EAI Endorsed Transactions
on Energy Web
Research Article EAI.EU

Estimation of Distribution Algorithm for solving the Multi-mode Resource Constrained Project Scheduling Problem

Gaafar Sadeq S. Mahdi[1,*], Julio Madera Quintana[2], Pedro Piñero Pérez[1] and Salah Hassan Al-subhi[1]

[1]Research Laboratory in Project Management, University of Informatics Sciences, San Antonio km 2 ½, CP 17830, La Habana, Cuba
[2]Department of Computer Science, Camagüey University, Camagüey, Cuba

Abstract

The Multi-mode Resource Constrained Project Scheduling Problem is characterized by a set of tasks, resources and an objective function. All tasks of a project must be organized carefully taking into account precedence relations, the mode in which they are performed and availability of resources at all times. At present, around 71% of the projects related to the software industry are renegotiated or canceled causing negative impacts on both, social and economic areas. Among the root causes of these failures, deficiencies in planning processes and a lack of tools to help generate quasi-optimal project schedules are found. This kind of problem can be presented as an optimization problem subjected to two groups of restrictions: precedence relations and resource constraints. This paper aims at proposing a new Estimation of Distribution Algorithm applied for the resolution of the Multi-mode Resource Constrained Project Scheduling Problem. In particular, this algorithm is based on Factorized Distribution Algorithm in which the precedence relations of the problem are represented by the factorization. A comprehensive computational experiment is described, performed on a set of benchmark instances of the well-known Project Scheduling Problem Library (PSPLIB) in its Multi-mode variant. The results show that the proposed algorithm can find similar or sometimes even shorter makespans than others reported in bibliography.

Keywords: Multi-mode Resource Constrained Project Scheduling Problem, Estimation of Distribution Algorithm, Optimization.

Received on 28 February 2020, accepted on 17 April 2020, published on 24 April 2020

doi: 10.4108/eai.13-7-2018.164111

1. Introduction

The project management keeps on a growing trend during recent years. Generally, investment processes are organized as projects with a high impact on society and economy. In this sense, factors such as technological changes, economic pressures, work in multidisciplinary teams, limited resources and time, are essential elements that must be carefully coordinated to achieve project objectives with an adequate balance between costs and time [1].

Within this context, the planning and construction process of optimal or quasi-optimal project schedules is a constant concern of the project managers. It is generally identified that the Project Scheduling Problem (PSP) is a complex problem of the NP-hard type [2], where numerous factors and variables are involved.

The difficulties of planning in project management continue and, for example, in the field of software development, they are clearly manifested in the CHAOS Reports of the Standish Group [3], [4]. These studies show that, in the last years, the software industry projects have behaved as follows:

- The number of unsatisfactory projects remains in a high range between 66% and 71%.
- The root causes of project failure [5] can be frequent errors in the planning processes, scope, time or logistics [6], [7], [8]. Furthermore, poor management of control and monitoring processes [9], [10] causes delays and quality problems.

*Corresponding author. Email: gaafarsadeq@yahoo.com

This situation can be mitigated by actions aimed at improving the quality of planning, supported in the construction processes of project schedules.

There are several schools that try to standardize the concepts and practices associated with project management and enhance the construction of project plans adjusted to the needs of the projects. Some of these institutions are the Software Engineering Institute with its CMMI proposal [11], [12], the Project Management Body of Knowledge (PMBOK) [13], and the ISO 21500 standard developed by the International Standards Organization ISO [14], [15].

These schools propose new techniques and in their recent versions, they have introduced the need for simulation techniques, data analysis and resource optimization in the construction of project schedules. However, some difficulties still persist, such as:

- They do not propose concrete optimization techniques for the construction of project schedules.
- They explain the need to consider restrictions on the availability of resources, but do not take into account restrictions related to the competences of human resources or other specific characteristics of resources that influence the duration of the tasks of a project.
- PERT and CPM classic techniques for project scheduling do not explicitly consider the allocation of resources to tasks, but rather constitute tools to help graph and analyze the schedules once they have been built [16].

The aim of this work is to present the Constraint-based Learning Factorization of Distribution Algorithm (CL_{FDA}) for the resolution of the Multi-mode Resource Constrained Project Scheduling Problem (MMRCPSP). The experiments were done on a set of benchmark instances of the Project Scheduling Problem Library (PSPLIB) in its Multi-mode variant.

The rest of the paper is structured as follows: In section 2, a background to the study is presented where the different categories of PSP and formulate the Multi-mode Resource Constrained Project Scheduling Problem (MMRCPSP) are explained. Section 3 provides the algorithm proposed, while results and analysis are described in Section 4. Section 5 presents the conclusions and suggests directions for future research.

2. Project Scheduling Problems

Four theoretical PSP are listed below:

Resource Constrained Project Scheduling Problem (RCPSP)

It consists of establishing the sequence of a set of tasks of a single project, subjected to two types of constraints, precedence relations and the number of resources available to perform the tasks in every moment. In this problem, the objective is to minimize the makespan of the project [17]. A limitation of this problem is that it does not distinguish between the particularities of these resources that can significantly influence the duration of the tasks.

Resource Constrained Multi-project Scheduling Problem (RCMPSP)

It is a generalized case of the RCPSP where it is desired to develop multiple project schedules simultaneously with limited resources [18]. Unlike the previous case, a new variable emerges that is the priority among projects for the consumption of the resources.

Multi-mode Resource Constrained Project Scheduling Problem (MMRCPSP)

It is an extension of the RCPSP that involves the selection of a performance mode for each task, where each mode is associated with a duration and a quantity of renewable and non-renewable resources required for the performance of that task. Moreover, this problem takes into account restrictions such as the precedence between tasks and the availability of resources. The use of modes helps to identify, for example, that similar resources, but with different characteristics, can have a significant impact on the duration of the tasks [19], [20].

