A GENETIC ALGORITHM FOR PROJECT SCHEDULING IN ACTIVITY NETWORKS UNDER RESOURCE COMPLEMENTARITY

Helder Silva

IFAM - Instituto Federal de Educação
Tecnológica do Amazonas
Manaus, Brazil
E-mail: helder@ifam.edu.br

José Oliveira
Anabela Tereso
Dept. Produção e Sistemas,
Universidade do Minho
Campus de Gualtar
4710-057 Braga – P, Portugal
E-mail: {zan|anabelat@dps.uminho.pt}

KEYWORDS
Project Management, Scheduling, Complementarity of resources.

ABSTRACT

We address the issue of optimal resource allocation, and more specifically, the analysis of complementarity of resources (primary resource or *P*-resource and supportive resource or *S*-resource) to activities in a project. The concept of complementarity can be incorporated into the engineering domain as an enhancement of the efficacy of a "primary" resource (*P*-resource) by adding to it other "supportive" resources (*S*-resources). We developed a Genetic Algorithm capable of determining the ideal mixture of resources allocated to the activities of a project, such that the project is completed with minimal cost. This problem has a circularity issue that greatly increases its complexity.
In this paper we present a constructive algorithm to build solutions from a chromosome that will be integrated in a Genetic Algorithm, which we illustrate by application to a small instance of the problem. The Genetic Algorithm is based on a random keys chromosome that is very easy to implement and allows using conventional genetic operators for combinatorial optimization problems. A project is formed by a set of activities. Each activity uses a specific set of resources, and it is also necessary to guarantee that there is no overlap in the time it takes to process activities in the same resource.

INTRODUCTION

This paper is concerned with the optimal resource allocation in activity networks under conditions of resource complementarity. The concept of complementarity which has been discussed from an economic point of view (Kremer, 1993) can be incorporated into the engineering domain as an enhancement of the efficacy of a “primary” resource (*P*-resource) by adding to it other “supportive” resources (*S*-resources). Aspects related to performance improvement, short duration, quality improvement have been presented by Silva et al. (2010) as well as the effect of the “supportive” resource for project cost.
We developed a mathematical model capable of determining the ideal mixture of resources allocated to the activities of a project, such that the project is completed with minimal cost (Silva et al. 2010; Silva et al. 2010b). This problem has a circularity issue that greatly increases its complexity. We have developed a procedure which we illustrate by application to small instances of the problem, using complete enumeration over the decision space (Silva et al. 2010b).
The optimal resource allocation in activity networks under conditions of resource complementarity is a generalization of the well known RCPSP which belongs to the NP-hard class (Brucker et al. 1998). This problem is a highly complex optimization problem due to its combinatorial nature, and thus an efficient algorithm for obtaining exact solutions is unknown. The development of a more computationally efficient procedure is now presented and is implemented in a Genetic Algorithm.

PROBLEM DESCRIPTION

Consider a project network in the activity-on-node (AoN) representation: $G=(N,A)$ with the set of nodes $|N|=n$ (representing the “activities”) and the set of arcs $|A|=m$ (representing the precedence relations between the activities). In general each activity requires the simultaneous use of several resources (Tereso et al. 2008; Tereso et al. 2009; Tereso et al. 2009b).
There is a set of “primary” resources, denoted by *P*, with $|P|=\rho$. Typically, a primary resource has a capacity of several units (say workers, m/c’s, processors; etc.) (Mulcahy 2005). Additionally, there is a pool of “supportive” resources, denoted by *S*, with $|S|=\sigma$ (such as less-skilled labor, or computers and electronic devices; etc.) that may be utilized in conjunction with the primary resources to enhance their performance. The number of supportive resources varies with the activity and the primary resources required for its execution. The impact on the *P*-resource is evaluated using a variable $0<v(r_p,s_q)\leq 1$ that indicates the fraction by which the *S*-resource s_q improves the performance of *P*-resource r_p. Typically, $v(r_p,s_q)\in[0.1,0.5]$. Consider $x_a(r_p)$ as the level of allocation of (primary) resource r_p to activity a, and $x_a\left(r_p,\{s_q\}_{q=1}^{\sigma}\right)$ as the total allocation of resource r_p (including complementary resources) to activity a.

