Genetic Algorithm for Solving Software Project Scheduling Problems

Kalaouzis Stavros
Dept. of Production & Management Engineering, Democritus University of Thrace, Greece
stavros.kalaouzis@yahoo.gr

Introduction

What is the problem: The Project "TRACER (Project Scheduling Problem): TIC2002-04498-C05-02" is a Software Project Scheduling Problem, which consists in deciding who does what during the software project lifetime.

Why is important: This is a capital issue in the practice of software engineering, since the total budget and human resources involved must be managed optimally in order to end in a successful project. In short, companies are principally concerned with reducing the duration and cost of projects, and these two goals are in conflict with each other.

Problem Definitions

The set of skills required in the software project:

$$SK = \left\{ s_{k=1}, s_{k=2}, ..., s_{k=|S|} \right\}, k \in \{1, 2, ..., S\} \quad (1)$$

The set of tasks included in the software project:

$$TK = \left\{ t_{j=1}, t_{j=2}..., t_{j=|T|} \right\}, j \in \{1, 2, ..., T\} \quad (2)$$

Set of required skills of task: t^{skills}, Required dedication time of task: t^{effort}

The set of employees involved in the software project:

$$EM = \left\{ e_{i=1}, e_{i=2}..., e_{i=|E|} \right\}, i \in \{1, 2, ..., E\} \quad (3)$$

Set of skills of employee: e^{skills}, Salary of employee: e^{salary}, The maximum dedication of employee: e^{maxed}

Set of Vertex (task) for TPG in project Scheduling: $V = TK = \left\{ t_1, t_2, ..., t_{|T|} \right\}$

Set of arc (precedence relation between task): $A = \{(t_1, t_2), (t_2, t_V), ..., (t_m, t_T)\}$ $m, n \in \aleph^+$

The Project Graph in Project Scheduling: $G(V, A)$

Solution Matrix (Workload of Employee for each Task)

$$X_{E \cdot T} = \begin{Bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,T} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,T} \\ \vdots & \vdots & \ddots & \vdots \\ x_{E,1} & x_{E,2} & \cdots & x_{E,T} \end{Bmatrix} \quad (4)$$

Total Cost of Software Project:

$$p_{cost} = \sum_{i=1}^{E} \sum_{j=1}^{T} e_i^{salary} \cdot x_{ij} \cdot t_j^{dur} \quad (5)$$

Duration of task $task_j$: $t_j^{dur} = \frac{t_j^{effort}}{t_j^{ahr}}$

Workload of task $task_j$ which is in Project: $t_j^{ahr} = \sum_{i=1}^{E} x_{ij}$

The participation of employee e_i in the project:

$$e_i^{par} = \frac{\sum_{j=1}^{T} x_{ij} \cdot t_j^{dur}}{\sum_{j=1}^{T} t_j^{effort}} = \frac{\sum_{j=1}^{T} \frac{x_{ij}}{\sum_{k=1}^{E} x_{kj}} \cdot t_j^{effort}}{\sum_{j=1}^{T} t_j^{effort}}$$

Start time of task $task_j$: $t_j^{start} = \begin{cases} 0 & if \quad \nexists \ t_i, (t_i, t_j) \in A \\ max\left\{ t_i^{end} \right\} & otherwise \quad t_i, (t_i, t_j) \in A \end{cases}$

End time of task $task_j$: $t_j^{end} = t_j^{start} + t_j^{dur}$

Total Duration of Software Project:

$$p_{dur} = \max_{j=1}^{T} \left\{ t_j^{end} \right\} \quad (6)$$

$e_i^{over} = \int_{\tau=0}^{\tau=p_{dur}} ramp(e_i^{work} - e_i^{maxed}) d\tau$

$ramp(x) = \begin{cases} x & if \quad x > 0 \\ 0 & if \quad x \leq 0 \end{cases}$

Total overwork of Whole Software Project.

$$p_{over} = \sum_{i=1}^{E} e_i^{over} \quad (7)$$

Problem Constrains

Every task must involve at least one employee:

$$t_j^{ahr} > 0 \quad \forall j \in \{1, 2, ..., T\} \quad (8)$$

Employee involved into task should be able to supply all skill required by this task:

$$t_j^{skills} \subseteq \bigcup_{\{i | x_{ij} > 0\}} e_i^{skills} \quad \forall j \in \{1, 2, ..., T\} \quad (9)$$

Feasible solution should not make employee to work overtime:

$$p_{over} = \sum_{i=1}^{E} e_i^{over} = 0 \quad (10)$$

Each task satisfies the precedence constraint.

Installation and Usage

The source code in Matlab and compiled executables with an interactive interface are available at:
`https://github.com/skalaouzis/Genetic_Algorithm_for_Scheduling`

The command to execute the instance generator is:
> java pfc.ingsw.ProblemGenerator <configuration file> <output file>
The instance generated can be invalid, that is, it can not be found a solution for it. The first line of the output file is a comment line that indicates whether the instance is valid or not.