Multi-mode Resource Constrained Multi-project Scheduling Problem (MMRCMPSP)

It combines the two previous concepts [21].

Considering the complexity involved in solving these problems, many authors make use of soft computing methods, especially meta-heuristics techniques [2, 21]. Some authors [22], [23] propose algorithms based on Particle Swarm Optimization (PSO), others propose techniques based on Tabu Search [24], [25], while the most used meta-heuristic is the based on Genetic Algorithms (GA) [26], [27], [16], [28].

Ayodele [29][30] and collaborators apply an Estimation of Distribution Algorithm (EDA), but based on static learning, where new individuals are generated from exploring the most promising areas in the search space, based on the distribution of the best individuals of the previous generation. In this algorithm, a solution is coded using mode assignment and a list of tasks.

These algorithms implement known mechanisms such as, data pre-processing, instance generation using Sequence Generator Scheme (SGS), and a penalty function for the violation of restrictions associated with the availability of resources.

While these metaheuristics provide solutions to simple PSP, in the MMRCPSP case, they do not adequately handle the work with restrictions, especially they fail in problems with a high level of complexity. This fact means that the solutions obtained are not the best regarding the optimization of the execution time of the projects. In addition to this, they do not handle the difference between human resources competencies to execute each type of task, an element that significantly influences the execution time and the cost of the projects.

2.1. MMRCPSP Modelling

Multi-mode Resource Constrained Project Scheduling Problem (MMRCPSP) is a generalized version of RCPSP. The term multimode indicates that project tasks can be carried out in different ways (modes); each mode has a specific duration and corresponds to a certain number of resources. Due to this approach, planners take into account situations in which, for example, incorporating new resources for a task with the objective of minimizing its duration. So, the computational time required to solve the MMRCPSP problem is much more than the one needed in RCPSP.

A project contains a set of $j \in J$ tasks. These tasks have a set $M = \{1, \dots, m\}$ processing modes. A task j, performed in mode m takes a period of time d_{jm}, needs a certain number of resources R and N (renewable and non-renewable).

The first mathematical formulation of the MMRCPSP problem that took into account non-renewable resources was presented by Talbot in [31]. He proposes a linear model with binary variables and defines the variables as follows: $x_{jmt} = 1$ if the task j has started in mode m and is completed in period $t \in [EF_j, LF_j]$, and 0 in other cases. The parameters are summarized in Table 1. See [32] For more understanding of these parameters.

Table 1. Notation and definitions

J	Set of tasks
M_j	Number of modes task j can be performed in
d_{jm}	Duration of task j being performed in mode m
R / N	Set of renewable/ nonrenewable resources
T	Upper bound on the project's makespan
K_r^ρ	Number of units of renewable resource r, $r \in R$, available in period t, $t = 1 \dots T$
K_r^v	Number of units of non-renewable resource r, $r \in N$
K_{jmr}^ρ	Number of units of renewable resource r, $r \in R$, consumed by task j being performed in mode m
K_{jmr}^v	Number of units of non-renewable resource r, $r \in N$ used by task j being performed in mode m
H_j	Set of immediate predecessors of task j
ES_j	Earliest start time of task j
EF_j	Earliest finish time of task j, calculated by using minimal task durations and ignoring resource consumption
LS_j	Latest start time of task j, calculated by using minimal task durations taking into account the upper bound T on the project's duration
LF_j	Latest finish time of task j, calculated by using minimal task durations and ignoring resource usage

The formulation of this problem following a linear model in integers is represented as follows:

$$minimize \sum_{m \in M} \sum_{t=EF_j}^{LF_j} t x_{jmt} \qquad (2.1.1)$$

Where the completion time t of all tasks is minimized (minimization of the project's makespan), subjected to the following restrictions:

$$\sum_{m \in M} \sum_{t=EF_j}^{LF_j} x_{jmt} = 1 \qquad \forall j \in J \qquad (2.1.2)$$

Equation 2.1.2 controls that all tasks are accomplished in a processing mode and at some point during the progress of the project.

$$\sum_{m \in M} \sum_{t=EF_h}^{LF_h} t \cdot x_{hmt} \leq \sum_{m \in M} \sum_{t=EF_j}^{LF_j} (t - d_{jm})\, x_{jmt} \quad \forall (h,j) \in J, h \in H \qquad (2.1.3)$$

Restriction 2.1.3 ensures that the precedence relations are taken into account, where H is the set of predecessors of task j.

$$\sum_{j \in J} \sum_{m \in M} K_{jmr}^\rho \sum_{q=max\{t, EF_j\}}^{min\{t+d_{jm}-1, LF_j\}} x_{jmq} \leq K_r^\rho \quad \forall r \in R, t = 1, \dots, T \qquad (2.1.4)$$

Equation 2.1.4 guarantees that the per-period availabilities of the renewable resources are not exceeded. The value of T is upper bound, in days, on the project's makespan.

$$\sum_{j \in J} \sum_{m \in M} K_{jmr}^v \sum_{t=EF_j}^{LF_j} x_{jmt} \leq K_r^v \qquad \forall r \in N \qquad (2.1.5)$$

Finally, restriction 2.1.5 is the one that controls that the number of consumable (non-renewable) resources available is not exceeded.

3. Design of EDA for the resolution of the MMRCPSP

The Estimation of Distribution Algorithms (EDA) has been developed in 1996 by the authors Muehlenbein, Mahnig and Ochoa [33]. In general, EDA constitutes a family of algorithms to solve various optimization problems [34] and arises as an alternative to the difficulties of Genetic Algorithms (GA). These difficulties are associated with the fact that GA, by their nature, do not explicitly express the interdependencies between the variables of the problem and do not use this information sufficiently during the search process [35], [36]. The main characteristic of EDA is the identification of probabilistic distribution functions that model the dependency relations among the variables of the

problem to be solved and the generation of new individuals from that distribution.