We assume that the impact of the *S*-resources is additive: if a subset $\{s_q\}_{q=1}^{v}$ of the *S*-resources is used in support of *P*-resource r_p in activity *a*, and only one unit of each *S*-resource is used, then the performance of the former is enhanced to,

$$x_a\left(r_p,\{s_q\}_{q=1}^{\sigma}\right)=x_a\left(r_p\right)+\sum_{q=1}^{\sigma}v\left(r_p,s_q\right) \qquad (1)$$

With $w_a\left(r_p\right)$ representing the work content of activity *a* for *P*-resource *r*, the primary resource $r_p \in P$ would accomplish activity *a* in time $y_a\left(r_p\right)$ (see (2)). If it is enhanced by the addition of support resources, then its processing time decreases to $y_a\left(r_p,s_q\right)$ (see (3)).

$$y_a\left(r_p\right)=\frac{w_a\left(r_p\right)}{x_a\left(r_p\right)} \qquad (2)$$

$$y_a\left(r_p,s_q\right)=\frac{w_a\left(r_p\right)}{x_a\left(r_p,s_q\right)} \qquad (3)$$

An activity normally requires the simultaneous utilization of more than one *P*-resource for its execution. The problem then becomes: “At what level should each resource be utilized and which supportive resource(s) should be added to it (if any) in order to optimize a given objective?”
Recall that the processing time of an activity is given by the maximum of the durations that would result from a specific allocation to each resource (see a previous discussion on the evaluation of the duration considering multiple resources in (Tereso et al. 2008; Tereso et al. 2009; Tereso et al. 2009b)).

$$y\left(a\right)=\max_{all\ r_p}\left\{y_a\left(r_p\right)\right\} \qquad (4)$$

To better understand this representation, consider a simple project in AoN mode of representation (see Figure 1).

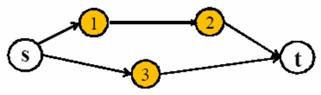

Figure 1: Project with 3 activities AoN

This project is formed by three activities, 1, 2 and 3, for which we will assume that it is required the utilization of four *P*-resources (in man-days); not all resources are required by all the activities (see Table 1).

Table 1: Work content of the activities of project

P-resource →	1	2	3	4
↓ Activity/Availability→	2	1	3	2
A1	16	0	12	12
A2	0	7	10	8
A3	20	0	22	0

The relevance and impact of the support resources on *P*-resources are represented in Table 2.

Table 2: The P-S matrix

	P-Resource →	1	2	3	4
↓*S*-Resource	↓ Availability				
1	1	0.25	ϕ	0.25	ϕ
2	2	0.15	0.35	ϕ	ϕ

We consider the cost of the resource utilization, a bonus for early completion, and a penalty for late completion of the project, after specifying a due date. A model was developed to minimize the total cost, considering that the activities should start as soon as they are sequence feasible (if there are enough primary resources to start them) (Silva et al. 2010). Some results were also reported using a procedure based on the analysis of the network and the concept of state space (Silva et al. 2010b).

GENETIC ALGORITHM

Since the Job Shop Scheduling Problem (JSSP) can be seen as a particular case of the Resource Constrained Project Scheduling Problem (RCPSP), we will extend a Genetic Algorithm (GA) developed for the JSSP (Oliveira et al. 2010) to the RCPSP, and particularly to the Project Scheduling in Activity Networks under Resource Complementarity. The GA is based on a random keys chromosome representation, that allows an easy reconfiguration to be applied in other problems.
The GA’s simplicity to model more complex problems and its easy integration with other optimization methods were factors that were considered before it was chosen. Initially, the algorithm proposed was conceived to solve the classical JSSP (Oliveira 2007), but it is possible to use the same method to solve other variants of the JSSP (Oliveira 2006), or in this case to solve a generalization of JSSP, that is the RCPSP with complementary resources.
One of the features that differentiates conventional genetic algorithms is the fact that the algorithm does not deal directly with the problem’s solutions, but with a solution representation - the chromosome. The algorithm manipulations are performed over the representation and not directly over the solution (Goldberg 1989).
We represent the project scheduling problem in a graph in AoN (Activity-on-Nodes) because it is similar to the disjunctive graph that is used to represent the JSSP (Roy and Sussmann, 1964). An activity can only be started if its predecessors are completed and if all the primary resources required are available. A project has a technological definition that determines a specific order to process some activities, and it is necessary to guarantee that there is no overlap in the time it takes to process such activities on the common resources. Considering this characteristic, we use the concept of the schedulable activity (an activity that could be started), and at each decision moment, it is only necessary to choose an activity from the set of schedulable activities. The choice of the activity is driven by the genetic algorithm attending to the alleles existent in the chromosome that give the priority of each activity.

