling [9].

To reproduce generations of individuals in GEP except a simple elitism and the roulette-wheel sampling several class of operators are used. They are named as: insertion, transposition, recombination and mutation. For combinatorial optimization problems two additional classes of operators were proposed: deletion/insertion and permutation [7].

4 Using GEP for Solving RCPSP/max

The foundations of gene expression programming have been used to construct the algorithm for solving RCPSP/max. A high level pseudocode of this algorithm, denoted as GEP4RCPSPmax is presented in figure 1.

```
GEP4RCPSPmax(GenerationNumber, PopulationSize)
{
  Create initial population Pop of size PopulationSize
  for(i=1; i<GenerationNumber; i++)
  {
    Pop.mutationConstruct(0.02)
    Pop.mutationLC(0.08)
    Pop.geneDelIns(0.04)
    Pop.inversion(0.04)
    Pop.inversion2g(0.02)
    Pop.restrictedPermutation(0.06)
    Pop.onePointRecombination(0.6)
    Pop.twoPointRecombination(0.8)
    Pop.nPointRecombination(0.8)
    Pop.geneRecombination(0.5)
    Remember the best individual
    Create next generation
  }
}
```

Fig. 1. The pseudocode of GEP4RCPSPmax algorithm

Each chromosome representing the schedule consists of n+2 genes. Genes represent project activities in sequence from 0 to $n+1$. In each gene the information of activity starting time is coded as a non negative integer value. In each gene values from the same interval $c_j^{gep} \in [0, h]$, $j = 0, \ldots, n+1$ are stored. To calculate the real starting time for each activity the following formula is used: $c_j = c_j^e + c_j^{gep} \% (c_j^l - c_j^e)$, where c_j^e is the earliest, and c_j^l is the latest possible starting time for the activity j.

To calculate the fitness the function described in section 3 is used. The initial population of solution is generated randomly. Each individual is used to produce a solution and for each individual the fitness function value is calculated. In case of failure the fitness function is decreased by the duration of these activities which caused the failure.

Individuals for the next generation are selected by the roulette-wheel sampling with fitness as the criterion. The best individual from each generation is cloned to the next. To evolve population some classic GEP operators are used together with several of their modifications. The considered GEP operators are: gene deletion/insertion (geneDelIns), inversion, restricted permutation, one and two point recombination and gene recombination. The modified ones include: two genes inversion, n point recombination and mutation. All considered operators except mutationConstruct and mutationLC are used only once for one chromosome in one generation. Descriptions of these operators follows:

- mutationConstruct - the RCPSP/max SGSU (serial generation scheme with unscheduling) [11] algorithm is used for the randomly chosen individual;

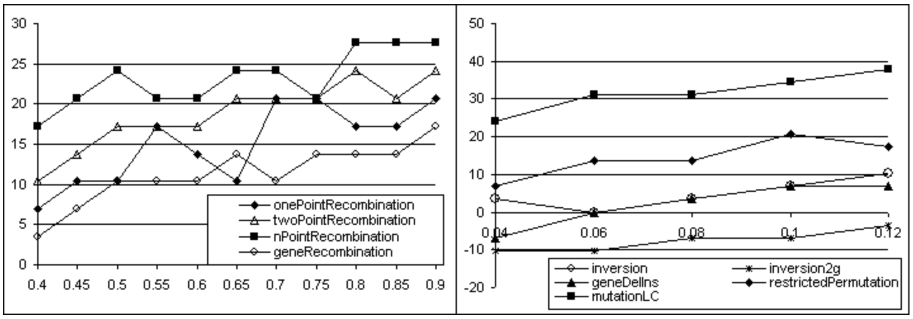

Fig. 2. Percentage of the maximum fitness for different probability values for operators from the crossover class (on the left) and the mutation class (on the right)