This section presents CL_{FDA} which constitutes an EDA algorithm. It includes the constraint handling inside the probabilistic model, for the resolution of the MMRCPSP problem presented in the previous section. A brief analysis is carried out that shows the differences between the CL_{FDA} (Algorithm 2) and an EDA in its classical form (Algorithm 1).

Algorithm 1. Pseudo-code of an EDA algorithm in its classical form

1. *t ← 1*
2. *R = load constraints*
3. *p(t) = Initialize Population*
4. *Evaluate Population (p(t), R)*
5. *WHILE the stop criterion does not be met, DO*
   a. *p^S(t) = Select Individuals (p(t))*
   b. *gp = Estimate Distribution (p^S(t))*
   c. *p(t + 1) = Generate new individuals from (gp)*
   d. *Evaluate Population (p(t + 1), R)*
   e. *t ←t + 1*
6. *END WHILE*
7. *Return p(t)*

The fundamental characteristics of CL_{FDA} can be defined as follows:

- Estimation of the probabilistic model, inspired by the FDA algorithm [37], which describes the dependency relations among variables of the selected individuals.
- Construction of the probabilistic model by considering the problem constraints.

Algorithm 2. The pseudo-code of the algorithm proposed in this work

1. *t ← 1*
2. *R = load_constraints*
3. *p(t) = Initialize Population (R)*
4. *Evaluate Population (p(t), R)*
5. *WHILE the stop condition is not met, DO*
   a. *p^S(t) = Select Individuals (p(t))*
   b. *gp = Estimate Distribution (p^S(t))*
   c. *gp^R = Incorporate restrictions to the model (gp)*
   d. *p(t + 1) ←Best elitism (p^S(t))*
   e. *p(t + 1) ← Generate new individuals from (gp^R)*
   f. *Evaluate Population (p(t + 1), R)*
   g. *t ← t + 1*
6. *END WHILE*
7. *Return p(t)*

A detailed explanation of the proposed model is provided below:

(i) The constraints of the optimization problem are presented as follows:
- Precedence restrictions, where are known, from each task its successor tasks.
- Restrictions associated with the number of renewable and non-renewable resources available for project accomplishment.
- Each task mode contains the duration and constraints of renewable and non-renewable resources.

(ii) *Population definition*: each population has a fixed size specified as a parameter. The initial population is randomly generated.

(iii) *Design of the individual*: each individual constitutes a possible solution to the scheduling problem and is formed by a sequence of tasks (see Figure 1). Each task has two relevant features: start date of the task *(s)* and the mode *(m)* it was performed in. In addition, there is a set of complementary attributes of each task that are estimated from the mode and the start date. For example, the closing date of a task can be calculated from the starting date and the mode duration of the task.

(iv) *Evaluation method and objective function:* individuals are evaluated taking into account the objective function described in equation 3.1, where:
- $i \in I$ represent the individuals of each population.
- f_j represents the final day of task j.
- f_{ij} represents the closing date of task j of individual i.
- $cost_{ij}$ represents the cost of carrying out the task j of the individual i calculated from the sum of the costs of the resources associated with that task.

(v) *Definition of the selection strategy:* the selection method is based on *Pareto optimization* [38] or ranking, considering the cost and time objectives. The selection process is carried out by iterations until an amount equal to 30% of the population size.

(vi) *Elitism:* it was applied the best elitism.

(vii) *Definition of the stop condition:* it finds the optimum or reach the maximum number of generations.

The objective function realizes the minimization of the project's makespan and cost.

$$\text{Minimize } (O_1, O_2) \tag{3.1}$$

Where:

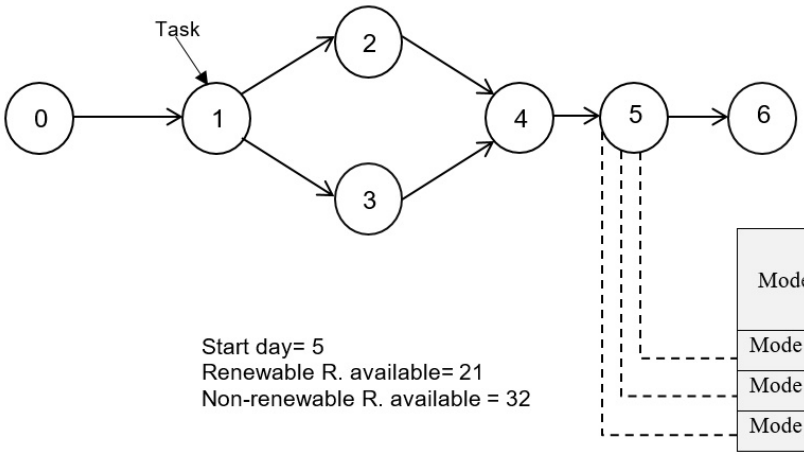
$$O_1 = \sum_{j \in J} cost_{ij} \tag{3.2}$$

$$O_2 = \max_{j \in J} f_j \tag{3.3}$$

Subjected to the following restrictions:

$$\sum_{j \in J} g(r_k, j) \leq R_k \qquad \forall k \in K_{nv} \tag{3.4}$$

$$\sum_{j \in J} g(r_k, j, t) \leq 1 \qquad \forall k \in K_v, t = 1, \dots, T \tag{3.5}$$



Mode	Required Resources		Duration	Completion day
	Renewable	Non-renewable		
Mode 1	13	5	7	11
Mode 2	10	4	8	12
Mode 3	7	3	11	15

Figure 1. MMRCPSP proposal model

$$s_j \leq f_j \qquad \forall j \in J \qquad (3.6)$$

$$f_h \leq s_j \qquad \forall h \in H, \forall j \in J \qquad (3.7)$$

Equation 3.4 presents the restriction that ensures that the use of available material (non-renewable) resources is not exceeded. Where $g(r_k, j)$ represents the number of times the resource $k \in K_{nv}$ is being used considering all the tasks $j \in J$ during the project progress.