Traditionally, genetic algorithms used bit string chromosomes. These chromosomes consisted of only '0s' and '1s.' Modern genetic algorithms more often use problem-specific chromosomes, as the balance between flexibility and raw efficiency tends away from the latter, and with evidence that use of real-valued chromosomes often outperformed bit string chromosomes anyway. Another alternative is the Gray code that is a binary numeral system where two successive values differ in only one digit (Goldberg 1989).
The permutation code was adequate for permutation problems. In this kind of representation, the chromosome is a literal translation of the operations sequence on the machines. In the classical JSSP case, the chromosome is composed by *m* sub-chromosomes, one for each machine, each one composed by *n* genes, one for each operation (Oliveira 2007). The *i* gene of the sub-chromosome corresponds to the operation processed in *i* place in the corresponding machine. The allele identifies the operation's index in the disjunctive graph (Roy and Sussmann, 1964).
For the Activity Networks under Resource Complementarity, we define a chromosome with $n(\rho+\sigma+1)$ genes. For each activity, the chromosome gives the quantity of each *P*-resource and the quantity of the complementary *S*-recourse. The chromosome also indicates the priority of each activity.
Nevertheless, in this work, the random key code presented by Bean (1994) is used. The important feature of random keys is that all offspring formed by crossover are feasible solutions, when it is used jointly with a constructive procedure based on the available operations to schedule and the priority is given by the random key allele. Through the dynamics of the genetic algorithm, the system learns the relationship between random key vectors and solutions with good objective function values. Another advantage of the random key representation is the possibility of using the conventional genetic operators. This characteristic allows the use of the genetic algorithm with other optimization problems, adapting only a few routines related with the problem.
A chromosome represents a solution to the problem and is encoded as a vector of random keys (random numbers). In this work, according to Cheng et al. (1996), the problem representation is indeed a mix from priority rule-based representation and random keys representation.
The genetic algorithm has a very simple structure and can be represented in the Algorithm 1. It begins with population generation and her evaluation. Attending to the fitness of the chromosomes the individuals are selected to be parents.

Algorithm 1: Genetic algorithm

```
begin
P ← GenerateInitialPopulation()
Evaluate(P)
while termination conditions not meet do
    P' ← Recombine(P)
    Evaluate(P')
    P ← Select(P ∪P')
end while
```

The crossover is applied and it generates a new temporary population that also is evaluated. Comparing the fitness of the new elements and of their progenitors the former population is updated.
The Uniform Crossover (UX) is used this work. This genetic operator uses a new sequence of random numbers and swaps both progenitors' alleles if the random key is greater than a prefixed value. Figure 2 illustrates the UX's application on two parents (prnt1, prnt2), and swaps alleles if the random key is greater or equal than 0.75. The genes 3, 4 and 12 are changed and it originates two descendants (dscndt1, dscndt2). Descendant 1 is similar to parent 1, because it has about 75% of genes of this parent.

gene	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***
prnt1	0,89	0,49	0,24	0,03	0,41	0,11	0,24	0,12	0,33	0,30	0,27	0,18
prnt2	0,83	0,41	0,40	0,04	0,29	0,35	0,38	0,01	0,42	0,32	0,28	0,13
randkey	0,64	0,72	0,75	0,83	0,26	0,56	0,28	0,31	0,09	0,11	0,37	0,76
dscndt1	0,89	0,49	0,40	0,04	0,41	0,11	0,24	0,12	0,33	0,30	0,27	0,13
dscndt2	0,83	0,41	0,24	0,03	0,29	0,35	0,38	0,01	0,42	0,32	0,28	0,18