- mutationLC - the new gene value is randomly selected from the fixed set of values;
- geneDelIns - one individual is randomly selected from the population, the gene from chromosome is selected randomly and moved in the new randomly chosen place in the chromosome, the genes between the two selected places are moved as well;
- inversion - one individual is randomly selected from the population, two points from the chromosome are selected randomly and the sequence of genes between them is inverted;
- inversion2g - inversion where the sequences of only two genes are used;
- restrictedPermutation - one individual is randomly selected from the population, two genes from randomly selected positions of chromosome are replaced;
- onePointRecombination - in the considered case it is equivalent to the one point crossover operator;
- twoPointRecombination - in the considered case it is equivalent to the two point crossover operator;
- nPointRecombination - two individuals are randomly selected from the population, and the two childes are created in such a way that for each gene position the gene from one randomly selected parent is allowed to the first child and the gene from the other parent is allocated to the second child;
- geneRecombination - two genes from randomly selected position are swapped between two randomly selected individuals from the population.

To evaluate the effectiveness of the approach several fine-tuning computational experiments have been carried out. In the experiments three factors have been investigated:

- effectiveness of the different operators,
- relationships between different operators,
- number of generations needed.

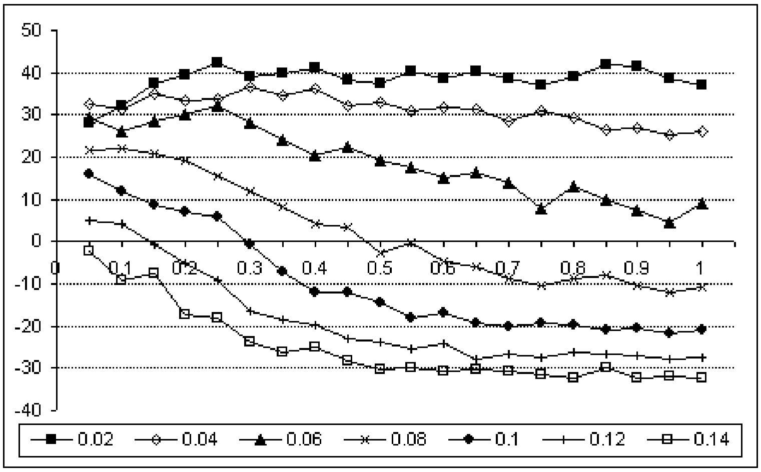

Fig. 3. Percentage of mean fitness for different probability values for all operators from the crossover class

In the fine-tuning experiments the considered operators have been divided into two broad classes: the mutation class and the crossover class. The percentage of maximum fitness obtained for different values of probability of applying various operators are shown in figure 2. The maximum fitness is understood as fitness value achived by the best solution. The experiment results suggest that the most effective operators for RCPSP/max are: nPointRecombination, twoPointRecombination, mutationLC and restrictedPermutation. In the final computational experiments higher probabilities for applying these operators have been used.

The next experiment investigates relationships between operators. Actually, relationships between the two considered classes have been considered, where each operator from the same class has the same probability value. The operators from mutation class have been investigated for probability values from 0.02 to 0.14, and operators from crossover class for probability values from 0.05 to 1. Results as the mean and maximum percentage fitness are presented in figure 3 and 4. As it is shown in these figures, probability values that maximize the mean fitness do not necessarily maximize the maximum fitness and vice versa. In the final experiment probability values have been set at the level in the middle of interval given by values maximizing the maximum fitness and the mean fitness respectively. The values used in the proposed GEP4RCPSPmax algorithm are shown in figure 1.

The last experiment within the fine-tuning stage aimed at investigating the influence of the number of generation evolved. In most cases to produce the best individual it has been enough to evolve 20 generations. However in case of inversion, restrictedPermutation and mutationLC the best individuals have been