Equation 3.5 presents the restriction that guarantees the per-period availabilities of human resources and equipment r_k at time t for the execution of task $j \in J$ (resource $k \in K_v$ is not shared for more than one task at the same time t). $g(r_k, j, t)$ represents the number of times resource k is being used considering all tasks $j \in J$ at one time t.

Equation 3.6 represents the restriction that the starting task moment j is always less than that of its completion day.

Equation 3.7 represents the restriction that ensures that the precedence relations among tasks are not violated, where h is the set of the predecessor tasks of j.

4. Experimental results and discussion

This section presents the results of the application of the proposed algorithm in solving the Multi-mode Resource Constrained Project Scheduling Problem (MMRCPSP). The main objective is to minimize the duration of the project, taking into account restrictions associated with the precedence among tasks and restrictions on resource constraints, whether renewable or nonrenewable.

The experiments were performed on the datasets "j30_17.mm", "c15_9.mm", "c15_10.mm" and "c15_12.mm" from the PSPLIB repository (Project Scheduling Problem Library) [32], [39] in its variant of multiple modes. The number of tasks, resources, modes and constraints have been selected to present a diverse set of problems. Each dataset consists of ten instances with two types of renewable resources and two non-renewable resources. The number of tasks varies between 16 and 30, and the number of modes is three for all instances.

The PSPLIB library available at http://www.om-db.wi.tum.de/psplib contains sets of instances that represent different categories of scheduling problems. These instances were generated using the ProGen generator. In addition, PSPLIB presents for each instance, the optimal solution and the best solution reached by different authors so far. Datasets can be used by researchers to evaluate their procedures for solving scheduling problems.

The general format of the file proposed by PSPLIB as a problem instance is described below:

- Number of tasks (*jobs*).
- Number of modes in which each task can be executed (*#modes*).
- Number of types of renewable resources existing in the problem (*renewable*).
- Maximum availability of each type of renewable resources (*RESOURCEAVAILABILITIES*).
- Number of types of non-renewable resources existing in the problem (*nonrenewable*).
- Number of successors of each activity (*#successors*).
- Set of successor tasks of each task (successors).
- Duration of each task (*completion time*).
- The minimum possible time to carry out the project (*duedate*), which represents the optimum makespan value to be achieved.

In the experiments, the results of the CL_{FDA} algorithm are compared with:

- A genetic algorithm (GA).
- A Univariate Marginal Distribution Algorithm (UMDA).
- The optimal makespan to reach (Optimum).
- The best results reported in the PSPLIB repository (Reported_Bibliography).

The validation stage used two optimization strategies crucial to this study. The first strategy is called *just_time* where the optimization process is based only on the time objective, and therefore the objective function only evaluates the solutions with respect to the makespan variable. While the second optimization strategy is based on the Pareto Optimal (*pareto*) approach and is oriented simultaneously on the time (makespan) and cost objectives, in this case, the optimization of plans based on equation (3.1) was used.

For a fair comparison, the following common parameters are established for all the algorithms:

(i) The initial population was built randomly with a size of 100 individuals.
(ii) Thirty percent of the individuals were selected for the generation of the new population.
(iii) Elitism was applied.
(iv) In the case of the GA, the crossover probability is 0.8, while the mutation probability is 0.2.
(v) The stop condition was to reach 100 generations.

The experiments have been performed by running each algorithm 20 times for every instance of the four datasets. The results of the algorithms were evaluated with respect to the following variables:

- *Mean_Makespan:* average makespan considering the 20 runs for each dataset instance.
- *StdDev_Makespan:* standard deviation to the optimum value over the 20 runs for each dataset instance.
- *%Optimum:* percent of times where the algorithm finds the optimal makespan of the dataset over the 20 runs.
- *Execution_time:* average time used by the algorithm to execute 100 iterations.

In order to compare the algorithms, authors used *SPSS version 25* and the *Wilcoxon's non-parametric test* for two samples related, with 95% of confidence interval and 0.05 significance level.

In the comparison, the groups of algorithms were organized according to the quality of the results, that means "Group A results" > "Group B results" > "Group C results" > "Group D results". The algorithms in the same group did not have significant differences between them.

Table 1, 2, 9 and 10 show the descriptive analysis and the results of comparisons using Wilcoxon test for the *Mean_Makespan* variable.

For *StdDev Makespan* variable, comparison results are summarized in Table 3, 4, 11 and 12. Findings for the *%Optimum* variable are presented in Table 5, 6, 13 and 14.

Finally, Tables 7, 8, 15 and 16 show the same statistical analysis for the *Execution_time* variable.