Figure 2: The UX crossover

CONSTRUCTIVE ALGORITHM

The solutions are decoded by an algorithm, which is based on Giffler and Thompson's algorithm (GT) (Giffler and Thompson 1960). While the GT algorithm can generate all the active plans for the JSSP, the constructive algorithm only generates the plan in agreement with the chromosome. As advantages of this strategy, we have pointed out the minor dimension of solution space, which includes the optimum solution and the fact that it does not produce impossible or disinteresting solutions from the optimization point of view. On the other hand, since the dimensions between the representation space and the solution space are very different, this option can represent a problem because many chromosomes can represent the same solution.
The constructive algorithm has *n* stages and in each stage an activity is scheduled. To assist the algorithm's presentation, consider the following notation existing in stage *t*:
P_t - the partial schedule of the $(t-1)$ scheduled activities;
S_t - the set of activities schedulable at stage t, i.e. all the activities that must precede those in S_t are in P_t;
σ_k - the earliest time that activity a_k in S_t could be started. This time respects the conclusion of all predecessors of a_k and the availability of all resources that a_k will use (primary and supportive resources);
ϕ_k - the earliest time that activity a_k in S_t could be finished;
M^* - the set of resources used by a_k in S_t which has $\phi^* = \min_{a_k \in S_k}\{\phi_k\}$;
S_t^* - the conflict set formed by a_k in S_t which use at least one resource of M^* and $\sigma_j < \phi^*$;
A^* - the activity where $\varphi^* = \min_{a_k \in S_k}\{\varphi_k\}$;
S_t^* - the conflict set formed by $a_k \in S_t$ that have $\delta_k < \varphi^*$;
a_k^* - the selected activity to be scheduled at stage *t*.
The constructive algorithm of solutions is presented in a format, similar to the one used by Cheng et al. (1996) to

present the GT algorithm. In Step 3, instead of using a priority dispatching rule, the information given by the chromosome is used. If the maximum allele value is equal for two or more activities, one is chosen randomly.

Algorithm 2: Constructive algorithm

```
Step 1  Let $t=1$ with $P_1$ being null. $S_1$ will be
        the set of all activities with no
        predecessors; in other words those that
        are connected to start vetex.
Step 2  Find $\phi^* = \min_{a_k \in S_t}\{\phi_k\}$ and identify $A^*$.
        Form $S_t^*$.
Step 3  Select activity $a_k^*$ in $S_t^*$, with the
        greatest priority.
Step 4  Move to next stage by
            (1) adding $a_k^*$ to $P_t$, so creating $P_{t+1}$;
            (2) deleting $a_k^*$ from $S_t$ and creating
                $S_{t+1}$ by adding to $S_t$ the activities
                that directly follows $a_k^*$ and have
                all predecessors in $P_{t+1}$;
            (3) incrementing $t$ by 1.
Step 5  If there are any activities left
        unscheduled $(t<N)$, go to Step 2.
        Otherwise, stop.
```

Consider the example presented in Figure 1, Table 1, and Table 2 with three activities (A_1, A_2, A_3). In this instance, there are $\rho = 4$ primary resources and $\sigma = 2$ supportive resources. A chromosome to represent a solution for this instance has 21 genes, since there are six genes for each activity to establish the number of elements of each resource that will be used, plus a gene A_k that defines the activity's priority. Table 3 presents a chromosome of random keys values for this instance. The values are generated randomly between 0 and 99.

Table 3: A chromosome

A1	P1	P2	P3	P4	S1	S2	**A2**	P1	P2	P3	P4	S1	S2	**A3**	P1	P2	P3	P4	S1	S2
94	51	88	76	52	23	68	**36**	73	60	61	53	75	35	**47**	7	15	42	86	16	16

Activity A_1 has a priority 94, which is the greatest, while A_2 has the lowest priority (36). To define the number of elements of each resource, we use Table 1. The availability of *P*-resource 3 is 3 units. We define three equal intervals between 0 and 99. For this resource, if the allele is a value between 0 and 32 one unit is assigned. For values between 33 and 66, we assign two units, and for values between 67 and 99, three units will be assigned. To establish the number of units for the supportive resources, the procedure is similar, but it also includes the possibility to assign 0 units, because it is not required to use at least one unit. Considering these rules, the chromosome presented in Table 3 defines the following units of resources to be used, which are presented in Table 4.