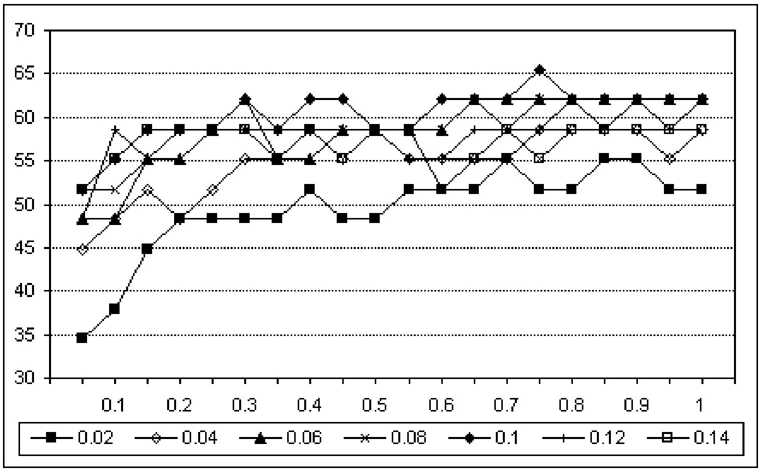

Fig. 4. Percentage of maximal fitness for different probability values for all operators from the crossover class

found after having evolved 80–90 generation. Thus in the final experiment the number of generations has been set to 100.

5 Implementing GEP4RCPSPmax Algorithm as an A-Team

To implement the proposed GEP algorithm as a team of agents the JABAT platform described in details in [10] [3], and [4] has been used. JABAT is a middleware allowing to design and implement A-Team architectures for solving various combinatorial optimization problems. The problem-solving paradigm on which the proposed system is based can be best defined as the population-based approach.

JABAT produces solutions to combinatorial optimization problems using a set of optimization agents, each representing a, so called, improvement algorithm. Each improvement algorithm when supplied with a potential solution to the problem at hand, tries to improve this solution. To escape getting trapped into the local optimum an initial population of solutions (individuals) is generated or constructed. Individuals forming an initial population are, at the following computation stages, improved by independently acting agents, thus increasing chances for reaching the global optimum.

JABAT may be used to implement the GEP4RCPSPmax algorithm in several ways. In case described in this paper the GEP4RCPSPmax algorithm has been implemented as an improvement algorithm embedded within the dedicated optimization agent. During computation several instances of this agent are used in parallel to speed up the process of improving individuals stored in the common memory.

6 Computational Experiment Results

To validate the proposed approach computational experiment has been carried out using benchmark instances of the RCPSP/max from PSPLIB [13]. The experiment involved computation with a fixed number of optimization agents used by the GEP4RCPSPmax algorithm. The respectie results are shown in table 1. The discussed results have been obtained using 5 optimization agents. At the beginning the population of solutions in the common memory of JABAT has been generated randomly. During computation solutions from common memory are drawn by optimization agents which try to improve their fitness through tuning the GEP4RCPSPmax algorithm. Each solution drawn by an optimization agent is incorporated into the population of 49 solutions generated internally by this agent. Such population is then evolved in 100 generations. If the best

Table 1. Experiment results for agent-based GEP4RCPSPmax

Number of activities	Mean RE	% FS	Mean CT
10	0.8 %	97.33 %	0.48 s
20	4.94 %	94.02 %	0.63 s
30	9.57 %	83.91 %	0.89 s

Table 2. Experiment results for ISES [2]

Number of activities	Mean RE	% FS	Mean CT
10	0.99 %	100.00 %	0.71 s
20	4.99 %	100.00 %	4.48 s
30	10.37 %	100.00 %	22.68 s

Table 3. Experiment results for B&B [2]

Number of activities	Mean RE	% FS	Mean CT
10	0.00 %	100.00 %	–
20	4.29 %	100.00 %	–
30	9.56 %	98.92 %	–

Table 4. Experiment results for C-BEST [2]

Number of activities	Mean RE	% FS	Mean CT
10	0.00 %	100.00 %	–
20	3.97 %	100.00 %	–
30	8.91 %	100.00 %	–

individual thus produced is an improvement as compared with the solution originally drown from the common memory then the improved solution replaces the worst individual in the JABAT common memory.

The computation results have been evaluated in terms of the mean relative error (Mean RE) calculated as the deviation from the lower bound, percent of feasible solutions (% FS) and mean computation time (Mean CT). The maximum computational time for one instance of the problem has, in all cases been not longer then 1 second during computations on 2 PC's with 2.0 GHz processors.