Table 1. Descriptive analysis, variable *Mean_Makespan* for j30_17 dataset

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
CL_{FDA}_just_time	10	26.20	39.15	31.4150	1.21262	3.83464
Reported_Bibliography	10	26.00	39.00	31.7000	1.24766	3.94546
UMDA_just_time	10	26.40	39.05	31.8100	1.18528	3.74817
CL_{FDA}_pareto	10	28.65	39.85	33.2450	1.00657	3.18307
UMDA _pareto	10	29.10	40.55	33.5300	1.05998	3.35196
GA_just_time	10	28.10	39.45	33.8300	1.02746	3.24912
GA_pareto	10	29.65	40.40	35.0850	0.93634	2.96095

Table 2. Comparison results using Wilcoxon test, variable *Mean_Makespan* for j30_17 dataset

Algorithm1	Algorithm2	Signification *(p-value)*	Groups
CL_{FDA}_just_time	Reported_Bibliography	0.33	
CL_{FDA}_just_time	UMDA_just_time	**0.014**	*Group 1:* CL_{FDA}_just_time, Reported_ Bibliography
UMDA_just_time	CL_{FDA}_pareto	**0.037**	*Group 2:* UMDA_just_time
CL_{FDA}_pareto	UMDA_pareto	0.240	*Group 3:* CL_{FDA}_pareto, UMDA_pareto, GA_just_time
CL_{FDA}_pareto	GA_just_time	0.374	*Group 4:* GA_pareto
CL_{FDA}_pareto	GA_pareto	**0.005**	

Regarding variable *Mean_Makespan* in j30_17 dataset, there were not significant differences between *CL_{FDA}_just_time* and results reported in the bibliography. The best results were obtained by *CL_{FDA}_just_time,* whereas the worst results were obtained by *GA_pareto.*

Table 3. Descriptive analysis, variable *StdDev_Makespan* for j30_17 dataset

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
Reported_Bibliography	10	0.00	4.00	1.3000	0.36667	1.15950
CL_{FDA}_just_time	10	0.39	3.27	1.4998	0.30655	0.96940
UMDA_just_time	10	0.22	3.13	1.8599	0.28301	0.89495
CL_{FDA}_pareto	10	1.16	6.36	3.2956	0.44275	1.40008
UMDA _pareto	10	1.66	6.59	3.5752	0.45077	1.42547
GA_just_time	10	0.81	5.88	3.7808	0.48026	1.51872
GA_pareto	10	1.76	7.34	5.1077	0.49409	1.56244

Table 4. Comparison results using Wilcoxon test, variable *StdDev_Makespan* for j30_17 dataset

Algorithm1	Algorithm2	Signification (*p-value*)	Groups
Reported_Bibliography	CL_{FDA}_just_time	0.646	
Reported_Bibliography	UMDA_just_time	**0.037**	*Group 1:* Reported_ Bibliography, CL_{FDA}_just_time
UMDA_just_time	CL_{FDA}_pareto	**0.005**	*Group 2:* UMDA_just_time
CL_{FDA}_pareto	UMDA_pareto	0.202	*Group 3:* CL_{FDA}_pareto, UMDA_pareto, GA_just_time
CL_{FDA}_pareto	GA_just_time	0.241	*Group 4:* GA_pareto
CL_{FDA}_pareto	GA_pareto	**0.005**	

As regards variable *StdDev Makespan* in dataset j30_17, there are not significant difference between CL_{FDA}*_just_time* and results reported in the bibliography. These algorithms reported the best results; whereas the worst results were obtained by *GA_pareto* algorithm.

Table 5. Descriptive analysis, variable *%Optimum* for j30_17 dataset

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
CL_{FDA}_just_time	10	15.00	85.00	56.0000	8.81287	27.86874
UMDA_just_time	10	5.00	95.00	37.5000	7.82624	24.74874
GA_just_time	10	0.00	65.00	10.0000	6.32456	20.00000
CL_{FDA}_pareto	10	0.00	40.00	6.0000	3.92994	12.42757
UMDA _pareto	10	0.00	15.00	3.5000	1.97906	6.25833
GA_pareto	10	0.00	25.00	2.5000	2.50000	7.90569

Table 6. Comparison results using Wilcoxon test, variable *%Optimum*, j30_17 dataset

Algorithm1	Algorithm2	Signification (*p-value*)	Groups
CL_{FDA}_just_time	UMDA_just_time	**0.028**	*Group 1*: CL_{FDA}_just_time
UMDA_just_time	GA_just_time	**0.005**	*Group 2*: UMDA_just_time
GA_just_time	CL_{FDA}_pareto	0.161	*Group 3*: GA_just_time, CL_{FDA}_pareto, UMDA_pareto
GA_just_time	UMDA_pareto	0.572	
GA_just_time	GA_pareto	**0.042**	*Group 4*: GA_pareto

In dataset j30_17, concerning variable *%Optimum,* the best results were obtained by the algorithm CL_{FDA}*_just_time* and the worst results were obtained by *GA_pareto* algorithm. In this dataset, *just time* optimization strategy reports better results than *pareto* optimization.

Table 7. Descriptive analysis, variable *Execution_time*, j30_17 dataset

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
GA_just_time	10	16.55	20.42	18.7630	0.35353	1.11797
GA_pareto	10	16.92	20.95	19.2350	0.37585	1.18853
UMDA_just_time	10	24.02	29.05	26.9220	0.47009	1.48657
UMDA _pareto	10	26.88	33.48	29.4730	0.56849	1.79773
CL_{FDA}_just_time	10	37.49	42.91	40.2670	0.48567	1.53584
CL_{FDA}_pareto	10	39.98	46.25	42.4040	0.57165	1.80771

Table 8. Comparison results using Wilcoxon test, variable *Execution_time* (j30_17 dataset)

Algorithm1	Algorithm2	Signification (*p-value*)	Groups
GA_just_time	GA_pareto	**0.017**	*Group 1*: GA_just_time
GA_pareto	UMDA_just_time	**0.005**	*Group 2*: GA_pareto
UMDA_just_time	UMDA_pareto	**0.005**	*Group 3*: UMDA_just_time
UMDA_pareto	CL_{FDA}_just_time	**0.005**	*Group 4*: UMDA_pareto
CL_{FDA}_just_time	CL_{FDA}_pareto	**0.005**	*Group 5*: CL_{FDA}_just_time
			Group 6: CL_{FDA}_pareto,

In dataset J30_17, respect to variable *Execution time,* the best results were obtained by the *Genetic Algorithms* approach. In particular, *GA_just_time* was the fastest algorithm; whereas CL_{FDA} were the highest time consume algorithms.