Table 4: Amount of units of resources to be used

A1	P1	P2	P3	P4	S1	S2	**A2**	P1	P2	P3	P4	S1	S2	**A3**	P1	P2	P3	P4	S1	S2
Units	2	0	3	2	0	2	Units	0	1	2	2	1	1	Units	1	0	2	0	0	0

The procedure assigns zero units of a *P*-resource to an activity, if the activity does not use that primary resource, which is the case of P_2 in the activity A_1, according to Table 1. The assignment of supportive resources to the primary resources is performed considering the "amount" of Work content existent after the assignment of primary resources. The first unit is assigned to the *P*-resource with the largest Work content. After the assignment, the amount of work is recalculated, and then the next assignment is made, and so on. Activity A1 has the following Work content (see Table 1):

P-resource →	1	2	3	4
A1	16	0	12	12

Assigning the units of primary resources defined by the chromosome, the duration is then:

P-resource →	1	2	3	4
A1	8	0	4	6

The first unit of supportive resource S_2 is assigned to P_1. Recalculating the durations by (3), we have:

P-resource →	1	2	3	4
A1	7.442	0	4	6

The second unit of supportive resource S_2 is also assigned to P_1. Recalculating the durations by (3) we have:

P-resource →	1	2	3	4
A1	6.957	0	4	6

Applying the same procedure to the remaining activities, we will have the durations presented in Table 5.

Table 5: Activities duration

P-resource →	1	2	3	4
A1	6.957	0	4	6
A2	0	5.185	4.444	4
A3	20	0	11	0

NUMERICAL EXAMPLE

Figure 3 illustrates the results of applying the constructive algorithm. It presents the evolution of the schedulable set $\overline{S}$ and the corresponding staring and conclusion times for each activity during the execution of the constructive algorithm. The final Gant Chart of the project is also presented, and it shows the occupation of all the resource units.

S_1	1	3
δ_k	0	0
φ_k	6.957	20
a_k^*	1	

S_2	2	3
δ_k	6.957	6.957
φ_k	12,14	26,96
a_k^*		3

S_3	2
δ_k	17,97
φ_k	23,14
a_k^*	2

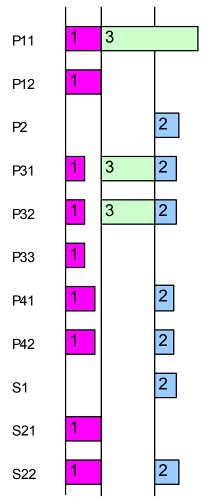

Figure 2: Constructive Algorithm execution

Consider the due date for the project equal to 24 units of time and the following parameters:

• each unit of *P*-resource costs 4 per unit of Work content;
• each unit of *S*-resource costs 1 per unit of Work content;
• the delay costs 60 per unit of time;
• the earliness costs 40 per unit of time.

The project is complete at time 26.96 with 2.96 units of lateness. The resources cost is equal to 845, and the delay cost is 177.39. The total cost of the solution is 1022.39.

CONCLUSIONS

The goal of this paper was to provide a formal model to some unresolved issues in the management of projects, especially as related to the utilization of supportive resources, and to its implementation. The relevance of the problem is the opportunity to shape a system that allows not only for improvement in the allocation of often scarce resource(s), but also results in reduced uncertainties within the projects, combined with increased performance and lower project costs. The model was first presented in (Silva et al. 2010), but it still needed to be implemented and applied to some project networks to demonstrate its validity. We presented the procedure developed to solve the mathematical model, and we applied it to two simple networks, obtaining the desired results through an initial implementation in C (Silva et al. 2010b).

Considering the feasibility of the model proposed, we believe it can provide the user a new option to plan and determine the best combination of resources and the lowest project cost, pushing the planning phase and increasing the estimation ability of the companies.

This paper presents a structure to implement a genetic algorithm to solve the Project Scheduling in Activity Networks under Resource Complementarity. The schedules are built using information given by the genetic algorithm to order the activities. We presented an example of the application of the model, and we achieved preliminary results for a small example. In future work, we intend to test this approach on a set of instances taken from the literature of RCPSP, which will be modified to accommodate the supportive resources.