The results obtained by the proposed agent-based implementation of the GEP4RCPSPmax algorithm can be compared with these reported in the literature. Such a comparison are presented in tables 2, 3, and 4. It can be observed that the presented algorithm is efficient and the results are comparable. The advantege of the presented approach seems to be computation times, although they are not always easy to identify and there are some difficulties in comparing them precisely. The disadvantege is rather low percent of feasible solutions produced, which should be improved in future.

7 Conclusions

Experiment results show that the proposed agent-based GEP implementation is an effective tool for solving resource-constrained project scheduling problems with time lags. Presented results are comparable with the others known from the literature. It is worth noticing that the computation times are very short, even for more complex problem instances. Implementing such algorithm is quite simple, but some effort is needed to select GEP operators during the fine-tuning stage. The most important disadvantage of this algorithm is possible lack of solutions in some cases. This can be partly eliminated by using more then one instance of the algorithm.

Future research will concentrate on finding and testing more effective operators and eliminating a possibility of not finding a feasible solution at all. The other direction of research would be to using GEP algorithm in cooperation with other algorithms, like for instance local search, tabu search or simulated annealing in one agent-based system.

References

1. Bartusch, M., Mohring, R.H., Radermacher, F.J.: Scheduling project networks with resource constraints and time windows. Annual Operational Research 16, 201–240 (1988)
2. Cesta, A., Oddi, A., Smith, S.F.: A Constraint-Based Metod for Project Scheduling with Time Windows. Journal of Heuristics 8, 108–136 (2002)
3. Barbucha, D., Czarnowski, I., Jedrzejowicz, P., Ratajczak, E., Wierzbowska, I.: JADE-Based A-Team as a Tool for Implementing Population-Based Algorithms. In: Proc. VI Int. Conf. on Intelligent Systems Design and Applications, vol. 3, IEEE Computer Society, Los Alamitos (2006)

4. Barbucha, D., Czarnowski, I., Jedrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: e-JABAT–An Implementation of the Web-Based A-Team, Intelligent Agents in the Evolution of Web and Applications. Studies in Computational Intelligence, vol. 167, pp. 57–86. Springer, Heidelberg (2009)
5. Blazewicz, J., Lenstra, J., Rinnooy, A.: Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics 5, 11–24 (1983)
6. Ferreira, C.: Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. Complex Systems 13(2), 87–129 (2001), `http://www.gene-expression-programming.com/webpapers/Ferreira-CS2001/Introduction.htm`
7. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence, on-line book (2002), `http://www.gene-expression-programming.com/GepBook/Introduction.htm`
8. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence, 2nd edn. Springer, Heidelberg (2006)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
10. Jedrzejowicz, P., Wierzbowska, I.: JADE-Based A-Team Environment. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3993, pp. 719–726. Springer, Heidelberg (2006)
11. Neumann, K., Schwindt, C., Zimmermann, J.: Project Scheduling with Time Windows and Scarce Resources. Springer, Heidelberg (2002)
12. Neumann, K., Schwindt, C., Zimmermann, J.: Resource-Constrained project Scheduling with Time Windows. In: Recent developments and new applications, Perspectives in Modern Project Scheduling, pp. 375–407. Springer, Heidelberg (2006)
13. PSPLIB, `http://129.187.106.231/psplib`
14. Wilson, S.W.: Classifier Conditions Using Gene Expression Programming, IlliGAL Report No. 2008001, University of Illinois at Urbana-Champaign, USA (2008)
15. Talukdar, S., Baerentzen, L., Gove, A., de Souza, P.: Asynchronous Teams: Cooperation Schemes for Autonomous, Computer-Based Agents, Technical Report EDRC 18-59-96, Carnegie Mellon University, Pittsburgh (1996)
16. Valls, V., Ballestin, F., Barrios, A.: An evolutionary algorithm for the resource-constrained project scheduling problem subject to temporal constraints. In: Proc. of PMS 2006, Tenth International Workshop on Project Management and Scheduling, Nakom, Poznan, pp. 363–369 (2006)