Table 9. Descriptive analysis, variable *Mean_Makespan* (c15_9, c15_10 and c15_12 datasets)

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
Reported_Bibliography	30	12.00	39.00	22.1333	1.23338	6.75550
CL_{FDA}_pareto	30	14.70	38.45	22.4333	1.06220	5.81789
CL_{FDA}_just_time	30	14.30	40.10	22.5567	1.05235	5.76397
UMDA _pareto	30	15.90	40.90	23.0517	1.09634	6.00489
UMDA_just_time	30	16.20	42.05	23.3283	1.10486	6.05156
GA_just_time	30	17.10	42.10	23.5250	1.06103	5.81147
GA_pareto	30	16.50	41.40	23.8117	1.05626	5.78540

Table 10. Comparison results using Wilcoxon test, variable *Mean_Makespan* (c15_9, c15_10 and c15_12 datasets)

Algorithm1	Algorithm2	Signification (p-value)	Groups
CL_{FDA}_just_time	Reported_Bibliography	0.33	
CL_{FDA}_just_time	UMDA_just_time	**0.014**	*Group 1*: CL_{FDA}_just_time, Reported_ Bibliography
UMDA_just_time	CL_{FDA}_pareto	**0.037**	*Group 2*: UMDA_just_time
CL_{FDA}_pareto	UMDA_pareto	0.240	*Group 3*: CL_{FDA}_pareto, UMDA_pareto, GA_just_time
CL_{FDA}_pareto	GA_just_time	0.374	*Group 4*: GA_pareto
CL_{FDA}_pareto	GA_pareto	**0.005**	

As for variable *Mean Makespan* in dataset c15 with all 30 instances from c15_9, c15_10 and c15_12, there were not significant differences between CL_{FDA} *just time* and results reported in the bibliography. The best results were obtained by CL_{FDA} *just time and* the worst results were obtained by *GA_pareto.* EDA algorithms (CL_{FDA} and *UMDA*) using *just time* strategy reported better results than the same algorithms with the *pareto* optimization strategy.

Table 11. Descriptive analysis, variable *StdDev_Makespan* (c15_9, c15_10 and c15_12 datasets)

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
Reported_Bibliography	30	0.00	8.00	1.2000	0.33356	1.82700
CL_{FDA}_pareto	30	0.00	4.73	1.4962	0.23377	1.28044
CL_{FDA}_just_time	30	0.00	4.24	1.6584	0.26565	1.45501
UMDA _pareto	30	0.00	5.27	2.0655	0.31213	1.70962
UMDA_just_time	30	0.00	6.32	2.3699	0.36208	1.98317
GA_just_time	30	0.00	6.24	2.7220	0.31062	1.70133
GA_pareto	30	0.00	6.48	2.8390	0.29116	1.59476

Table 12. Comparison results using Wilcoxon test, variable *Stddev_Makespan* (c15_9, c15_10 and c15_12 datasets)

Algorithm1	Algorithm2	Signification (p-value)	Groups
Reported_Bibliography	CL_{FDA}_pareto	0.13	*Group 1*: Reported_ Bibliography, CL_{FDA}_pareto, CL_{FDA}_just_time
Reported_Bibliography	CL_{FDA}_just_time	0.092	
Reported_Bibliography	UMDA_pareto	**0.016**	*Group 2*: UMDA_pareto
UMDA_pareto	UMDA_just_time	**0.014**	*Group 3*: UMDA_just_time
UMDA_just_time	GA_just_time	**0.003**	
GA_just_time	GA_pareto	0.846	*Group 4*: GA_just_time, GA_pareto

In these datasets, for variable *StdDev Makespan*, the algorithms *Reported_ Bibliography, CL_{FDA}_pareto, CL_{FDA}_just_time* did not have significant differences, whereas the worst results were obtained by *GA* algorithms. In this case, differences between *just_time* and *pareto* optimization strategies were not found to be significant.

Table 13. Descriptive analysis variable *%Optimum* (c15_9, c15_10 and c15_12 datasets)

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
CL_{FDA}_pareto	30	0.00	100.00	52.6667	6.45379	35.34884
CL_{FDA}_just_time	30	0.00	100.00	50.0000	6.84391	37.48563
UMDA _pareto	30	0.00	100.00	41.3333	6.96846	38.16781
UMDA_just_time	30	0.00	100.00	38.0000	8.07508	44.22903
GA_just_time	30	0.00	100.00	24.3333	5.63616	30.87051
GA_pareto	30	0.00	100.00	17.8333	4.30639	23.58708

Table 14. Comparison results using Wilcoxon test, variable *%Optimum* (c15_9, c15_10 and c15_12 datasets)

Algorithm1	Algorithm2	Signification (p-value)	Groups
CL_{FDA}_pareto	CL_{FDA}_just_time	0.173	*Group 1*: CL_{FDA}_pareto, CL_{FDA}_just_time
CL_{FDA}_pareto	UMDA_pareto	**0.000**	*Group 2*: UMDA_ pareto, UMDA_ just_time
UMDA_pareto	UMDA_just_time	0.314	
UMDA_pareto	GA_just_time	**0.000**	*Group 3*: GA_just_time
GA_just_time	GA_pareto	**0.023**	*Group 4*: GA_pareto

In c15_9, c15_10 and c15_12 datasets and in relation to variable *%Optimum,* the best results were obtained by the algorithms *CL_{FDA}_pareto, CL_{FDA}_just_time,* whereas the worst results were obtained by *GA_pareto* algorithm. In this variable, as in *Mean Makespan,* there were not significant differences between *just time* and *pareto* strategies. The worst results were obtained by *GA* approach.