REFERENCES

Bean, J.C. 1994. “Genetics and random keys for sequencing and optimization.” *ORSA Journal on Computing*, 6, 154–160.

Cheng, R.; M. Gen; and Y. Tsujimura. 1996. “A tutorial survey of job-shop scheduling problems using genetic algorithms - I. Representation.” *Computers & Industrial Engineering*, 30, 983-997.

Giffler, B.; and G.L. Thompson. 1960. “Algorithms for solving production scheduling problems.” *Operations Research*, 8, 487-503.

Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Kremer, M. 1993. “The O-Ring Theory of Economic Development.” *The Quartely Journal of Economics*, 108, 551-575.

Mulcahy, R. 2005. *PMP Exam Prep, Fifth Edition: Rita's Course in a Book for Passing the PMP Exam*. RMC Publications, Inc.

Oliveira, J.A.; L. Dias; and G. Pereira. 2010. “Solving the Job Shop Problem with a random keys genetic algorithm with instance parameters, In *Proceedings of the 2nd International Conference on Engineering Optimization*. Lisbon, Portugal.

Oliveira, J.A. 2007. “Scheduling the truckload operations in automatic warehouses.” *European Journal of Operational Research*, 179, 3, 723-735.

Oliveira, J.A. 2006. “A genetic algorithm with a quasi-local search for the job shop problem with recirculation”. *Applied Soft Computing Technologies: The Challenge of Complexity*. Springer, Berlin / Heidelberg, 221-234.

Roy, B.; and B. Sussmann. 1964. “Les problemes d’ordonnancement avec constraintes disjonctives.” Note DS, No. 9 Bis, SEMA, Paris.

Silva, H.C.; A.P. Tereso; and J.A. Oliveira. 2010. “On Resource Complementarity.” In *Proceedings of the 3rd International Conference on Information Systems*, Logistics and Supply Chain. Casablanca.

Silva, H.C.; A.P. Tereso; and J.A. Oliveira. 2010b. “On Resource Complementarity in Activity Networks – Further Results”. In *Proceedings of the 2nd International Conference on Engineering Optimization*, Lisbon, Portugal.

Tereso, A.; M. Araújo; R. Moutinho; and S. Elmaghraby. 2008. “Project management: multiple resources allocation.” In *Proceedings of the International Conference on Engineering Optimization*. Rio de Janeiro, Brazil.

Tereso, A.; M. Araújo; R. Moutinho; and S. Elmaghraby. 2009. “Duration Oriented Resource Allocation Strategy on Multiple Resources Projects under Stochastic Conditions,” In *Proceedings of the International Conference on Industrial Engineering and Systems Management*. Montreal, Canada.

Tereso, A.; M. Araújo; R. Moutinho; and S. Elmaghraby. 2009b. “Quantity Oriented Resource Allocation Strategy on Multiple Resources Projects under Stochastic Conditions.” In *Proceedings of the International Conference on Industrial Engineering and Systems Management*. Montreal, Canada.

BIOGRAPHY

HELDER SILVA was born in Minas Gerais, Brazil and went to the University of Uberlândia, where he studied Electrical Engineering and obtained his degrees in 1998. He holds an MSc in the same area (2001) and nowadays he is doing a PhD at University of Minho in Portugal. Professor of some universities in Brazil (UNIP, IDAAM and IFAM), he has several international publications related to Project Management, Resource Allocation in Projects, Complementarity and Project Cost Optimization. He is currently the Coordinator of Project Management Office in a Chinese Worldwide Company in Brazil.

JOSÉ A. OLIVEIRA was born 1966 in Matosinhos, Portugal. He studied Mechanical Engineering at the University of Porto, Portugal. He graduated with a Ph.D. in Production and Systems Engineering at University of Minho, Portugal. His main research interests are Optimization with Heuristic Methods in Systems Engineering.

ANABELA TERESO was born in Cantanhede, Portugal. She has a degree in Systems and Informatics Engineering (1990), an MSc in Informatics (1997) and a PhD in Production and Systems Engineering (2002) all from University of Minho – Portugal. She is Professor at University of Minho since 1995, and does research in the area of Project Management since her PhD.