You can either make a local or a global installation.
- **Open** the folder `GeneticAlgorithmforScheduling`
- **Edit** the `.cfg` files in the folder `GeneticAlgorithmforScheduling`
- **Run** `GeneticAlgorithmforScheduling/main.m`

Algorithm and Results

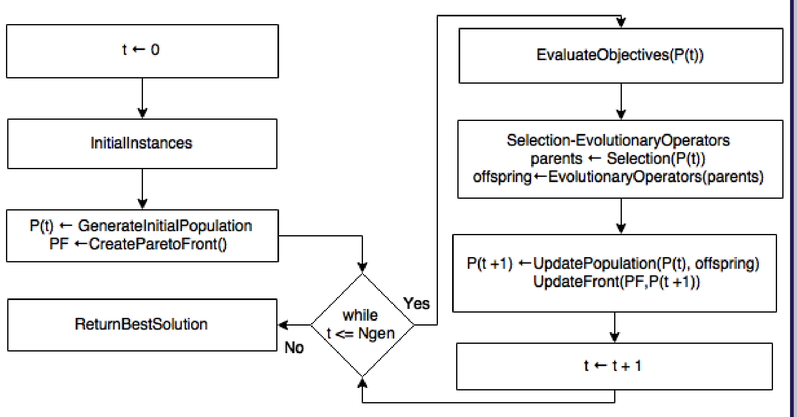


Fig. 1: Used Algorithm Flowchart

The results of "inst 30-10-10-5" are presented and this means: 30 tasks which have 1 to 10 required skills per task and exactly 10 employees which have 1 to 5 skills.

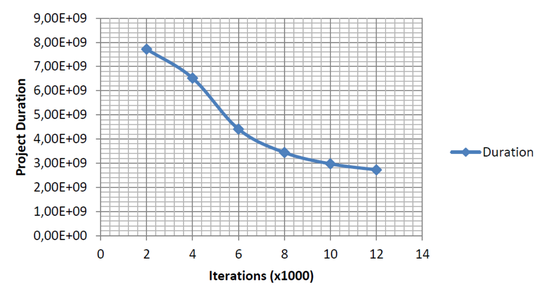


Fig. 2: Project Duration Minimization per Algorithm Iteration

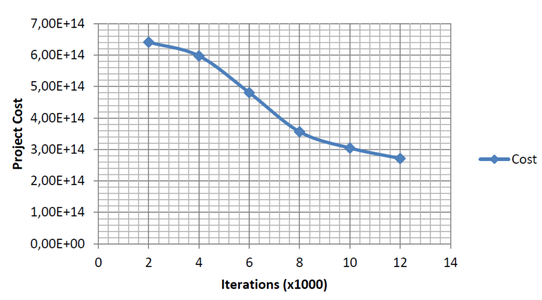


Fig. 3: Project Cost Minimization per Algorithm Iteration

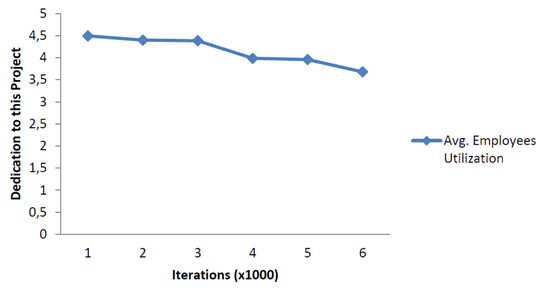


Fig. 4: Variance in Employees Utilization per Algorithm Iteration

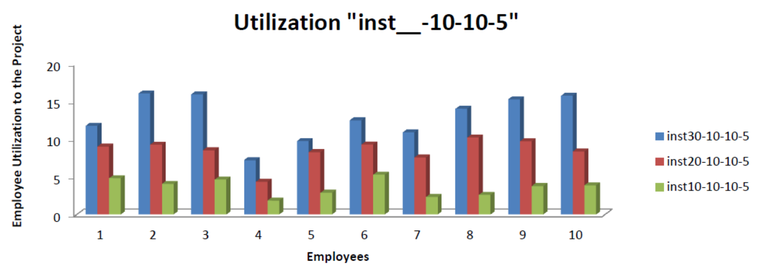


Fig. 5: Comparison of Final Employees Utilization per Instance

Feedback

- The results showed that algorithm may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem, when the highest ranking solutions fitness is reaching or has reached a plateau such that successive iterations no longer produce better results.
- Demonstrated that proposal gives the best results for smaller instances. For more complex instances was more difficult to find solutions, but these solutions always obtained a low cost of the project, in spite of increasing the duration of the whole project.
- SPSP is a problem that brings us closer to the estimation of software projects, but still has simplified features that should be changed to real cases and the problem is NP-hard by now. Due to its great complexity, only a few algorithms such as the genetic algorithms are available for the problem and the success rate of GA application to SPSP is still not satisfying.

Future Work

- In the future work, are planned to extend an new hybrid algorithm to deal with the multi-software project scheduling problem and the multi-mode resource constrained software project scheduling problem will also be investigated in future work.

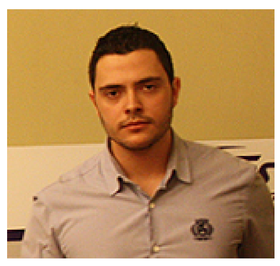
References

[1] The Project Scheduling Problem: *An Instance Generator for the Project Scheduling Problem*, `http://tracer.lcc.uma.es/problems/psp/`

[2] E. Alba, F. Chicano, (2007). *Software project management with gas.* Information Sciences, 177(1) 2380–2401.

[3] F. Chicano, F. Luna, A. J. Nebro, E. Alba, (2011). *Using multi-objective metaheuristics to solve the software project scheduling problem.* in: Proceedings of GECCO, pp. 1915–1922.

Biography



Stavros Kalaouzis
Diplom (Univ.) Production & Management Engineer - D.U.TH (Greece)
Software Developer at **Upwork Global Inc.**

RESEARCH INTERESTS

- Artificial Intelligence
- Operations Research
- Computational Mechanics
- Industrial Informatics

CONTACT

e-mail: stavros.kalaouzis@yahoo.gr
Phone: +306988620368
Website: skalaouzis.github.io