Table 15. Descriptive analysis variable *Execution_time* (c15_9, c15_10 and c15_12 datasets)

Algorithm	N Statistic	Minimum Statistic	Maximum Statistic	Mean Statistic	Std. Error	Std. Deviation Statistic
GA_pareto	30	7.29	12.22	9.0757	0.20911	1.14536
GA_just_time	30	7.56	12.79	9.2257	0.21902	1.19963
UMDA _pareto	30	10.37	16.57	12.4103	0.25598	1.40203
UMDA_just_time	30	10.35	17.47	12.4783	0.30701	1.68155
CL_{FDA}_just_time	30	16.08	24.12	18.9783	0.32382	1.77365
CL_{FDA}_pareto	30	16.31	23.27	19.0227	0.28145	1.54156

Table 16. Comparison results using Wilcoxon test, variable *Execution_time* (c15_9, c15_10 and c15_12 datasets)

Algorithm1	Algorithm2	Signification (p-value)	Groups
GA_pareto	GA_just_time	0.064	*Group 1*: GA_pareto, GA_just_time
GA_pareto	UMDA_pareto	**0.000**	
UMDA_pareto	UMDA_just_time	0.51	*Group 2*: UMDA_ pareto, UMDA_ just_time
UMDA_pareto	CL_{FDA}_just_time	**0.000**	
CL_{FDA}_just_time	CL_{FDA}_pareto	0.959	*Group 3*: CL_{FDA}_just_time, CL_{FDA}_pareto

In c15_9, c15_10 and c15_12 datasets, regarding variable *Execution_time,* the best results were obtained by *Genetic Algorithms* approach, whereas the CL_{FDA} approach used a higher time for the same stop criterion (100 generations). This is because CL_{FDA} algorithms spend more time to detect the dependency relation among variables, but it is able to find solutions which have never been found by *GA* or *UMDA*. Furthermore, the objective is to minimize the makespan of the project, not to minimize the execution time of the algorithms.

5. Conclusion

In this paper, a new approach on Estimation of Distribution Algorithms with constraints handling inside the probabilistic model to solve the Multi-mode Resource Constrained Project Scheduling Problems was developed. The proposal was applied to four datasets of PSPLIB in its multi-mode variant, which have several complexity degrees (task numbers, number of modes, and number of resources). The cost component to be optimized along with time was added, always looking for a balance between them.

The obtained results prove to be very effective on the benchmark instances and improved others reported in the bibliography, especially in j30_17, c15_9 and c15_12 datasets.

Overall, CL_{FDA}*_just_time* has been selected as the best algorithm. It achieved the best results for the *Mean_Makespan, StdDev_MakeSpan* and *%Optimum* variables.

Towards further improvement, the flexibility of adding different components like project quality to the model makes the procedure particularly useful. Moreover, further research on different strategies to diversify the search process can lead to a superior performance of the algorithm.

References

[1] INFANTE, A. (2008) Teamsoft: Sistema Para La Gestión Del Trabajo En Equipo En El Desarrollo de Proyectos de Software, Version 2.0, Universidad de las Ciencias Informáticas, La Habana.

[2] AWASTHI, A. (2016) Optimization of NP-Hard Scheduling Problems by Developing Timing Algorithms and Parallelization, PhD Thesis, Carl von Ossietzky Universität Oldenburg, Germany 8–48 pp.

[3] DUNBAR, G. (2015) Project Management Failures - Standish (Chaos) Reports (1994-2015), https://www.linkedin.com/pulse/project-management-failures-standish-chaos-report-2015-dunbar.

[4] JOHNSON, J. (2018) CHAOS Report: Decision Latency Theory: It Is All About the Interval, Second Edition., The Standish Group 70 pp.

[5] VILLAVICENCIO, N. (2016) Modelo Integrado Para La Mejora de La Productividad En Organizaciones Orientadas a Proyectos de Tecnologías de La Información, PhD Thesis, Tesis para optar al grado de: Máster en Diseño, Gestión, Funiber.

[6] AMOUI, M., DERAKHSHANMANESH, M., EBERT, J., TAHVILDARI, L. (2012) Achieving dynamic adaptation via management and interpretation of runtime models, Journal of Systems and Software **85** 12 2720.

[7] PIÑERO, P.Y. (2013) Paquete para la Dirección Integrada de Proyectos y ayuda a la toma de decisiones: GESPRO, Informática 2223.

[8] MOSSALAM, A., ARAFA, M. (2016) The role of project manager in benefits realization management as a project constraint/driver, HBRC Journal **12** 3 305.

[9] BERMÚDEZ, N.V., ABREU, M.P., VALARESO, S.B., PUPO, I.P. (2016) Experiencias en la integración de procesos en las organizaciones orientadas a proyectos de software, Revista Cubana de Ciencias Informáticas **10** 171.

[10] PACELLI, L. (2004) Grandes errores en la gestión de proyectos, Finantial Times Prenice Hall, ISBN **131490478** 167.

[11] EQUIPO DEL PRODUCTO CMMI, CMMI® Para Desarrollo (2010) Versión 1.3, Technical Report CMU/SEI-2010-TR-033 ESC-TR-2010-033, Editorial Universitaria Ramón Areces, Software Engineering Process Management Program, Carnegie Mellon University.

[12] CHAUDHARY, M., CHOPRA, A. (2017) CMMI for Development - Implementation Guide, Apress.

[13] PROJECT MANAGEMENT INSTITUTE, INC (2017) A Guide to the Project Management Body of Knowledge (PMBOK Guide), Sixth edition., Project Management Institute, Newtown Square, Pennsylvania 19073-3299 USA.

[14] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, ISO 21500 (2012): Guidance on Project Management, Geneva.

[15] BRIOSO, X. (2015) Integrating ISO 21500 guidance on project management, lean construction and PMBOK, Procedia Engineering **123** 76.

[16] VACACELA, R.G., PUPO, I.P., VILLAVICENCIO, N., PEREZ, P.Y.P., LUIS, S.B. (2016) Experiencias usando algoritmos genéticos en la planificación de proyectos, Revista Cubana de Ciencias Informáticas **10** 71.

[17] BRUCKER, P., DREXL, A., MÖHRING, R., NEUMANN, K., PESCH, E. (1999) Resource-constrained project scheduling: Notation, classification, models, and methods, European journal of operational research **112** 1 3.

[18] BROWNING, T.R., YASSINE, A.A. (2010) Resource-constrained multi-project scheduling: Priority rule performance revisited, International Journal of Production Economics **126** 2 212.

[19] AYODELE, M., MCCALL, J., REGNIER-COUDERT, O. (2016) BPGA-EDA for the Multi-Mode Resource Constrained Project Scheduling Problem, Evolutionary Computation (CEC), 2016 IEEE Congress On, IEEE 3417–3424.

[20] AYODELE, M., MCCALL, J., REGNIER-COUDERT, O. (2017) Estimation of Distribution Algorithms for the Multi-Mode Resource Constrained Project Scheduling Problem, 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, Donostia, San Sebastián, Spain 1579–1586.

[21] PINHA, D.C., AHLUWALIA, R.S., SENNA, P. (2016) The Combinatorial Multi-Mode Resource Constrained Multi-Project Scheduling Problem, International Journal of Supply and Operations Managment **3** 3 1391.

[22] ZHANG, H., LI, H., TAM, C.M. (2006) Permutation-based particle swarm optimization for resource-constrained project scheduling, Journal of Computing in Civil Engineering **20** 2 141.

[23] MEDRANO, B.E. (2013) Planificación de múltiples proyectos de desarrollo de software utilizando métodos metaheurísticos, Master Thesis, Universidad de La Habana, La Habana, 16–50 pp.

[24] BAAR, T., BRUCKER, P., KNUST, S. (1999) "Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project Scheduling Problem", Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization (VOß, S., MARTELLO, S., OSMAN, I.H., ROUCAIROL, C., Eds), Springer US, Boston, MA 1–18.

[25] ARTIGUES, C., MICHELON, P., REUSSER, S. (2003) Insertion techniques for static and dynamic resource-constrained project scheduling, European Journal of Operational Research **149** 2 249.

[26] DONG, N., GE, D., FISCHER, M., HADDAD, Z. (2012) A genetic algorithm-based method for look-ahead scheduling in the finishing phase of construction projects, Advanced Engineering Informatics **26** 4 737.

[27] WANG, W., WANG, X., GE, X., DENG, L. (2014) Multi-objective optimization model for multi-project scheduling on critical chain, Advances in Engineering Software **68** 33.

[28] KUEHN, M., ZAHID, T., VOELKER, M., ZHOU, Z., ROSE, O. (2016) Investigation Of Genetic Operators And Priority Heuristics for Simulation Based Optimization Of Multi-Mode Resource Constrained Multi-Project Scheduling Problems (MMRCMPSP), ECMS 2016 Proceedings Edited by Thorsten Claus, Frank Herrmann, Michael Manitz, Oliver Rose, ECMS 481–487.

[29] AYODELE, M., MCCALL, J., REGNIER-COUDERT, O. (2016) RK-EDA: A Novel Random Key Based Estimation of Distribution Algorithm, Springer 849–858.

[30] AYODELE, M., MCCALL, J., REGNIER-COUDERT, O. (2017) Estimation of Distribution Algoprithms for the Multi-Mode Resource Constrained Project Scheduling Problem, 1579.

[31] TALBOT, F.B., Resource-Constrained Project Scheduling with Time-Resource Trade offs: The Nonpreemptive Case, Management Science **28** 10 (1982) 1197.

[32] KOLISCH, R., SPRECHER, A. (1997) PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program, European Journal of Operational Research **96** 1 205.

[33] MÜHLENBEIN, H., MAHNIG, T., OCHOA-RODRÍGUEZ, A. (1999) Schemata, Distributions and Graphical Models in Evolutionary Optimization, J. Heuristics **5** 2 215.

[34] CHEN, Y. et al. (2017) Personalized Search Inspired Fast Interactive Estimation of Distribution Algorithm and Its Application, IEEE Transactions on Evolutionary Computation **21** 4 588.

[35] ECHEGARAY, T.M. (2010) Avances en Algoritmos de Estimación de Distribuciones. Alternativas en el Aprendizaje y Representación de Problemas, Ph. D. dissertation, Universidad del País Vasco, España.

[36] RODRIGUEZ, R.P. et al. (2016). Un algoritmo de estimación de distribuciones para el problema de ruteo de autobuses escolares con selección de paradas, DYNA New Technologies **3** 1.

[37] MÜHLENBEIN, H., MAHNIG, T. (1999) Convergence Theory and Applications of the Factorized Distribution Algorithm, Journal of Computing and Information Technology **7** 19.

[38] EMMERICH, M.T.M., DEUTZ, A.H. (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods, Natural Computing **17** 3 585.

[39] Multi Mode Data Sets, http://www.om-db.wi.tum.de/psplib/getdata.cgi?mode=mm.